```
**********************************************************
     703 Tue Jan 14 20:13:01 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggencoding.d
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 */

#pragma D option quiet
#pragma D option encoding=utf8
#pragma D option aggzoom

tick-1ms
/i++ < 320/
{
        @ = lquantize(i, 0, 640, 1, i);
        @ = lquantize(641 - i, 0, 640, 1, i);
}

tick-1ms
/i == 320/
{
        printa(@);
        exit(0);
}
#endif /* ! codereview */
```

```
**********************************************************
     998 Tue Jan 14 20:13:01 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.agghist.d
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 */

#pragma D option agghist
#pragma D option quiet

BEGIN
{
        @["demerit"] = sum(-10);
        @["wtf"] = sum(10);
        @["bot"] = sum(20);

        @bagnoogle["SOAP/XML"] = sum(1);
        @bagnoogle["XACML store"] = sum(5);
        @bagnoogle["SAML token"] = sum(6);

        @stalloogle["breakfast"] = sum(-5);
        @stalloogle["non-diet pepsi"] = sum(-20);
        @stalloogle["parrot"] = sum(-100);

        printa(@);
        printa(@bagnoogle);
        printa(@stalloogle);

        printf("\nzoomed:");

        setopt("aggzoom");
        printa(@);
        printa(@bagnoogle);
        printa(@stalloogle);

        exit(0);
}

#endif /* ! codereview */
```

```
**********************************************************
    1701 Tue Jan 14 20:13:02 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.agghist.d.out
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************


               key  ------------- Distribution ------------- count
          demerit                @@@@@|                     -10
              wtf                     |@@@@@                10
              bot                     |@@@@@@@@@@           20


               key  ------------- Distribution ------------- count
         SOAP/XML |@@@                                      1
       XACML store |@@@@@@@@@@@@@@@@@@                       5
        SAML token |@@@@@@@@@@@@@@@@@@@@@@                   6


               key  ------------- Distribution ------------- count
            parrot         @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@| -100
    non-diet pepsi                                   @@@@@@| -20
         breakfast                                       @@| -5

zoomed:

               key  ------------- Distribution ------------- count
          demerit           @@@@@@@@@@|                     -10
              wtf                     |@@@@@@@@@@           10
              bot                     |@@@@@@@@@@@@@@@@@@@@  20


               key  ------------- Distribution ------------- count
         SOAP/XML |@@@@@@@                                  1
       XACML store |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  5
        SAML token |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 6


               key  ------------- Distribution ------------- count
            parrot   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@| -100
    non-diet pepsi                                 @@@@@@@@| -20
         breakfast                                       @@| -5

#endif /* ! codereview */
```

****
1032 Tue Jan 14 20:13:02 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpack.d
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
****

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 */

#pragma D option aggpack
#pragma D option encoding=ascii
#pragma D option quiet

BEGIN
{
        @x = quantize(1 << 32);
        @y[1] = quantize(1);
        @z["mumble"] = quantize(1);
        @xx["foo", (char)1, (short)2, (long)3] = quantize(1);

        @neg = lquantize(-10, -10, 20, 1, -1);
        @neg = lquantize(-5, -10, 20, 1, 1);
        @neg = lquantize(0, -10, 20, 1, 1);

        i = 0;
}

tick-1ms
{
        @a[i] = quantize(0, i);
        @a[i] = quantize(1, 100 - i);
        i++;
}

tick-1ms
/i > 100/
{
        exit(0);
}

END
{
        setopt("aggzoom", "true");
        printa(@neg);
        setopt("aggzoom", "false");
        printa(@neg);
}
#endif /* ! codereview */
```

```
**********************************************************
    3860 Tue Jan 14 20:13:02 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpack.d.out
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************


  3      min .---------------------------------. max      | count
  4    < -10 : V    X    X                    : >= 20    | 1


  7      min .---------------------------------. max      | count
  8    < -10 : v    x    x                    : >= 20    | 1


 11         min .---. max        | count
 12  2147483648 : X : 8589934592 | 1

 14       key  min .---. max  | count
 15         1    0 : X : 2    | 1

 17               key  min .---. max  | count
 18            mumble    0 : X : 2    | 1

 20                                        key  min .---. max  | count
 21               foo   1     2              3    0 : X : 2    | 1

 23       key  min .---. max  | count
 24       100    0 :X  : 2    | 100
 25        99    0 :X_ : 2    | 100
 26        98    0 :X_ : 2    | 100
 27        97    0 :X_ : 2    | 100
 28        96    0 :X_ : 2    | 100
 29        95    0 :X_ : 2    | 100
 30        94    0 :X_ : 2    | 100
 31        93    0 :X_ : 2    | 100
 32        92    0 :X_ : 2    | 100
 33        91    0 :X_ : 2    | 100
 34        90    0 :X_ : 2    | 100
 35        89    0 :X_ : 2    | 100
 36        88    0 :X_ : 2    | 100
 37        87    0 :X_ : 2    | 100
 38        86    0 :X_ : 2    | 100
 39        85    0 :X_ : 2    | 100
 40        84    0 :X_ : 2    | 100
 41        83    0 :X_ : 2    | 100
 42        82    0 :X_ : 2    | 100
 43        81    0 :X_ : 2    | 100
 44        80    0 :X_ : 2    | 100
 45        79    0 :X_ : 2    | 100
 46        78    0 :xx : 2    | 100
 47        77    0 :xx : 2    | 100
 48        76    0 :xx : 2    | 100
 49        75    0 :xx : 2    | 100
 50        74    0 :xx : 2    | 100
 51        73    0 :xx : 2    | 100
 52        72    0 :xx : 2    | 100
 53        71    0 :xx : 2    | 100
 54        70    0 :xx : 2    | 100
 55        69    0 :xx : 2    | 100
 56        68    0 :xx : 2    | 100
 57        67    0 :xx : 2    | 100
 58        66    0 :xx : 2    | 100
 59        65    0 :xx : 2    | 100
```



```
 60        64    0 :xx : 2    | 100
 61        63    0 :xx : 2    | 100
 62        62    0 :xx : 2    | 100
 63        61    0 :xx : 2    | 100
 64        60    0 :xx : 2    | 100
 65        59    0 :xx : 2    | 100
 66        58    0 :xx : 2    | 100
 67        57    0 :xx : 2    | 100
 68        56    0 :xx : 2    | 100
 69        55    0 :xx : 2    | 100
 70        54    0 :xx : 2    | 100
 71        53    0 :xx : 2    | 100
 72        52    0 :xx : 2    | 100
 73        51    0 :xx : 2    | 100
 74        50    0 :xx : 2    | 100
 75        49    0 :xx : 2    | 100
 76        48    0 :xx : 2    | 100
 77        47    0 :xx : 2    | 100
 78        46    0 :xx : 2    | 100
 79        45    0 :xx : 2    | 100
 80        44    0 :xx : 2    | 100
 81        43    0 :xx : 2    | 100
 82        42    0 :xx : 2    | 100
 83        41    0 :xx : 2    | 100
 84        40    0 :xx : 2    | 100
 85        39    0 :xx : 2    | 100
 86        38    0 :xx : 2    | 100
 87        37    0 :xx : 2    | 100
 88        36    0 :xx : 2    | 100
 89        35    0 :xx : 2    | 100
 90        34    0 :xx : 2    | 100
 91        33    0 :xx : 2    | 100
 92        32    0 :xx : 2    | 100
 93        31    0 :xx : 2    | 100
 94        30    0 :xx : 2    | 100
 95        29    0 :xx : 2    | 100
 96        28    0 :xx : 2    | 100
 97        27    0 :xx : 2    | 100
 98        26    0 :xx : 2    | 100
 99        25    0 :xx : 2    | 100
100        24    0 :xx : 2    | 100
101        23    0 :xx : 2    | 100
102        22    0 :xx : 2    | 100
103        21    0 :_X : 2    | 100
104        20    0 :_X : 2    | 100
105        19    0 :_X : 2    | 100
106        18    0 :_X : 2    | 100
107        17    0 :_X : 2    | 100
108        16    0 :_X : 2    | 100
109        15    0 :_X : 2    | 100
110        14    0 :_X : 2    | 100
111        13    0 :_X : 2    | 100
112        12    0 :_X : 2    | 100
113        11    0 :_X : 2    | 100
114        10    0 :_X : 2    | 100
115         9    0 :_X : 2    | 100
116         8    0 :_X : 2    | 100
117         7    0 :_X : 2    | 100
118         6    0 :_X : 2    | 100
119         5    0 :_X : 2    | 100
120         4    0 :_X : 2    | 100
121         3    0 :_X : 2    | 100
122         2    0 :_X : 2    | 100
123         1    0 :_X : 2    | 100
124         0    0 : X : 2    | 100
125 #endif /* ! codereview */
```

```
**********************************************************
    1611 Tue Jan 14 20:13:02 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpackbanner.ksh
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright (c) 2013 Joyent, Inc.  All rights reserved.
#

let width=8

function outputchar
{
        banner $3 | /bin/nawk -v line=$1 -v pos=$2 -v width=$width '{ \
                for (i = 1; i <= length($0); i++) { \
                        if (substr($0, i, 1) == " ") \
                                continue; \
                        printf("\t@letter%d[%d] = lquantize(%d, 0, 40, 1);\n", \
                            line, NR, i + (pos * width));
                } \
        }'
}

function outputstr
{
        let pos=0;
        let line=0

        printf "#pragma D option aggpack\n#pragma D option aggsortkey\n"

        printf "BEGIN\n{\n"
        for c in `echo "$1" | /bin/nawk '{ \
                for (i = 1; i <= length($0); i++) { \
                        c = substr($0, i, 1); \
                        printf("%s\n", c == " " ? "space" : \
                            c == "\n" ? "newline" : c); \
                } \
        }'`; do
                if [[ "$c" == "space" ]]; then
                        let line=line+1
                        let pos=0
                        continue
                fi

                outputchar $line $pos $c
                let pos=pos+1
        done

        let i=0

        while [[ $i -le $line ]]; do
                printf "\tprinta(@letter%d);\n" $i
                let i=i+1
        done
        printf "\texit(0);\n}\n"
}

dtrace -qs /dev/stdin -x encoding=utf8 <<EOF
`outputstr "why must i do this"`
EOF

dtrace -qs /dev/stdin -x encoding=ascii -x aggzoom <<EOF
`outputstr "i am not well"`
EOF

dtrace -qs /dev/stdin -x encoding=utf8 -x aggzoom <<EOF
`outputstr "send help"`
EOF

#endif /* ! codereview */
```

```
**********************************************************
    6569 Tue Jan 14 20:13:02 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpackbanner.ksh.out
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************


  3       key     min .---------------------------------------. max      | count
  4         2     < 0 :   â      â    â      â     â    â                  : >= 40
  5         3     < 0 :   â      â    â      â      â   â                  : >= 40
  6         4     < 0 :   â      â    â  â  â  â  â  â      â               :
  7         5     < 0 :   â   â  â    â    â      â      â                   : >=
  8         6     < 0 :   â  â    â  â    â      â      â                   : >=
  9         7     < 0 :   â      â    â      â      â                  : >= 40


 12       key     min .---------------------------------------. max      | count
 13         2     < 0 :   â      â    â      â     â  â  â  â     â  â  â  â  â
 14         3     < 0 :   â  â    â  â    â      â    â             â           : >
 15         4     < 0 :   â   â  â  â    â      â     â  â  â  â       â
 16         5     < 0 :   â      â    â      â       â      â                  : >= 40
 17         6     < 0 :   â      â    â      â    â      â     â              : >=
 18         7     < 0 :   â      â     â  â  â  â     â  â  â  â       â


 21       key     min .---------------------------------------. max      | count
 22         2     < 0 :      â                                  : >= 40    | 1
 23         3     < 0 :      â                                  : >= 40    | 1
 24         4     < 0 :      â                                  : >= 40    | 1
 25         5     < 0 :      â                                  : >= 40    | 1
 26         6     < 0 :      â                                  : >= 40    | 1
 27         7     < 0 :      â                                  : >= 40    | 1


 30       key     min .---------------------------------------. max      | count
 31         2     < 0 :   â  â  â  â  â     â  â  â  â                      :
 32         3     < 0 :   â     â    â     â                          : >= 40
 33         4     < 0 :   â     â    â     â                          : >= 40
 34         5     < 0 :   â     â    â     â                          : >= 40
 35         6     < 0 :   â     â    â     â                          : >= 40
 36         7     < 0 :   â  â  â  â  â     â  â  â  â                      :


 39       key     min .---------------------------------------. max      | count
 40         2     < 0 :    â  â  â  â  â    â      â      â      â  â  â  â
 41         3     < 0 :      â     â     â     â     â          : >= 40
 42         4     < 0 :      â     â  â  â  â  â     â      â  â  â  â
 43         5     < 0 :      â     â     â     â        â          : >= 40
 44         6     < 0 :      â     â     â     â     â     â          : >= 40
 45         7     < 0 :      â     â     â     â      â  â  â  â          : >


 49       key     min .---------------------------------------. max      | count
 50         2     < 0 :      X                                  : >= 40    | 1
 51         3     < 0 :      X                                  : >= 40    | 1
 52         4     < 0 :      X                                  : >= 40    | 1
 53         5     < 0 :      X                                  : >= 40    | 1
 54         6     < 0 :      X                                  : >= 40    | 1
 55         7     < 0 :      X                                  : >= 40    | 1


 58       key     min .---------------------------------------. max      | count
 59         2     < 0 :     XX    X    X                       : >= 40    | 4
```



```
 60         3     < 0 :     X  X   XX  XX                      : >= 40    | 6
 61         4     < 0 :   X    X  X XX X                      : >= 40    | 6
 62         5     < 0 :   XXXXXX  X    X                     : >= 40    | 8
 63         6     < 0 :   X    X  X    X                     : >= 40    | 4
 64         7     < 0 :   X    X  X    X                     : >= 40    | 4


 67       key     min .---------------------------------------. max      | count
 68         2     < 0 :   X    X   XXXX    XXXXX              : >= 40    | 11
 69         3     < 0 :   XX   X  X    X     X                : >= 40    | 6
 70         4     < 0 :   X X  X  X    X     X                : >= 40    | 6
 71         5     < 0 :   X  X X  X    X     X                : >= 40    | 6
 72         6     < 0 :   X   XX  X    X     X                : >= 40    | 6
 73         7     < 0 :   X    X   XXXX     X                : >= 40    | 7


 76       key     min .---------------------------------------. max      | count
 77         2     < 0 :   X    X  XXXXXX  X       X           : >= 40    | 10
 78         3     < 0 :   X    X  X       X       X           : >= 40    | 5
 79         4     < 0 :   X    X  XXXXX   X       X           : >= 40    | 9
 80         5     < 0 :   X XX X  X       X       X           : >= 40    | 7
 81         6     < 0 :   XX  XX  X       X       X           : >= 40    | 7
 82         7     < 0 :   X    X  XXXXXX  XXXXXX  XXXXXX      : >= 40    | 20


 86       key     min .---------------------------------------. max      | count
 87         2     < 0 :    â  â  â  â     â  â  â  â  â  â  â     â  â  â  â  â  â
 88         3     < 0 :   â        â        â  â     â    â     â         : >=
 89         4     < 0 :    â  â  â  â     â  â  â  â  â  â   â   â   â    â     â
 90         5     < 0 :       â   â       â   â  â   â     â         : >=
 91         6     < 0 :   â     â   â       â    â  â   â    â         : >
 92         7     < 0 :    â  â  â  â     â  â  â  â  â  â   â    â   â  â  â  â


 95       key     min .---------------------------------------. max      | count
 96         2     < 0 :   â      â    â  â  â  â  â  â    â       â  â  â  â  â
 97         3     < 0 :   â      â    â       â      â     â          : >= 40
 98         4     < 0 :   â  â  â  â  â  â    â  â  â  â  â    â       â     â
 99         5     < 0 :   â      â    â       â      â  â  â  â  â          :
100         6     < 0 :   â      â    â       â      â                : >= 40
101         7     < 0 :   â      â    â  â  â  â  â  â   â  â  â  â  â  â   â

103 #endif /* ! codereview */
```

```
**********************************************************
    1003 Tue Jan 14 20:13:03 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpackzoom.d
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
 */

#pragma D option encoding=ascii
#pragma D option quiet

tick-1ms
/i++ < 30/
{
        @[1] = lquantize(i, 0, 40, 1, 1000);
        @[2] = lquantize(i, 0, 40, 1, 1000);
        @[3] = lquantize(i, 0, 40, 1, 1000);
}

tick-1ms
/i == 40/
{
        @[1] = lquantize(0, 0, 40, 1, 1);
        @[1] = lquantize(i, 0, 40, 1, 2000);
        @[2] = lquantize(0, 0, 40, 1, 1);
        @[2] = lquantize(i, 0, 40, 1, 2000);
        @[3] = lquantize(0, 0, 40, 1, 1);
        @[3] = lquantize(i, 0, 40, 1, 2000);

        printa(@);
        setopt("aggpack");
        printa(@);
        setopt("aggzoom");
        printa(@);
        exit(0);
}
#endif /* ! codereview */
```

```
**********************************************************
    9580 Tue Jan 14 20:13:03 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggpackzoom.d.out
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************

        1
           value  ------------- Distribution ------------- count
             < 0 |                                         0
               0 |                                         1
               1 |@                                        1000
               2 |@                                        1000
               3 |@                                        1000
               4 |@                                        1000
               5 |@                                        1000
               6 |@                                        1000
               7 |@                                        1000
               8 |@                                        1000
               9 |@                                        1000
              10 |@                                        1000
              11 |@                                        1000
              12 |@                                        1000
              13 |@                                        1000
              14 |@                                        1000
              15 |@                                        1000
              16 |@                                        1000
              17 |@                                        1000
              18 |@                                        1000
              19 |@                                        1000
              20 |@                                        1000
              21 |@                                        1000
              22 |@                                        1000
              23 |@                                        1000
              24 |@                                        1000
              25 |@                                        1000
              26 |@                                        1000
              27 |@                                        1000
              28 |@                                        1000
              29 |@                                        1000
              30 |@                                        1000
              31 |                                         0
              32 |                                         0
              33 |                                         0
              34 |                                         0
              35 |                                         0
              36 |                                         0
              37 |                                         0
              38 |                                         0
              39 |                                         0
           >= 40 |@@                                       2000

        2
           value  ------------- Distribution ------------- count
             < 0 |                                         0
               0 |                                         1
               1 |@                                        1000
               2 |@                                        1000
               3 |@                                        1000
               4 |@                                        1000
               5 |@                                        1000
               6 |@                                        1000
               7 |@                                        1000
               8 |@                                        1000
               9 |@                                        1000
```



```
              10 |@                                        1000
              11 |@                                        1000
              12 |@                                        1000
              13 |@                                        1000
              14 |@                                        1000
              15 |@                                        1000
              16 |@                                        1000
              17 |@                                        1000
              18 |@                                        1000
              19 |@                                        1000
              20 |@                                        1000
              21 |@                                        1000
              22 |@                                        1000
              23 |@                                        1000
              24 |@                                        1000
              25 |@                                        1000
              26 |@                                        1000
              27 |@                                        1000
              28 |@                                        1000
              29 |@                                        1000
              30 |@                                        1000
              31 |                                         0
              32 |                                         0
              33 |                                         0
              34 |                                         0
              35 |                                         0
              36 |                                         0
              37 |                                         0
              38 |                                         0
              39 |                                         0
           >= 40 |@@                                       2000

        3
           value  ------------- Distribution ------------- count
             < 0 |                                         0
               0 |                                         1
               1 |@                                        1000
               2 |@                                        1000
               3 |@                                        1000
               4 |@                                        1000
               5 |@                                        1000
               6 |@                                        1000
               7 |@                                        1000
               8 |@                                        1000
               9 |@                                        1000
              10 |@                                        1000
              11 |@                                        1000
              12 |@                                        1000
              13 |@                                        1000
              14 |@                                        1000
              15 |@                                        1000
              16 |@                                        1000
              17 |@                                        1000
              18 |@                                        1000
              19 |@                                        1000
              20 |@                                        1000
              21 |@                                        1000
              22 |@                                        1000
              23 |@                                        1000
              24 |@                                        1000
              25 |@                                        1000
              26 |@                                        1000
              27 |@                                        1000
              28 |@                                        1000
              29 |@                                        1000
              30 |@                                        1000
```

```
126              31 |                                         0
127              32 |                                         0
128              33 |                                         0
129              34 |                                         0
130              35 |                                         0
131              36 |                                         0
132              37 |                                         0
133              38 |                                         0
134              39 |                                         0
135           >= 40 |@@                                       2000



139       key     min .-----------------------------------------. max      | count
140         1     < 0 : _____________________________________         _: >= 40    | 32001
141         2     < 0 : _____________________________________         _: >= 40    | 32001
142         3     < 0 : _____________________________________         _: >= 40    | 32001


145       key     min .-----------------------------------------. max      | count
146         1     < 0 : _xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx         X: >= 40    | 32001
147         2     < 0 : _xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx         X: >= 40    | 32001
148         3     < 0 : _xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx         X: >= 40    | 32001

150 #endif /* ! codereview */
```

```
**********************************************************
     751 Tue Jan 14 20:13:03 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggzoom.d
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright (c) 2013 Joyent, Inc.  All rights reserved.
  14  */

  16 #pragma D option encoding=ascii
  17 #pragma D option quiet

  19 tick-1ms
  20 /i++ < 90/
  21 {
  22         @ = lquantize(i, 0, 100, 1, 1000);
  23 }

  25 tick-1ms
  26 /i == 100/
  27 {
  28         @ = lquantize(i++, 0, 100, 1, 2000);
  29         @ = lquantize(i++, 0, 100, 1, 3000);

  31         printa(@);
  32         setopt("aggzoom");
  33         printa(@);
  34         exit(0);
  35 }
  36 #endif /* ! codereview */
```

```
**********************************************************
   14083 Tue Jan 14 20:13:03 2014
new/usr/src/cmd/dtrace/test/tst/common/aggs/tst.aggzoom.d.out
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************


           value  ------------- Distribution ------------- count
               0 |                                         0
               1 |                                         1000
               2 |                                         1000
               3 |                                         1000
               4 |                                         1000
               5 |                                         1000
               6 |                                         1000
               7 |                                         1000
               8 |                                         1000
               9 |                                         1000
              10 |                                         1000
              11 |                                         1000
              12 |                                         1000
              13 |                                         1000
              14 |                                         1000
              15 |                                         1000
              16 |                                         1000
              17 |                                         1000
              18 |                                         1000
              19 |                                         1000
              20 |                                         1000
              21 |                                         1000
              22 |                                         1000
              23 |                                         1000
              24 |                                         1000
              25 |                                         1000
              26 |                                         1000
              27 |                                         1000
              28 |                                         1000
              29 |                                         1000
              30 |                                         1000
              31 |                                         1000
              32 |                                         1000
              33 |                                         1000
              34 |                                         1000
              35 |                                         1000
              36 |                                         1000
              37 |                                         1000
              38 |                                         1000
              39 |                                         1000
              40 |                                         1000
              41 |                                         1000
              42 |                                         1000
              43 |                                         1000
              44 |                                         1000
              45 |                                         1000
              46 |                                         1000
              47 |                                         1000
              48 |                                         1000
              49 |                                         1000
              50 |                                         1000
              51 |                                         1000
              52 |                                         1000
              53 |                                         1000
              54 |                                         1000
              55 |                                         1000
```



```
              56 |                                         1000
              57 |                                         1000
              58 |                                         1000
              59 |                                         1000
              60 |                                         1000
              61 |                                         1000
              62 |                                         1000
              63 |                                         1000
              64 |                                         1000
              65 |                                         1000
              66 |                                         1000
              67 |                                         1000
              68 |                                         1000
              69 |                                         1000
              70 |                                         1000
              71 |                                         1000
              72 |                                         1000
              73 |                                         1000
              74 |                                         1000
              75 |                                         1000
              76 |                                         1000
              77 |                                         1000
              78 |                                         1000
              79 |                                         1000
              80 |                                         1000
              81 |                                         1000
              82 |                                         1000
              83 |                                         1000
              84 |                                         1000
              85 |                                         1000
              86 |                                         1000
              87 |                                         1000
              88 |                                         1000
              89 |                                         1000
              90 |                                         1000
              91 |                                         0
              92 |                                         0
              93 |                                         0
              94 |                                         0
              95 |                                         0
              96 |                                         0
              97 |                                         0
              98 |                                         0
              99 |                                         0
         >= 100 |@@                                       5000


           value  ------------- Distribution ------------- count
               0 |                                         0
               1 |@@@@@@@@@                                1000
               2 |@@@@@@@@@                                1000
               3 |@@@@@@@@@                                1000
               4 |@@@@@@@@@                                1000
               5 |@@@@@@@@@                                1000
               6 |@@@@@@@@@                                1000
               7 |@@@@@@@@@                                1000
               8 |@@@@@@@@@                                1000
               9 |@@@@@@@@@                                1000
              10 |@@@@@@@@@                                1000
              11 |@@@@@@@@@                                1000
              12 |@@@@@@@@@                                1000
              13 |@@@@@@@@@                                1000
              14 |@@@@@@@@@                                1000
              15 |@@@@@@@@@                                1000
              16 |@@@@@@@@@                                1000
```

```
126              17 |@@@@@@@@@                                1000
127              18 |@@@@@@@@@                                1000
128              19 |@@@@@@@@@                                1000
129              20 |@@@@@@@@@                                1000
130              21 |@@@@@@@@@                                1000
131              22 |@@@@@@@@@                                1000
132              23 |@@@@@@@@@                                1000
133              24 |@@@@@@@@@                                1000
134              25 |@@@@@@@@@                                1000
135              26 |@@@@@@@@@                                1000
136              27 |@@@@@@@@@                                1000
137              28 |@@@@@@@@@                                1000
138              29 |@@@@@@@@@                                1000
139              30 |@@@@@@@@@                                1000
140              31 |@@@@@@@@@                                1000
141              32 |@@@@@@@@@                                1000
142              33 |@@@@@@@@@                                1000
143              34 |@@@@@@@@@                                1000
144              35 |@@@@@@@@@                                1000
145              36 |@@@@@@@@@                                1000
146              37 |@@@@@@@@@                                1000
147              38 |@@@@@@@@@                                1000
148              39 |@@@@@@@@@                                1000
149              40 |@@@@@@@@@                                1000
150              41 |@@@@@@@@@                                1000
151              42 |@@@@@@@@@                                1000
152              43 |@@@@@@@@@                                1000
153              44 |@@@@@@@@@                                1000
154              45 |@@@@@@@@@                                1000
155              46 |@@@@@@@@@                                1000
156              47 |@@@@@@@@@                                1000
157              48 |@@@@@@@@@                                1000
158              49 |@@@@@@@@@                                1000
159              50 |@@@@@@@@@                                1000
160              51 |@@@@@@@@@                                1000
161              52 |@@@@@@@@@                                1000
162              53 |@@@@@@@@@                                1000
163              54 |@@@@@@@@@                                1000
164              55 |@@@@@@@@@                                1000
165              56 |@@@@@@@@@                                1000
166              57 |@@@@@@@@@                                1000
167              58 |@@@@@@@@@                                1000
168              59 |@@@@@@@@@                                1000
169              60 |@@@@@@@@@                                1000
170              61 |@@@@@@@@@                                1000
171              62 |@@@@@@@@@                                1000
172              63 |@@@@@@@@@                                1000
173              64 |@@@@@@@@@                                1000
174              65 |@@@@@@@@@                                1000
175              66 |@@@@@@@@@                                1000
176              67 |@@@@@@@@@                                1000
177              68 |@@@@@@@@@                                1000
178              69 |@@@@@@@@@                                1000
179              70 |@@@@@@@@@                                1000
180              71 |@@@@@@@@@                                1000
181              72 |@@@@@@@@@                                1000
182              73 |@@@@@@@@@                                1000
183              74 |@@@@@@@@@                                1000
184              75 |@@@@@@@@@                                1000
185              76 |@@@@@@@@@                                1000
186              77 |@@@@@@@@@                                1000
187              78 |@@@@@@@@@                                1000
188              79 |@@@@@@@@@                                1000
189              80 |@@@@@@@@@                                1000
190              81 |@@@@@@@@@                                1000
191              82 |@@@@@@@@@                                1000
```



```
192              83 |@@@@@@@@@                                1000
193              84 |@@@@@@@@@                                1000
194              85 |@@@@@@@@@                                1000
195              86 |@@@@@@@@@                                1000
196              87 |@@@@@@@@@                                1000
197              88 |@@@@@@@@@                                1000
198              89 |@@@@@@@@@                                1000
199              90 |@@@@@@@@@                                1000
200              91 |                                         0
201              92 |                                         0
202              93 |                                         0
203              94 |                                         0
204              95 |                                         0
205              96 |                                         0
206              97 |                                         0
207              98 |                                         0
208              99 |                                         0
209         >= 100 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   5000


212 #endif /* ! codereview */
```

```
**********************************************************
   52498 Tue Jan 14 20:13:03 2014
new/usr/src/lib/libdtrace/common/dt_aggregate.c
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2013, Joyent, Inc. All rights reserved.
 * Copyright (c) 2011, Joyent, Inc. All rights reserved.
 * Copyright (c) 2012 by Delphix. All rights reserved.
 */

#include <stdlib.h>
#include <strings.h>
#include <errno.h>
#include <unistd.h>
#include <dt_impl.h>
#include <assert.h>
#include <alloca.h>
#include <limits.h>

#define DTRACE_AHASHSIZE        32779           /* big 'ol prime */

/*
 * Because qsort(3C) does not allow an argument to be passed to a comparison
 * function, the variables that affect comparison must regrettably be global;
 * they are protected by a global static lock, dt_qsort_lock.
 */
static pthread_mutex_t dt_qsort_lock = PTHREAD_MUTEX_INITIALIZER;

static int dt_revsort;
static int dt_keysort;
static int dt_keypos;

#define DT_LESSTHAN     (dt_revsort == 0 ? -1 : 1)
#define DT_GREATERTHAN  (dt_revsort == 0 ? 1 : -1)

static void
dt_aggregate_count(int64_t *existing, int64_t *new, size_t size)
```



```
{
        int i;

        for (i = 0; i < size / sizeof (int64_t); i++)
                existing[i] = existing[i] + new[i];
}
```

*unchanged_portion_omitted_*

```
static int
dt_aggregate_total(dtrace_hdl_t *dtp, boolean_t clear)
{
        dt_ahashent_t *h;
        dtrace_aggdata_t **total;
        dtrace_aggid_t max = DTRACE_AGGVARIDNONE, id;
        dt_aggregate_t *agp = &dtp->dt_aggregate;
        dt_ahash_t *hash = &agp->dtat_hash;
        uint32_t tflags;

        tflags = DTRACE_A_TOTAL | DTRACE_A_HASNEGATIVES | DTRACE_A_HASPOSITIVES;

        /*
         * If we need to deliver per-aggregation totals, we're going to take
         * three passes over the aggregate:  one to clear everything out and
         * determine our maximum aggregation ID, one to actually total
         * everything up, and a final pass to assign the totals to the
         * individual elements.
         */
        for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
                dtrace_aggdata_t *aggdata = &h->dtahe_data;

                if ((id = dt_aggregate_aggvarid(h)) > max)
                        max = id;

                aggdata->dtada_total = 0;
                aggdata->dtada_flags &= ~tflags;
        }

        if (clear || max == DTRACE_AGGVARIDNONE)
                return (0);

        total = dt_zalloc(dtp, (max + 1) * sizeof (dtrace_aggdata_t *));

        if (total == NULL)
                return (-1);

        for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
                dtrace_aggdata_t *aggdata = &h->dtahe_data;
                dtrace_aggdesc_t *agg = aggdata->dtada_desc;
                dtrace_recdesc_t *rec;
                caddr_t data;
                int64_t val, *addr;

                rec = &agg->dtagd_rec[agg->dtagd_nrecs - 1];
                data = aggdata->dtada_data;
                addr = (int64_t *)(uintptr_t)(data + rec->dtrd_offset);

                switch (rec->dtrd_action) {
                case DTRACEAGG_STDDEV:
                        val = dt_stddev((uint64_t *)addr, 1);
                        break;

                case DTRACEAGG_SUM:
                case DTRACEAGG_COUNT:
                        val = *addr;
                        break;
```

```
1351                 case DTRACEAGG_AVG:
1352                         val = addr[0] ? (addr[1] / addr[0]) : 0;
1353                         break;

1355                 default:
1356                         continue;
1357                 }

1359                 if (total[agg->dtagd_varid] == NULL) {
1360                         total[agg->dtagd_varid] = aggdata;
1361                         aggdata->dtada_flags |= DTRACE_A_TOTAL;
1362                 } else {
1363                         aggdata = total[agg->dtagd_varid];
1364                 }

1366                 if (val > 0)
1367                         aggdata->dtada_flags |= DTRACE_A_HASPOSITIVES;

1369                 if (val < 0) {
1370                         aggdata->dtada_flags |= DTRACE_A_HASNEGATIVES;
1371                         val = -val;
1372                 }

1374                 if (dtp->dt_options[DTRACEOPT_AGGZOOM] != DTRACEOPT_UNSET) {
1375                         val = (int64_t)((long double)val *
1376                             (1 / DTRACE_AGGZOOM_MAX));

1378                         if (val > aggdata->dtada_total)
1379                                 aggdata->dtada_total = val;
1380                 } else {
1381                         aggdata->dtada_total += val;
1382                 }
1383         }

1385         /*
1386          * And now one final pass to set everyone's total.
1387          */
1388         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
1389                 dtrace_aggdata_t *aggdata = &h->dtahe_data, *t;
1390                 dtrace_aggdesc_t *agg = aggdata->dtada_desc;

1392                 if ((t = total[agg->dtagd_varid]) == NULL || aggdata == t)
1393                         continue;

1395                 aggdata->dtada_total = t->dtada_total;
1396                 aggdata->dtada_flags |= (t->dtada_flags & tflags);
1397         }

1399         dt_free(dtp, total);

1401         return (0);
1402 }

1404 static int
1405 dt_aggregate_minmaxbin(dtrace_hdl_t *dtp, boolean_t clear)
1406 {
1407         dt_ahashent_t *h;
1408         dtrace_aggdata_t **minmax;
1409         dtrace_aggid_t max = DTRACE_AGGVARIDNONE, id;
1410         dt_aggregate_t *agp = &dtp->dt_aggregate;
1411         dt_ahash_t *hash = &agp->dtat_hash;

1413         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
1414                 dtrace_aggdata_t *aggdata = &h->dtahe_data;

1416                 if ((id = dt_aggregate_aggvarid(h)) > max)
```



```
1417                         max = id;

1419                 aggdata->dtada_minbin = 0;
1420                 aggdata->dtada_maxbin = 0;
1421                 aggdata->dtada_flags &= ~DTRACE_A_MINMAXBIN;
1422         }

1424         if (clear || max == DTRACE_AGGVARIDNONE)
1425                 return (0);

1427         minmax = dt_zalloc(dtp, (max + 1) * sizeof (dtrace_aggdata_t *));

1429         if (minmax == NULL)
1430                 return (-1);

1432         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
1433                 dtrace_aggdata_t *aggdata = &h->dtahe_data;
1434                 dtrace_aggdesc_t *agg = aggdata->dtada_desc;
1435                 dtrace_recdesc_t *rec;
1436                 caddr_t data;
1437                 int64_t *addr;
1438                 int minbin = -1, maxbin = -1, i;
1439                 int start = 0, size;

1441                 rec = &agg->dtagd_rec[agg->dtagd_nrecs - 1];
1442                 size = rec->dtrd_size / sizeof (int64_t);
1443                 data = aggdata->dtada_data;
1444                 addr = (int64_t *)(uintptr_t)(data + rec->dtrd_offset);

1446                 switch (rec->dtrd_action) {
1447                 case DTRACEAGG_LQUANTIZE:
1448                         /*
1449                          * For lquantize(), we always display the entire range
1450                          * of the aggregation when aggpack is set.
1451                          */
1452                         start = 1;
1453                         minbin = start;
1454                         maxbin = size - 1 - start;
1455                         break;

1457                 case DTRACEAGG_QUANTIZE:
1458                         for (i = start; i < size; i++) {
1459                                 if (!addr[i])
1460                                         continue;

1462                                 if (minbin == -1)
1463                                         minbin = i - start;

1465                                 maxbin = i - start;
1466                         }

1468                         if (minbin == -1) {
1469                                 /*
1470                                  * If we have no data (e.g., due to a clear()
1471                                  * or negative increments), we'll use the
1472                                  * zero bucket as both our min and max.
1473                                  */
1474                                 minbin = maxbin = DTRACE_QUANTIZE_ZEROBUCKET;
1475                         }

1477                         break;

1479                 default:
1480                         continue;
1481                 }
```

```
1483                 if (minmax[agg->dtagd_varid] == NULL) {
1484                         minmax[agg->dtagd_varid] = aggdata;
1485                         aggdata->dtada_flags |= DTRACE_A_MINMAXBIN;
1486                         aggdata->dtada_minbin = minbin;
1487                         aggdata->dtada_maxbin = maxbin;
1488                         continue;
1489                 }

1491                 if (minbin < minmax[agg->dtagd_varid]->dtada_minbin)
1492                         minmax[agg->dtagd_varid]->dtada_minbin = minbin;

1494                 if (maxbin > minmax[agg->dtagd_varid]->dtada_maxbin)
1495                         minmax[agg->dtagd_varid]->dtada_maxbin = maxbin;
1496         }

1498         /*
1499          * And now one final pass to set everyone's minbin and maxbin.
1500          */
1501         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
1502                 dtrace_aggdata_t *aggdata = &h->dtahe_data, *mm;
1503                 dtrace_aggdesc_t *agg = aggdata->dtada_desc;

1505                 if ((mm = minmax[agg->dtagd_varid]) == NULL || aggdata == mm)
1506                         continue;

1508                 aggdata->dtada_minbin = mm->dtada_minbin;
1509                 aggdata->dtada_maxbin = mm->dtada_maxbin;
1510                 aggdata->dtada_flags |= DTRACE_A_MINMAXBIN;
1511         }

1513         dt_free(dtp, minmax);

1515         return (0);
1516 }

1518 static int
1519 #endif /* ! codereview */
1520 dt_aggregate_walk_sorted(dtrace_hdl_t *dtp,
1521     dtrace_aggregate_f *func, void *arg,
1522     int (*sfunc)(const void *, const void *))
1523 {
1524         dt_aggregate_t *agp = &dtp->dt_aggregate;
1525         dt_ahashent_t *h, **sorted;
1526         dt_ahash_t *hash = &agp->dtat_hash;
1527         size_t i, nentries = 0;
1528         int rval = -1;

1530         agp->dtat_flags &= ~(DTRACE_A_TOTAL | DTRACE_A_MINMAXBIN);

1532         if (dtp->dt_options[DTRACEOPT_AGGHIST] != DTRACEOPT_UNSET) {
1533                 agp->dtat_flags |= DTRACE_A_TOTAL;

1535                 if (dt_aggregate_total(dtp, B_FALSE) != 0)
1536                         return (-1);
1537         }

1539         if (dtp->dt_options[DTRACEOPT_AGGPACK] != DTRACEOPT_UNSET) {
1540                 agp->dtat_flags |= DTRACE_A_MINMAXBIN;

1542                 if (dt_aggregate_minmaxbin(dtp, B_FALSE) != 0)
1543                         return (-1);
1544         }
1545 #endif /* ! codereview */

1547         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall)
1548                 nentries++;
```



```
1550         sorted = dt_alloc(dtp, nentries * sizeof (dt_ahashent_t *));

1552         if (sorted == NULL)
1553                 goto out;
1294                 return (-1);

1555         for (h = hash->dtah_all, i = 0; h != NULL; h = h->dtahe_nextall)
1556                 sorted[i++] = h;

1558         (void) pthread_mutex_lock(&dt_qsort_lock);

1560         if (sfunc == NULL) {
1561                 dt_aggregate_qsort(dtp, sorted, nentries,
1562                     sizeof (dt_ahashent_t *), NULL);
1563         } else {
1564                 /*
1565                  * If we've been explicitly passed a sorting function,
1566                  * we'll use that -- ignoring the values of the "aggsortrev",
1567                  * "aggsortkey" and "aggsortkeypos" options.
1568                  */
1569                 qsort(sorted, nentries, sizeof (dt_ahashent_t *), sfunc);
1570         }

1572         (void) pthread_mutex_unlock(&dt_qsort_lock);

1574         for (i = 0; i < nentries; i++) {
1575                 h = sorted[i];

1577                 if (dt_aggwalk_rval(dtp, h, func(&h->dtahe_data, arg)) == -1)
1578                         goto out;
1318                 if (dt_aggwalk_rval(dtp, h, func(&h->dtahe_data, arg)) == -1) {
1319                         dt_free(dtp, sorted);
1320                         return (-1);
1321                 }
1579         }

1581         rval = 0;
1582 out:
1583         if (agp->dtat_flags & DTRACE_A_TOTAL)
1584                 (void) dt_aggregate_total(dtp, B_TRUE);

1586         if (agp->dtat_flags & DTRACE_A_MINMAXBIN)
1587                 (void) dt_aggregate_minmaxbin(dtp, B_TRUE);
1588 #endif /* ! codereview */

1590         dt_free(dtp, sorted);
1591         return (rval);
1323         return (0);
1592 }
```

*_____unchanged_portion_omitted_*

```
2108 int
2109 dtrace_aggregate_print(dtrace_hdl_t *dtp, FILE *fp,
2110     dtrace_aggregate_walk_f *func)
2111 {
2112         dt_print_aggdata_t pd;

2114         bzero(&pd, sizeof (pd));

2116 #endif /* ! codereview */
2117         pd.dtpa_dtp = dtp;
2118         pd.dtpa_fp = fp;
2119         pd.dtpa_allunprint = 1;

2121         if (func == NULL)
```

```
2122                 func = dtrace_aggregate_walk_sorted;

2124         if ((*func)(dtp, dt_print_agg, &pd) == -1)
2125                 return (dt_set_errno(dtp, dtp->dt_errno));

2127         return (0);
2128 }

2130 void
2131 dtrace_aggregate_clear(dtrace_hdl_t *dtp)
2132 {
2133         dt_aggregate_t *agp = &dtp->dt_aggregate;
2134         dt_ahash_t *hash = &agp->dtat_hash;
2135         dt_ahashent_t *h;
2136         dtrace_aggdata_t *data;
2137         dtrace_aggdesc_t *aggdesc;
2138         dtrace_recdesc_t *rec;
2139         int i, max_cpus = agp->dtat_maxcpu;

2141         for (h = hash->dtah_all; h != NULL; h = h->dtahe_nextall) {
2142                 aggdesc = h->dtahe_data.dtada_desc;
2143                 rec = &aggdesc->dtagd_rec[aggdesc->dtagd_nrecs - 1];
2144                 data = &h->dtahe_data;

2146                 bzero(&data->dtada_data[rec->dtrd_offset], rec->dtrd_size);

2148                 if (data->dtada_percpu == NULL)
2149                         continue;

2151                 for (i = 0; i < max_cpus; i++)
2152                         bzero(data->dtada_percpu[i], rec->dtrd_size);
2153         }
2154 }

2156 void
2157 dt_aggregate_destroy(dtrace_hdl_t *dtp)
2158 {
2159         dt_aggregate_t *agp = &dtp->dt_aggregate;
2160         dt_ahash_t *hash = &agp->dtat_hash;
2161         dt_ahashent_t *h, *next;
2162         dtrace_aggdata_t *aggdata;
2163         int i, max_cpus = agp->dtat_maxcpu;

2165         if (hash->dtah_hash == NULL) {
2166                 assert(hash->dtah_all == NULL);
2167         } else {
2168                 free(hash->dtah_hash);

2170                 for (h = hash->dtah_all; h != NULL; h = next) {
2171                         next = h->dtahe_nextall;

2173                         aggdata = &h->dtahe_data;

2175                         if (aggdata->dtada_percpu != NULL) {
2176                                 for (i = 0; i < max_cpus; i++)
2177                                         free(aggdata->dtada_percpu[i]);
2178                                 free(aggdata->dtada_percpu);
2179                         }

2181                         free(aggdata->dtada_data);
2182                         free(h);
2183                 }

2185                 hash->dtah_hash = NULL;
2186                 hash->dtah_all = NULL;
2187                 hash->dtah_size = 0;
```



```
2188         }

2190         free(agp->dtat_buf.dtbd_data);
2191         free(agp->dtat_cpus);
2192 }
```

```
**********************************************************
   75012 Tue Jan 14 20:13:04 2014
new/usr/src/lib/libdtrace/common/dt_consume.c
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2013, Joyent, Inc. All rights reserved.
 * Copyright (c) 2011, Joyent, Inc. All rights reserved.
 * Copyright (c) 2012 by Delphix. All rights reserved.
 */

#include <stdlib.h>
#include <strings.h>
#include <errno.h>
#include <unistd.h>
#include <limits.h>
#include <assert.h>
#include <ctype.h>
#include <alloca.h>
#include <dt_impl.h>
#include <dt_pq.h>

#define DT_MASK_LO 0x00000000FFFFFFFFULL

/*
 * We declare this here because (1) we need it and (2) we want to avoid a
 * dependency on libm in libdtrace.
 */
static long double
dt_fabsl(long double x)
{
        if (x < 0)
                return (-x);

        return (x);
}

static int
dt_ndigits(long long val)
```



```
{
        int rval = 1;
        long long cmp = 10;

        if (val < 0) {
                val = val == INT64_MIN ? INT64_MAX : -val;
                rval++;
        }

        while (val > cmp && cmp > 0) {
                rval++;
                cmp *= 10;
        }

        return (rval < 4 ? 4 : rval);
}

#endif /* ! codereview */
/*
 * 128-bit arithmetic functions needed to support the stddev() aggregating
 * action.
 */
static int
dt_gt_128(uint64_t *a, uint64_t *b)
{
        return (a[1] > b[1] || (a[1] == b[1] && a[0] > b[0]));
}

static int
dt_ge_128(uint64_t *a, uint64_t *b)
{
        return (a[1] > b[1] || (a[1] == b[1] && a[0] >= b[0]));
}

static int
dt_le_128(uint64_t *a, uint64_t *b)
{
        return (a[1] < b[1] || (a[1] == b[1] && a[0] <= b[0]));
}

/*
 * Shift the 128-bit value in a by b. If b is positive, shift left.
 * If b is negative, shift right.
 */
static void
dt_shift_128(uint64_t *a, int b)
{
        uint64_t mask;

        if (b == 0)
                return;

        if (b < 0) {
                b = -b;
                if (b >= 64) {
                        a[0] = a[1] >> (b - 64);
                        a[1] = 0;
                } else {
                        a[0] >>= b;
                        mask = 1LL << (64 - b);
                        mask -= 1;
                        a[0] |= ((a[1] & mask) << (64 - b));
                        a[1] >>= b;
                }
        } else {
                if (b >= 64) {
```

```
125                         a[1] = a[0] << (b - 64);
126                         a[0] = 0;
127                 } else {
128                         a[1] <<= b;
129                         mask = a[0] >> (64 - b);
130                         a[1] |= mask;
131                         a[0] <<= b;
132                 }
133         }
134 }

136 static int
137 dt_nbits_128(uint64_t *a)
138 {
139         int nbits = 0;
140         uint64_t tmp[2];
141         uint64_t zero[2] = { 0, 0 };

143         tmp[0] = a[0];
144         tmp[1] = a[1];

146         dt_shift_128(tmp, -1);
147         while (dt_gt_128(tmp, zero)) {
148                 dt_shift_128(tmp, -1);
149                 nbits++;
150         }

152         return (nbits);
153 }

155 static void
156 dt_subtract_128(uint64_t *minuend, uint64_t *subtrahend, uint64_t *difference)
157 {
158         uint64_t result[2];

160         result[0] = minuend[0] - subtrahend[0];
161         result[1] = minuend[1] - subtrahend[1] -
162             (minuend[0] < subtrahend[0] ? 1 : 0);

164         difference[0] = result[0];
165         difference[1] = result[1];
166 }

168 static void
169 dt_add_128(uint64_t *addend1, uint64_t *addend2, uint64_t *sum)
170 {
171         uint64_t result[2];

173         result[0] = addend1[0] + addend2[0];
174         result[1] = addend1[1] + addend2[1] +
175             (result[0] < addend1[0] || result[0] < addend2[0] ? 1 : 0);

177         sum[0] = result[0];
178         sum[1] = result[1];
179 }

181 /*
182  * The basic idea is to break the 2 64-bit values into 4 32-bit values,
183  * use native multiplication on those, and then re-combine into the
184  * resulting 128-bit value.
185  *
186  * (hi1 << 32 + lo1) * (hi2 << 32 + lo2) =
187  *     hi1 * hi2 << 64 +
188  *     hi1 * lo2 << 32 +
189  *     hi2 * lo1 << 32 +
190  *     lo1 * lo2
```



```
191  */
192 static void
193 dt_multiply_128(uint64_t factor1, uint64_t factor2, uint64_t *product)
194 {
195         uint64_t hi1, hi2, lo1, lo2;
196         uint64_t tmp[2];

198         hi1 = factor1 >> 32;
199         hi2 = factor2 >> 32;

201         lo1 = factor1 & DT_MASK_LO;
202         lo2 = factor2 & DT_MASK_LO;

204         product[0] = lo1 * lo2;
205         product[1] = hi1 * hi2;

207         tmp[0] = hi1 * lo2;
208         tmp[1] = 0;
209         dt_shift_128(tmp, 32);
210         dt_add_128(product, tmp, product);

212         tmp[0] = hi2 * lo1;
213         tmp[1] = 0;
214         dt_shift_128(tmp, 32);
215         dt_add_128(product, tmp, product);
216 }

218 /*
219  * This is long-hand division.
220  *
221  * We initialize subtrahend by shifting divisor left as far as possible. We
222  * loop, comparing subtrahend to dividend:  if subtrahend is smaller, we
223  * subtract and set the appropriate bit in the result.  We then shift
224  * subtrahend right by one bit for the next comparison.
225  */
226 static void
227 dt_divide_128(uint64_t *dividend, uint64_t divisor, uint64_t *quotient)
228 {
229         uint64_t result[2] = { 0, 0 };
230         uint64_t remainder[2];
231         uint64_t subtrahend[2];
232         uint64_t divisor_128[2];
233         uint64_t mask[2] = { 1, 0 };
234         int log = 0;

236         assert(divisor != 0);

238         divisor_128[0] = divisor;
239         divisor_128[1] = 0;

241         remainder[0] = dividend[0];
242         remainder[1] = dividend[1];

244         subtrahend[0] = divisor;
245         subtrahend[1] = 0;

247         while (divisor > 0) {
248                 log++;
249                 divisor >>= 1;
250         }

252         dt_shift_128(subtrahend, 128 - log);
253         dt_shift_128(mask, 128 - log);

255         while (dt_ge_128(remainder, divisor_128)) {
256                 if (dt_ge_128(remainder, subtrahend)) {
```

```
257                         dt_subtract_128(remainder, subtrahend, remainder);
258                         result[0] |= mask[0];
259                         result[1] |= mask[1];
260                 }

262                 dt_shift_128(subtrahend, -1);
263                 dt_shift_128(mask, -1);
264         }

266         quotient[0] = result[0];
267         quotient[1] = result[1];
268 }

270 /*
271  * This is the long-hand method of calculating a square root.
272  * The algorithm is as follows:
273  *
274  * 1. Group the digits by 2 from the right.
275  * 2. Over the leftmost group, find the largest single-digit number
276  *    whose square is less than that group.
277  * 3. Subtract the result of the previous step (2 or 4, depending) and
278  *    bring down the next two-digit group.
279  * 4. For the result R we have so far, find the largest single-digit number
280  *    x such that 2 * R * 10 * x + x^2 is less than the result from step 3.
281  *    (Note that this is doubling R and performing a decimal left-shift by 1
282  *    and searching for the appropriate decimal to fill the one's place.)
283  *    The value x is the next digit in the square root.
284  * Repeat steps 3 and 4 until the desired precision is reached.  (We're
285  * dealing with integers, so the above is sufficient.)
286  *
287  * In decimal, the square root of 582,734 would be calculated as so:
288  *
289  *      __7__6__3
290  *    | 58 27 34
291  *     -49       (7^2 == 49 => 7 is the first digit in the square root)
292  *      --
293  *       9 27    (Subtract and bring down the next group.)
294  * 146   8 76    (2 * 7 * 10 * 6 + 6^2 == 876 => 6 is the next digit in
295  *      -----     the square root)
296  *         51 34 (Subtract and bring down the next group.)
297  * 1523    45 69 (2 * 76 * 10 * 3 + 3^2 == 4569 => 3 is the next digit in
298  *         -----  the square root)
299  *          5 65 (remainder)
300  *
301  * The above algorithm applies similarly in binary, but note that the
302  * only possible non-zero value for x in step 4 is 1, so step 4 becomes a
303  * simple decision: is 2 * R * 2 * 1 + 1^2 (aka R << 2 + 1) less than the
304  * preceding difference?
305  *
306  * In binary, the square root of 11011011 would be calculated as so:
307  *
308  *      __1__1__1__0
309  *    | 11 01 10 11
310  *      01          (0 << 2 + 1 == 1 < 11 => this bit is 1)
311  *      --
312  *      10 01 10 11
313  * 101   1 01       (1 << 2 + 1 == 101 < 1001 => next bit is 1)
314  *      -----
315  *       1 00 10 11
316  * 1101    11 01    (11 << 2 + 1 == 1101 < 10010 => next bit is 1)
317  *       -------
318  *          1 01 11
319  * 11101    1 11 01 (111 << 2 + 1 == 11101 > 10111 => last bit is 0)
320  *
321  */
322 static uint64_t
```



```
323 dt_sqrt_128(uint64_t *square)
324 {
325         uint64_t result[2] = { 0, 0 };
326         uint64_t diff[2] = { 0, 0 };
327         uint64_t one[2] = { 1, 0 };
328         uint64_t next_pair[2];
329         uint64_t next_try[2];
330         uint64_t bit_pairs, pair_shift;
331         int i;

333         bit_pairs = dt_nbits_128(square) / 2;
334         pair_shift = bit_pairs * 2;

336         for (i = 0; i <= bit_pairs; i++) {
337                 /*
338                  * Bring down the next pair of bits.
339                  */
340                 next_pair[0] = square[0];
341                 next_pair[1] = square[1];
342                 dt_shift_128(next_pair, -pair_shift);
343                 next_pair[0] &= 0x3;
344                 next_pair[1] = 0;

346                 dt_shift_128(diff, 2);
347                 dt_add_128(diff, next_pair, diff);

349                 /*
350                  * next_try = R << 2 + 1
351                  */
352                 next_try[0] = result[0];
353                 next_try[1] = result[1];
354                 dt_shift_128(next_try, 2);
355                 dt_add_128(next_try, one, next_try);

357                 if (dt_le_128(next_try, diff)) {
358                         dt_subtract_128(diff, next_try, diff);
359                         dt_shift_128(result, 1);
360                         dt_add_128(result, one, result);
361                 } else {
362                         dt_shift_128(result, 1);
363                 }

365                 pair_shift -= 2;
366         }

368         assert(result[1] == 0);

370         return (result[0]);
371 }

373 uint64_t
374 dt_stddev(uint64_t *data, uint64_t normal)
375 {
376         uint64_t avg_of_squares[2];
377         uint64_t square_of_avg[2];
378         int64_t norm_avg;
379         uint64_t diff[2];

381         /*
382          * The standard approximation for standard deviation is
383          * sqrt(average(x**2) - average(x)**2), i.e. the square root
384          * of the average of the squares minus the square of the average.
385          */
386         dt_divide_128(data + 2, normal, avg_of_squares);
387         dt_divide_128(avg_of_squares, data[0], avg_of_squares);
```

```
389         norm_avg = (int64_t)data[1] / (int64_t)normal / (int64_t)data[0];
391         if (norm_avg < 0)
392                 norm_avg = -norm_avg;
394         dt_multiply_128((uint64_t)norm_avg, (uint64_t)norm_avg, square_of_avg);
396         dt_subtract_128(avg_of_squares, square_of_avg, diff);
398         return (dt_sqrt_128(diff));
399 }
401 static int
402 dt_flowindent(dtrace_hdl_t *dtp, dtrace_probedata_t *data, dtrace_epid_t last,
403     dtrace_bufdesc_t *buf, size_t offs)
404 {
405         dtrace_probedesc_t *pd = data->dtpda_pdesc, *npd;
406         dtrace_eprobedesc_t *epd = data->dtpda_edesc, *nepd;
407         char *p = pd->dtpd_provider, *n = pd->dtpd_name, *sub;
408         dtrace_flowkind_t flow = DTRACEFLOW_NONE;
409         const char *str = NULL;
410         static const char *e_str[2] = { " -> ", " => " };
411         static const char *r_str[2] = { " <- ", " <= " };
412         static const char *ent = "entry", *ret = "return";
413         static int entlen = 0, retlen = 0;
414         dtrace_epid_t next, id = epd->dtepd_epid;
415         int rval;
417         if (entlen == 0) {
418                 assert(retlen == 0);
419                 entlen = strlen(ent);
420                 retlen = strlen(ret);
421         }
423         /*
424          * If the name of the probe is "entry" or ends with "-entry", we
425          * treat it as an entry; if it is "return" or ends with "-return",
426          * we treat it as a return.  (This allows application-provided probes
427          * like "method-entry" or "function-entry" to participate in flow
428          * indentation -- without accidentally misinterpreting popular probe
429          * names like "carpentry", "gentry" or "Coventry".)
430          */
431         if ((sub = strstr(n, ent)) != NULL && sub[entlen] == '\0' &&
432             (sub == n || sub[-1] == '-')) {
433                 flow = DTRACEFLOW_ENTRY;
434                 str = e_str[strcmp(p, "syscall") == 0];
435         } else if ((sub = strstr(n, ret)) != NULL && sub[retlen] == '\0' &&
436             (sub == n || sub[-1] == '-')) {
437                 flow = DTRACEFLOW_RETURN;
438                 str = r_str[strcmp(p, "syscall") == 0];
439         }
441         /*
442          * If we're going to indent this, we need to check the ID of our last
443          * call.  If we're looking at the same probe ID but a different EPID,
444          * we _don't_ want to indent.  (Yes, there are some minor holes in
445          * this scheme -- it's a heuristic.)
446          */
447         if (flow == DTRACEFLOW_ENTRY) {
448                 if ((last != DTRACE_EPIDNONE && id != last &&
449                     pd->dtpd_id == dtp->dt_pdesc[last]->dtpd_id))
450                         flow = DTRACEFLOW_NONE;
451         }
453         /*
454          * If we're going to unindent this, it's more difficult to see if
```



```
455          * we don't actually want to unindent it -- we need to look at the
456          * _next_ EPID.
457          */
458         if (flow == DTRACEFLOW_RETURN) {
459                 offs += epd->dtepd_size;
461                 do {
462                         if (offs >= buf->dtbd_size)
463                                 goto out;
465                         next = *(uint32_t *)((uintptr_t)buf->dtbd_data + offs);
467                         if (next == DTRACE_EPIDNONE)
468                                 offs += sizeof (id);
469                 } while (next == DTRACE_EPIDNONE);
471                 if ((rval = dt_epid_lookup(dtp, next, &nepd, &npd)) != 0)
472                         return (rval);
474                 if (next != id && npd->dtpd_id == pd->dtpd_id)
475                         flow = DTRACEFLOW_NONE;
476         }
478 out:
479         if (flow == DTRACEFLOW_ENTRY || flow == DTRACEFLOW_RETURN) {
480                 data->dtpda_prefix = str;
481         } else {
482                 data->dtpda_prefix = "| ";
483         }
485         if (flow == DTRACEFLOW_RETURN && data->dtpda_indent > 0)
486                 data->dtpda_indent -= 2;
488         data->dtpda_flow = flow;
490         return (0);
491 }
493 static int
494 dt_nullprobe()
495 {
496         return (DTRACE_CONSUME_THIS);
497 }
499 static int
500 dt_nullrec()
501 {
502         return (DTRACE_CONSUME_NEXT);
503 }
505 static void
506 dt_quantize_total(dtrace_hdl_t *dtp, int64_t datum, long double *total)
507 {
508         long double val = dt_fabsl((long double)datum);
510         if (dtp->dt_options[DTRACEOPT_AGGZOOM] == DTRACEOPT_UNSET) {
511                 *total += val;
512                 return;
513         }
515         /*
516          * If we're zooming in on an aggregation, we want the height of the
517          * highest value to be approximately 95% of total bar height -- so we
518          * adjust up by the reciprocal of DTRACE_AGGZOOM_MAX when comparing to
519          * our highest value.
520          */
```

```
521         val *= 1 / DTRACE_AGGZOOM_MAX;
523         if (*total < val)
524                 *total = val;
525 }
527 static int
528 dt_print_quanthdr(dtrace_hdl_t *dtp, FILE *fp, int width)
529 {
530         return (dt_printf(dtp, fp, "\n%*s %41s %-9s\n",
531             width ? width : 16, width ? "key" : "value",
532             "------------- Distribution -------------", "count"));
533 }
535 static int
536 dt_print_quanthdr_packed(dtrace_hdl_t *dtp, FILE *fp, int width,
537     const dtrace_aggdata_t *aggdata, dtrace_actkind_t action)
538 {
539         int min = aggdata->dtada_minbin, max = aggdata->dtada_maxbin;
540         int minwidth, maxwidth, i;
542         assert(action == DTRACEAGG_QUANTIZE || action == DTRACEAGG_LQUANTIZE);
544         if (action == DTRACEAGG_QUANTIZE) {
545                 if (min != 0 && min != DTRACE_QUANTIZE_ZEROBUCKET)
546                         min--;
548                 if (max < DTRACE_QUANTIZE_NBUCKETS - 1)
549                         max++;
551                 minwidth = dt_ndigits(DTRACE_QUANTIZE_BUCKETVAL(min));
552                 maxwidth = dt_ndigits(DTRACE_QUANTIZE_BUCKETVAL(max));
553         } else {
554                 maxwidth = 8;
555                 minwidth = maxwidth - 1;
556                 max++;
557         }
559         if (dt_printf(dtp, fp, "\n%*s %*s .",
560             width, width > 0 ? "key" : "", minwidth, "min") < 0)
561                 return (-1);
563         for (i = min; i <= max; i++) {
564                 if (dt_printf(dtp, fp, "-") < 0)
565                         return (-1);
566         }
568         return (dt_printf(dtp, fp, ". %*s | count\n", -maxwidth, "max"));
569 }
571 /*
572  * We use a subset of the Unicode Block Elements (U+2588 through U+258F,
573  * inclusive) to represent aggregations via UTF-8 -- which are expressed via
574  * 3-byte UTF-8 sequences.
575  */
576 #define DTRACE_AGGUTF8_FULL     0x2588
577 #define DTRACE_AGGUTF8_BASE     0x258f
578 #define DTRACE_AGGUTF8_LEVELS   8
580 #define DTRACE_AGGUTF8_BYTE0(val)       (0xe0 | ((val) >> 12))
581 #define DTRACE_AGGUTF8_BYTE1(val)       (0x80 | (((val) >> 6) & 0x3f))
582 #define DTRACE_AGGUTF8_BYTE2(val)       (0x80 | ((val) & 0x3f))
584 static int
585 dt_print_quantline_utf8(dtrace_hdl_t *dtp, FILE *fp, int64_t val,
586     uint64_t normal, long double total)
```



```
587 {
588         uint_t len = 40, i, whole, partial;
589         long double f = (dt_fabsl((long double)val) * len) / total;
590         const char *spaces = "                                        ";
592         whole = (uint_t)f;
593         partial = (uint_t)((f - (long double)(uint_t)f) *
594             (long double)DTRACE_AGGUTF8_LEVELS);
596         if (dt_printf(dtp, fp, "|") < 0)
597                 return (-1);
599         for (i = 0; i < whole; i++) {
600                 if (dt_printf(dtp, fp, "%c%c%c",
601                     DTRACE_AGGUTF8_BYTE0(DTRACE_AGGUTF8_FULL),
602                     DTRACE_AGGUTF8_BYTE1(DTRACE_AGGUTF8_FULL),
603                     DTRACE_AGGUTF8_BYTE2(DTRACE_AGGUTF8_FULL)) < 0)
604                         return (-1);
605         }
607         if (partial != 0) {
608                 partial = DTRACE_AGGUTF8_BASE - (partial - 1);
610                 if (dt_printf(dtp, fp, "%c%c%c",
611                     DTRACE_AGGUTF8_BYTE0(partial),
612                     DTRACE_AGGUTF8_BYTE1(partial),
613                     DTRACE_AGGUTF8_BYTE2(partial)) < 0)
614                         return (-1);
616                 i++;
617         }
619         return (dt_printf(dtp, fp, "%s %-9lld\n", spaces + i,
620             (long long)val / normal));
621 }
623 static int
 57 int
624 dt_print_quantline(dtrace_hdl_t *dtp, FILE *fp, int64_t val,
625     uint64_t normal, long double total, char positives, char negatives)
626 {
627         long double f;
628         uint_t depth, len = 40;
630         const char *ats = "@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@";
631         const char *spaces = "                                        ";
633         assert(strlen(ats) == len && strlen(spaces) == len);
634         assert(!(total == 0 && (positives || negatives)));
635         assert(!(val < 0 && !negatives));
636         assert(!(val > 0 && !positives));
637         assert(!(val != 0 && total == 0));
639         if (!negatives) {
640                 if (positives) {
641                         if (dtp->dt_encoding == DT_ENCODING_UTF8) {
642                                 return (dt_print_quantline_utf8(dtp, fp, val,
643                                     normal, total));
644                         }
646 #endif /* ! codereview */
647                         f = (dt_fabsl((long double)val) * len) / total;
648                         depth = (uint_t)(f + 0.5);
649                 } else {
650                         depth = 0;
651                 }
```

```
653                 return (dt_printf(dtp, fp, "|%s%s %-9lld\n", ats + len - depth,
654                     spaces + depth, (long long)val / normal));
655         }
656
657         if (!positives) {
658                 f = (dt_fabsl((long double)val) * len) / total;
659                 depth = (uint_t)(f + 0.5);
660
661                 return (dt_printf(dtp, fp, "%s%s| %-9lld\n", spaces + depth,
662                     ats + len - depth, (long long)val / normal));
663         }
664
665         /*
666          * If we're here, we have both positive and negative bucket values.
667          * To express this graphically, we're going to generate both positive
668          * and negative bars separated by a centerline.  These bars are half
669          * the size of normal quantize()/lquantize() bars, so we divide the
670          * length in half before calculating the bar length.
671          */
672         len /= 2;
673         ats = &ats[len];
674         spaces = &spaces[len];
675
676         f = (dt_fabsl((long double)val) * len) / total;
677         depth = (uint_t)(f + 0.5);
678
679         if (val <= 0) {
680                 return (dt_printf(dtp, fp, "%s%s|%*s %-9lld\n", spaces + depth,
681                     ats + len - depth, len, "", (long long)val / normal));
682         } else {
683                 return (dt_printf(dtp, fp, "%20s|%s%s %-9lld\n", "",
684                     ats + len - depth, spaces + depth,
685                     (long long)val / normal));
686         }
687 }
688
689 /*
690  * As with UTF-8 printing of aggregations, we use a subset of the Unicode
691  * Block Elements (U+2581 through U+2588, inclusive) to represent our packed
692  * aggregation.
693  */
694 #define DTRACE_AGGPACK_BASE     0x2581
695 #define DTRACE_AGGPACK_LEVELS   8
696
697 static int
698 dt_print_packed(dtrace_hdl_t *dtp, FILE *fp,
699     long double datum, long double total)
700 {
701         static boolean_t utf8_checked = B_FALSE;
702         static boolean_t utf8;
703         char *ascii = "__xxxxXX";
704         char *neg = "vvvvVV";
705         unsigned int len;
706         long double val;
707
708         if (!utf8_checked) {
709                 char *term;
710
711                 /*
712                  * We want to determine if we can reasonably emit UTF-8 for our
713                  * packed aggregation.  To do this, we will check for terminals
714                  * that are known to be primitive to emit UTF-8 on these.
715                  */
716                 utf8_checked = B_TRUE;
```



```
718                 if (dtp->dt_encoding == DT_ENCODING_ASCII)
719                         utf8 = B_FALSE;
720                 else if (dtp->dt_encoding == DT_ENCODING_UTF8)
721                         utf8 = B_TRUE;
722                 else if ((term = getenv("TERM")) != NULL &&
723                     (strcmp(term, "sun") == 0 ||
724                     strcmp(term, "sun-color") == 0) ||
725                     strcmp(term, "dumb") == 0)
726                         utf8 = B_FALSE;
727                 else
728                         utf8 = B_TRUE;
729         }
730
731         if (datum == 0)
732                 return (dt_printf(dtp, fp, " "));
733
734         if (datum < 0) {
735                 len = strlen(neg);
736                 val = dt_fabsl(datum * (len - 1)) / total;
737                 return (dt_printf(dtp, fp, "%c", neg[(uint_t)(val + 0.5)]));
738         }
739
740         if (utf8) {
741                 int block = DTRACE_AGGPACK_BASE + (unsigned int)(((datum *
742                     (DTRACE_AGGPACK_LEVELS - 1)) / total) + 0.5);
743
744                 return (dt_printf(dtp, fp, "%c%c%c",
745                     DTRACE_AGGUTF8_BYTE0(block),
746                     DTRACE_AGGUTF8_BYTE1(block),
747                     DTRACE_AGGUTF8_BYTE2(block)));
748         }
749
750         len = strlen(ascii);
751         val = (datum * (len - 1)) / total;
752         return (dt_printf(dtp, fp, "%c", ascii[(uint_t)(val + 0.5)]));
753 }
754
755 #endif /* ! codereview */
756 int
757 dt_print_quantize(dtrace_hdl_t *dtp, FILE *fp, const void *addr,
758     size_t size, uint64_t normal)
759 {
760         const int64_t *data = addr;
761         int i, first_bin = 0, last_bin = DTRACE_QUANTIZE_NBUCKETS - 1;
762         long double total = 0;
763         char positives = 0, negatives = 0;
764
765         if (size != DTRACE_QUANTIZE_NBUCKETS * sizeof (uint64_t))
766                 return (dt_set_errno(dtp, EDT_DMISMATCH));
767
768         while (first_bin < DTRACE_QUANTIZE_NBUCKETS - 1 && data[first_bin] == 0)
769                 first_bin++;
770
771         if (first_bin == DTRACE_QUANTIZE_NBUCKETS - 1) {
772                 /*
773                  * There isn't any data.  This is possible if the aggregation
774                  * has been clear()'d or if negative increment values have been
775                  * used.  Regardless, we'll print the buckets around 0.
 75                  * There isn't any data.  This is possible if (and only if)
 76                  * negative increment values have been used.  In this case,
 77                  * we'll print the buckets around 0.
776                  */
777                 first_bin = DTRACE_QUANTIZE_ZEROBUCKET - 1;
778                 last_bin = DTRACE_QUANTIZE_ZEROBUCKET + 1;
779         } else {
780                 if (first_bin > 0)
```

```
781                         first_bin--;
783                 while (last_bin > 0 && data[last_bin] == 0)
784                         last_bin--;
786                 if (last_bin < DTRACE_QUANTIZE_NBUCKETS - 1)
787                         last_bin++;
788         }
790         for (i = first_bin; i <= last_bin; i++) {
791                 positives |= (data[i] > 0);
792                 negatives |= (data[i] < 0);
793                 dt_quantize_total(dtp, data[i], &total);
 95                 total += dt_fabsl((long double)data[i]);
794         }
796         if (dt_print_quanthdr(dtp, fp, 0) < 0)
 98         if (dt_printf(dtp, fp, "\n%16s %41s %-9s\n", "value",
 99             "------------- Distribution -------------", "count") < 0)
797                 return (-1);
799         for (i = first_bin; i <= last_bin; i++) {
800                 if (dt_printf(dtp, fp, "%16lld ",
801                     (long long)DTRACE_QUANTIZE_BUCKETVAL(i)) < 0)
802                         return (-1);
804                 if (dt_print_quantline(dtp, fp, data[i], normal, total,
805                     positives, negatives) < 0)
806                         return (-1);
807         }
809         return (0);
810 }
812 int
813 dt_print_quantize_packed(dtrace_hdl_t *dtp, FILE *fp, const void *addr,
814     size_t size, const dtrace_aggdata_t *aggdata)
815 {
816         const int64_t *data = addr;
817         long double total = 0, count = 0;
818         int min = aggdata->dtada_minbin, max = aggdata->dtada_maxbin, i;
819         int64_t minval, maxval;
821         if (size != DTRACE_QUANTIZE_NBUCKETS * sizeof (uint64_t))
822                 return (dt_set_errno(dtp, EDT_DMISMATCH));
824         if (min != 0 && min != DTRACE_QUANTIZE_ZEROBUCKET)
825                 min--;
827         if (max < DTRACE_QUANTIZE_NBUCKETS - 1)
828                 max++;
830         minval = DTRACE_QUANTIZE_BUCKETVAL(min);
831         maxval = DTRACE_QUANTIZE_BUCKETVAL(max);
833         if (dt_printf(dtp, fp, " %*lld :", dt_ndigits(minval),
834             (long long)minval) < 0)
835                 return (-1);
837         for (i = min; i <= max; i++) {
838                 dt_quantize_total(dtp, data[i], &total);
839                 count += data[i];
840         }
842         for (i = min; i <= max; i++) {
843                 if (dt_print_packed(dtp, fp, data[i], total) < 0)
```



```
844                         return (-1);
845         }
847         if (dt_printf(dtp, fp, ": %*lld | %lld\n",
848             -dt_ndigits(maxval), (long long)maxval, (long long)count) < 0)
849                 return (-1);
851         return (0);
852 }
854 int
855 #endif /* ! codereview */
856 dt_print_lquantize(dtrace_hdl_t *dtp, FILE *fp, const void *addr,
857     size_t size, uint64_t normal)
858 {
859         const int64_t *data = addr;
860         int i, first_bin, last_bin, base;
861         uint64_t arg;
862         long double total = 0;
863         uint16_t step, levels;
864         char positives = 0, negatives = 0;
866         if (size < sizeof (uint64_t))
867                 return (dt_set_errno(dtp, EDT_DMISMATCH));
869         arg = *data++;
870         size -= sizeof (uint64_t);
872         base = DTRACE_LQUANTIZE_BASE(arg);
873         step = DTRACE_LQUANTIZE_STEP(arg);
874         levels = DTRACE_LQUANTIZE_LEVELS(arg);
876         first_bin = 0;
877         last_bin = levels + 1;
879         if (size != sizeof (uint64_t) * (levels + 2))
880                 return (dt_set_errno(dtp, EDT_DMISMATCH));
882         while (first_bin <= levels + 1 && data[first_bin] == 0)
883                 first_bin++;
885         if (first_bin > levels + 1) {
886                 first_bin = 0;
887                 last_bin = 2;
888         } else {
889                 if (first_bin > 0)
890                         first_bin--;
892                 while (last_bin > 0 && data[last_bin] == 0)
893                         last_bin--;
895                 if (last_bin < levels + 1)
896                         last_bin++;
897         }
899         for (i = first_bin; i <= last_bin; i++) {
900                 positives |= (data[i] > 0);
901                 negatives |= (data[i] < 0);
902                 dt_quantize_total(dtp, data[i], &total);
116                 total += dt_fabsl((long double)data[i]);
903         }
905         if (dt_printf(dtp, fp, "\n%16s %41s %-9s\n", "value",
906             "------------- Distribution -------------", "count") < 0)
907                 return (-1);
```

```
909         for (i = first_bin; i <= last_bin; i++) {
910                 char c[32];
911                 int err;

913                 if (i == 0) {
914                         (void) snprintf(c, sizeof (c), "< %d", base);
128                         (void) snprintf(c, sizeof (c), "< %d",
129                             base / (uint32_t)normal);
915                         err = dt_printf(dtp, fp, "%16s ", c);
916                 } else if (i == levels + 1) {
917                         (void) snprintf(c, sizeof (c), ">= %d",
918                             base + (levels * step));
919                         err = dt_printf(dtp, fp, "%16s ", c);
920                 } else {
921                         err = dt_printf(dtp, fp, "%16d ",
922                             base + (i - 1) * step);
923                 }

925                 if (err < 0 || dt_print_quantline(dtp, fp, data[i], normal,
926                     total, positives, negatives) < 0)
927                         return (-1);
928         }

930         return (0);
931 }

933 /*ARGSUSED*/
934 int
935 dt_print_lquantize_packed(dtrace_hdl_t *dtp, FILE *fp, const void *addr,
936     size_t size, const dtrace_aggdata_t *aggdata)
937 {
938         const int64_t *data = addr;
939         long double total = 0, count = 0;
940         int min, max, base, err;
941         uint64_t arg;
942         uint16_t step, levels;
943         char c[32];
944         unsigned int i;

946         if (size < sizeof (uint64_t))
947                 return (dt_set_errno(dtp, EDT_DMISMATCH));

949         arg = *data++;
950         size -= sizeof (uint64_t);

952         base = DTRACE_LQUANTIZE_BASE(arg);
953         step = DTRACE_LQUANTIZE_STEP(arg);
954         levels = DTRACE_LQUANTIZE_LEVELS(arg);

956         if (size != sizeof (uint64_t) * (levels + 2))
957                 return (dt_set_errno(dtp, EDT_DMISMATCH));

959         min = 0;
960         max = levels + 1;

962         if (min == 0) {
963                 (void) snprintf(c, sizeof (c), "< %d", base);
964                 err = dt_printf(dtp, fp, "%8s :", c);
965         } else {
966                 err = dt_printf(dtp, fp, "%8d :", base + (min - 1) * step);
967         }

969         if (err < 0)
970                 return (-1);

972         for (i = min; i <= max; i++) {
```



```
973                 dt_quantize_total(dtp, data[i], &total);
974                 count += data[i];
975         }

977         for (i = min; i <= max; i++) {
978                 if (dt_print_packed(dtp, fp, data[i], total) < 0)
979                         return (-1);
980         }

982         (void) snprintf(c, sizeof (c), ">= %d", base + (levels * step));
983         return (dt_printf(dtp, fp, ": %-8s | %lld\n", c, (long long)count));
984 }

986 #endif /* ! codereview */
987 int
988 dt_print_llquantize(dtrace_hdl_t *dtp, FILE *fp, const void *addr,
989     size_t size, uint64_t normal)
990 {
991         int i, first_bin, last_bin, bin = 1, order, levels;
992         uint16_t factor, low, high, nsteps;
993         const int64_t *data = addr;
994         int64_t value = 1, next, step;
995         char positives = 0, negatives = 0;
996         long double total = 0;
997         uint64_t arg;
998         char c[32];

1000         if (size < sizeof (uint64_t))
1001                 return (dt_set_errno(dtp, EDT_DMISMATCH));

1003         arg = *data++;
1004         size -= sizeof (uint64_t);

1006         factor = DTRACE_LLQUANTIZE_FACTOR(arg);
1007         low = DTRACE_LLQUANTIZE_LOW(arg);
1008         high = DTRACE_LLQUANTIZE_HIGH(arg);
1009         nsteps = DTRACE_LLQUANTIZE_NSTEP(arg);

1011         /*
1012          * We don't expect to be handed invalid llquantize() parameters here,
1013          * but sanity check them (to a degree) nonetheless.
1014          */
1015         if (size > INT32_MAX || factor < 2 || low >= high ||
1016             nsteps == 0 || factor > nsteps)
1017                 return (dt_set_errno(dtp, EDT_DMISMATCH));

1019         levels = (int)size / sizeof (uint64_t);

1021         first_bin = 0;
1022         last_bin = levels - 1;

1024         while (first_bin < levels && data[first_bin] == 0)
1025                 first_bin++;

1027         if (first_bin == levels) {
1028                 first_bin = 0;
1029                 last_bin = 1;
1030         } else {
1031                 if (first_bin > 0)
1032                         first_bin--;

1034                 while (last_bin > 0 && data[last_bin] == 0)
1035                         last_bin--;

1037                 if (last_bin < levels - 1)
1038                         last_bin++;
```

```
1039         }
1041         for (i = first_bin; i <= last_bin; i++) {
1042                 positives |= (data[i] > 0);
1043                 negatives |= (data[i] < 0);
1044                 dt_quantize_total(dtp, data[i], &total);
 148                 total += dt_fabsl((long double)data[i]);
1045         }
1047         if (dt_printf(dtp, fp, "\n%16s %41s %-9s\n", "value",
1048             "------------- Distribution -------------", "count") < 0)
1049                 return (-1);
1051         for (order = 0; order < low; order++)
1052                 value *= factor;
1054         next = value * factor;
1055         step = next > nsteps ? next / nsteps : 1;
1057         if (first_bin == 0) {
1058                 (void) snprintf(c, sizeof (c), "< %lld", value);
1060                 if (dt_printf(dtp, fp, "%16s ", c) < 0)
1061                         return (-1);
1063                 if (dt_print_quantline(dtp, fp, data[0], normal,
1064                     total, positives, negatives) < 0)
1065                         return (-1);
1066         }
1068         while (order <= high) {
1069                 if (bin >= first_bin && bin <= last_bin) {
1070                         if (dt_printf(dtp, fp, "%16lld ", (long long)value) < 0)
1071                                 return (-1);
1073                         if (dt_print_quantline(dtp, fp, data[bin],
1074                             normal, total, positives, negatives) < 0)
1075                                 return (-1);
1076                 }
1078                 assert(value < next);
1079                 bin++;
1081                 if ((value += step) != next)
1082                         continue;
1084                 next = value * factor;
1085                 step = next > nsteps ? next / nsteps : 1;
1086                 order++;
1087         }
1089         if (last_bin < bin)
1090                 return (0);
1092         assert(last_bin == bin);
1093         (void) snprintf(c, sizeof (c), ">= %lld", value);
1095         if (dt_printf(dtp, fp, "%16s ", c) < 0)
1096                 return (-1);
1098         return (dt_print_quantline(dtp, fp, data[bin], normal,
1099             total, positives, negatives));
1100 }
```

unchanged_portion_omitted_

```
1126 /*ARGSUSED*/
```



```
1127 static int
 231 int
1128 dt_print_bytes(dtrace_hdl_t *dtp, FILE *fp, caddr_t addr,
1129     size_t nbytes, int width, int quiet, int forceraw)
1130 {
1131         /*
1132          * If the byte stream is a series of printable characters, followed by
1133          * a terminating byte, we print it out as a string.  Otherwise, we
1134          * assume that it's something else and just print the bytes.
1135          */
1136         int i, j, margin = 5;
1137         char *c = (char *)addr;
1139         if (nbytes == 0)
1140                 return (0);
1142         if (forceraw)
1143                 goto raw;
1145         if (dtp->dt_options[DTRACEOPT_RAWBYTES] != DTRACEOPT_UNSET)
1146                 goto raw;
1148         for (i = 0; i < nbytes; i++) {
1149                 /*
1150                  * We define a "printable character" to be one for which
1151                  * isprint(3C) returns non-zero, isspace(3C) returns non-zero,
1152                  * or a character which is either backspace or the bell.
1153                  * Backspace and the bell are regrettably special because
1154                  * they fail the first two tests -- and yet they are entirely
1155                  * printable.  These are the only two control characters that
1156                  * have meaning for the terminal and for which isprint(3C) and
1157                  * isspace(3C) return 0.
1158                  */
1159                 if (isprint(c[i]) || isspace(c[i]) ||
1160                     c[i] == '\b' || c[i] == '\a')
1161                         continue;
1163                 if (c[i] == '\0' && i > 0) {
1164                         /*
1165                          * This looks like it might be a string.  Before we
1166                          * assume that it is indeed a string, check the
1167                          * remainder of the byte range; if it contains
1168                          * additional non-nul characters, we'll assume that
1169                          * it's a binary stream that just happens to look like
1170                          * a string, and we'll print out the individual bytes.
1171                          */
1172                         for (j = i + 1; j < nbytes; j++) {
1173                                 if (c[j] != '\0')
1174                                         break;
1175                         }
1177                         if (j != nbytes)
1178                                 break;
1180                         if (quiet) {
 284                         if (quiet)
1181                                 return (dt_printf(dtp, fp, "%s", c));
1182                         } else {
1183                                 return (dt_printf(dtp, fp, " %s%*s",
1184                                     width < 0 ? " " : "", width, c));
1185                         }
 286                         else
 287                                 return (dt_printf(dtp, fp, "  %-*s", width, c));
1186                 }
1188                 break;
```

```
1189         }

1191         if (i == nbytes) {
1192                 /*
1193                  * The byte range is all printable characters, but there is
1194                  * no trailing nul byte.  We'll assume that it's a string and
1195                  * print it as such.
1196                  */
1197                 char *s = alloca(nbytes + 1);
1198                 bcopy(c, s, nbytes);
1199                 s[nbytes] = '\0';
1200                 return (dt_printf(dtp, fp, "  %-*s", width, s));
1201         }

1203 raw:
1204         if (dt_printf(dtp, fp, "\n%*s      ", margin, "") < 0)
1205                 return (-1);

1207         for (i = 0; i < 16; i++)
1208                 if (dt_printf(dtp, fp, "  %c", "0123456789abcdef"[i]) < 0)
1209                         return (-1);

1211         if (dt_printf(dtp, fp, "  0123456789abcdef\n") < 0)
1212                 return (-1);


1215         for (i = 0; i < nbytes; i += 16) {
1216                 if (dt_printf(dtp, fp, "%*s%5x:", margin, "", i) < 0)
1217                         return (-1);

1219                 for (j = i; j < i + 16 && j < nbytes; j++) {
1220                         if (dt_printf(dtp, fp, " %02x", (uchar_t)c[j]) < 0)
1221                                 return (-1);
1222                 }

1224                 while (j++ % 16) {
1225                         if (dt_printf(dtp, fp, "   ") < 0)
1226                                 return (-1);
1227                 }

1229                 if (dt_printf(dtp, fp, "  ") < 0)
1230                         return (-1);

1232                 for (j = i; j < i + 16 && j < nbytes; j++) {
1233                         if (dt_printf(dtp, fp, "%c",
1234                             c[j] < ' ' || c[j] > '~' ? '.' : c[j]) < 0)
1235                                 return (-1);
1236                 }

1238                 if (dt_printf(dtp, fp, "\n") < 0)
1239                         return (-1);
1240         }

1242         return (0);
1243 }
```

*unchanged_portion_omitted_*

```
1788 static int
1789 dt_print_datum(dtrace_hdl_t *dtp, FILE *fp, dtrace_recdesc_t *rec,
1790     caddr_t addr, size_t size, const dtrace_aggdata_t *aggdata,
1791     uint64_t normal, dt_print_aggdata_t *pd)
 892     caddr_t addr, size_t size, uint64_t normal)
1792 {
1793         int err, width;
 894         int err;
1794         dtrace_actkind_t act = rec->dtrd_action;
```



```
1795         boolean_t packed = pd->dtpa_agghist || pd->dtpa_aggpack;
1796         dtrace_aggdesc_t *agg = aggdata->dtada_desc;

1798         static struct {
1799                 size_t size;
1800                 int width;
1801                 int packedwidth;
1802         } *fmt, fmttab[] = {
1803                 { sizeof (uint8_t),     3,      3 },
1804                 { sizeof (uint16_t),    5,      5 },
1805                 { sizeof (uint32_t),    8,      8 },
1806                 { sizeof (uint64_t),    16,     16 },
1807                 { 0,                    -50,    16 }
1808         };

1810         if (packed && pd->dtpa_agghisthdr != agg->dtagd_varid) {
1811                 dtrace_recdesc_t *r;

1813                 width = 0;

1815                 /*
1816                  * To print our quantization header for either an agghist or
1817                  * aggpack aggregation, we need to iterate through all of our
1818                  * of our records to determine their width.
1819                  */
1820                 for (r = rec; !DTRACEACT_ISAGG(r->dtrd_action); r++) {
1821                         for (fmt = fmttab; fmt->size &&
1822                             fmt->size != r->dtrd_size; fmt++)
1823                                 continue;

1825                         width += fmt->packedwidth + 1;
1826                 }

1828                 if (pd->dtpa_agghist) {
1829                         if (dt_print_quanthdr(dtp, fp, width) < 0)
1830                                 return (-1);
1831                 } else {
1832                         if (dt_print_quanthdr_packed(dtp, fp,
1833                             width, aggdata, r->dtrd_action) < 0)
1834                                 return (-1);
1835                 }

1837                 pd->dtpa_agghisthdr = agg->dtagd_varid;
1838         }

1840         if (pd->dtpa_agghist && DTRACEACT_ISAGG(act)) {
1841                 char positives = aggdata->dtada_flags & DTRACE_A_HASPOSITIVES;
1842                 char negatives = aggdata->dtada_flags & DTRACE_A_HASNEGATIVES;
1843                 int64_t val;

1845                 assert(act == DTRACEAGG_SUM || act == DTRACEAGG_COUNT);
1846                 val = (long long)*((uint64_t *)addr);

1848                 if (dt_printf(dtp, fp, " ") < 0)
1849                         return (-1);

1851                 return (dt_print_quantline(dtp, fp, val, normal,
1852                     aggdata->dtada_total, positives, negatives));
1853         }

1855         if (pd->dtpa_aggpack && DTRACEACT_ISAGG(act)) {
1856                 switch (act) {
1857                 case DTRACEAGG_QUANTIZE:
1858                         return (dt_print_quantize_packed(dtp,
1859                             fp, addr, size, aggdata));
1860                 case DTRACEAGG_LQUANTIZE:
```

```
1861                        return (dt_print_lquantize_packed(dtp,
1862                            fp, addr, size, aggdata));
1863                default:
1864                        break;
1865                }
1866        }
1867 #endif /* ! codereview */

1869        switch (act) {
1870        case DTRACEACT_STACK:
1871                return (dt_print_stack(dtp, fp, NULL, addr,
1872                    rec->dtrd_arg, rec->dtrd_size / rec->dtrd_arg));

1874        case DTRACEACT_USTACK:
1875        case DTRACEACT_JSTACK:
1876                return (dt_print_ustack(dtp, fp, NULL, addr, rec->dtrd_arg));

1878        case DTRACEACT_USYM:
1879        case DTRACEACT_UADDR:
1880                return (dt_print_usym(dtp, fp, addr, act));

1882        case DTRACEACT_UMOD:
1883                return (dt_print_umod(dtp, fp, NULL, addr));

1885        case DTRACEACT_SYM:
1886                return (dt_print_sym(dtp, fp, NULL, addr));

1888        case DTRACEACT_MOD:
1889                return (dt_print_mod(dtp, fp, NULL, addr));

1891        case DTRACEAGG_QUANTIZE:
1892                return (dt_print_quantize(dtp, fp, addr, size, normal));

1894        case DTRACEAGG_LQUANTIZE:
1895                return (dt_print_lquantize(dtp, fp, addr, size, normal));

1897        case DTRACEAGG_LLQUANTIZE:
1898                return (dt_print_llquantize(dtp, fp, addr, size, normal));

1900        case DTRACEAGG_AVG:
1901                return (dt_print_average(dtp, fp, addr, size, normal));

1903        case DTRACEAGG_STDDEV:
1904                return (dt_print_stddev(dtp, fp, addr, size, normal));

1906        default:
1907                break;
1908        }

1910        for (fmt = fmttab; fmt->size && fmt->size != size; fmt++)
1911                continue;

1913        width = packed ? fmt->packedwidth : fmt->width;

1915 #endif /* ! codereview */
1916        switch (size) {
1917        case sizeof (uint64_t):
1918                err = dt_printf(dtp, fp, " %*lld", width,
 896                err = dt_printf(dtp, fp, " %16lld",
1919                    /* LINTED - alignment */
1920                    (long long)*((uint64_t *)addr) / normal);
1921                break;
1922        case sizeof (uint32_t):
1923                /* LINTED - alignment */
1924                err = dt_printf(dtp, fp, " %*d", width, *((uint32_t *)addr) /
 902                err = dt_printf(dtp, fp, " %8d", *((uint32_t *)addr) /
```



```
1925                        (uint32_t)normal);
1926                break;
1927        case sizeof (uint16_t):
1928                /* LINTED - alignment */
1929                err = dt_printf(dtp, fp, " %*d", width, *((uint16_t *)addr) /
 907                err = dt_printf(dtp, fp, " %5d", *((uint16_t *)addr) /
1930                    (uint32_t)normal);
1931                break;
1932        case sizeof (uint8_t):
1933                err = dt_printf(dtp, fp, " %*d", width, *((uint8_t *)addr) /
 911                err = dt_printf(dtp, fp, " %3d", *((uint8_t *)addr) /
1934                    (uint32_t)normal);
1935                break;
1936        default:
1937                err = dt_print_bytes(dtp, fp, addr, size, width, 0, 0);
 915                err = dt_print_bytes(dtp, fp, addr, size, 50, 0, 0);
1938                break;
1939        }

1941        return (err);
1942 }

1944 int
1945 dt_print_aggs(const dtrace_aggdata_t **aggsdata, int naggvars, void *arg)
1946 {
1947        int i, aggact = 0;
1948        dt_print_aggdata_t *pd = arg;
1949        const dtrace_aggdata_t *aggdata = aggsdata[0];
1950        dtrace_aggdesc_t *agg = aggdata->dtada_desc;
1951        FILE *fp = pd->dtpa_fp;
1952        dtrace_hdl_t *dtp = pd->dtpa_dtp;
1953        dtrace_recdesc_t *rec;
1954        dtrace_actkind_t act;
1955        caddr_t addr;
1956        size_t size;

1958        pd->dtpa_agghist = (aggdata->dtada_flags & DTRACE_A_TOTAL);
1959        pd->dtpa_aggpack = (aggdata->dtada_flags & DTRACE_A_MINMAXBIN);

1961 #endif /* ! codereview */
1962        /*
1963         * Iterate over each record description in the key, printing the traced
1964         * data, skipping the first datum (the tuple member created by the
1965         * compiler).
1966         */
1967        for (i = 1; i < agg->dtagd_nrecs; i++) {
1968                rec = &agg->dtagd_rec[i];
1969                act = rec->dtrd_action;
1970                addr = aggdata->dtada_data + rec->dtrd_offset;
1971                size = rec->dtrd_size;

1973                if (DTRACEACT_ISAGG(act)) {
1974                        aggact = i;
1975                        break;
1976                }

1978                if (dt_print_datum(dtp, fp, rec, addr,
1979                    size, aggdata, 1, pd) < 0)
 936                if (dt_print_datum(dtp, fp, rec, addr, size, 1) < 0)
1980                        return (-1);

1982                if (dt_buffered_flush(dtp, NULL, rec, aggdata,
1983                    DTRACE_BUFDATA_AGGKEY) < 0)
1984                        return (-1);
1985        }
```

```
1987         assert(aggact != 0);

1989         for (i = (naggvars == 1 ? 0 : 1); i < naggvars; i++) {
1990                 uint64_t normal;

1992                 aggdata = aggsdata[i];
1993                 agg = aggdata->dtada_desc;
1994                 rec = &agg->dtagd_rec[aggact];
1995                 act = rec->dtrd_action;
1996                 addr = aggdata->dtada_data + rec->dtrd_offset;
1997                 size = rec->dtrd_size;

1999                 assert(DTRACEACT_ISAGG(act));
2000                 normal = aggdata->dtada_normal;

2002                 if (dt_print_datum(dtp, fp, rec, addr,
2003                     size, aggdata, normal, pd) < 0)
 959                 if (dt_print_datum(dtp, fp, rec, addr, size, normal) < 0)
2004                         return (-1);

2006                 if (dt_buffered_flush(dtp, NULL, rec, aggdata,
2007                     DTRACE_BUFDATA_AGGVAL) < 0)
2008                         return (-1);

2010                 if (!pd->dtpa_allunprint)
2011                         agg->dtagd_flags |= DTRACE_AGD_PRINTED;
2012         }

2014         if (!pd->dtpa_agghist && !pd->dtpa_aggpack) {
2015 #endif /* ! codereview */
2016                 if (dt_printf(dtp, fp, "\n") < 0)
2017                         return (-1);
2018         }
2019 #endif /* ! codereview */

2021         if (dt_buffered_flush(dtp, NULL, NULL, aggdata,
2022             DTRACE_BUFDATA_AGGFORMAT | DTRACE_BUFDATA_AGGLAST) < 0)
2023                 return (-1);

2025         return (0);
2026 }

2028 int
2029 dt_print_agg(const dtrace_aggdata_t *aggdata, void *arg)
2030 {
2031         dt_print_aggdata_t *pd = arg;
2032         dtrace_aggdesc_t *agg = aggdata->dtada_desc;
2033         dtrace_aggvarid_t aggvarid = pd->dtpa_id;

2035         if (pd->dtpa_allunprint) {
2036                 if (agg->dtagd_flags & DTRACE_AGD_PRINTED)
2037                         return (0);
2038         } else {
2039                 /*
2040                  * If we're not printing all unprinted aggregations, then the
2041                  * aggregation variable ID denotes a specific aggregation
2042                  * variable that we should print -- skip any other aggregations
2043                  * that we encounter.
2044                  */
2045                 if (agg->dtagd_nrecs == 0)
2046                         return (0);

2048                 if (aggvarid != agg->dtagd_varid)
2049                         return (0);
2050         }
```



```
2052         return (dt_print_aggs(&aggdata, 1, arg));
2053 }

2055 int
2056 dt_setopt(dtrace_hdl_t *dtp, const dtrace_probedata_t *data,
2057     const char *option, const char *value)
2058 {
2059         int len, rval;
2060         char *msg;
2061         const char *errstr;
2062         dtrace_setoptdata_t optdata;

2064         bzero(&optdata, sizeof (optdata));
2065         (void) dtrace_getopt(dtp, option, &optdata.dtsda_oldval);

2067         if (dtrace_setopt(dtp, option, value) == 0) {
2068                 (void) dtrace_getopt(dtp, option, &optdata.dtsda_newval);
2069                 optdata.dtsda_probe = data;
2070                 optdata.dtsda_option = option;
2071                 optdata.dtsda_handle = dtp;

2073                 if ((rval = dt_handle_setopt(dtp, &optdata)) != 0)
2074                         return (rval);

2076                 return (0);
2077         }

2079         errstr = dtrace_errmsg(dtp, dtrace_errno(dtp));
2080         len = strlen(option) + strlen(value) + strlen(errstr) + 80;
2081         msg = alloca(len);

2083         (void) snprintf(msg, len, "couldn't set option \"%s\" to \"%s\": %s\n",
2084             option, value, errstr);

2086         if ((rval = dt_handle_liberr(dtp, data, msg)) == 0)
2087                 return (0);

2089         return (rval);
2090 }

2092 static int
2093 dt_consume_cpu(dtrace_hdl_t *dtp, FILE *fp, int cpu,
2094     dtrace_bufdesc_t *buf, boolean_t just_one,
2095     dtrace_consume_probe_f *efunc, dtrace_consume_rec_f *rfunc, void *arg)
2096 {
2097         dtrace_epid_t id;
2098         size_t offs;
2099         int flow = (dtp->dt_options[DTRACEOPT_FLOWINDENT] != DTRACEOPT_UNSET);
2100         int quiet = (dtp->dt_options[DTRACEOPT_QUIET] != DTRACEOPT_UNSET);
2101         int rval, i, n;
2102         uint64_t tracememsize = 0;
2103         dtrace_probedata_t data;
2104         uint64_t drops;

2106         bzero(&data, sizeof (data));
2107         data.dtpda_handle = dtp;
2108         data.dtpda_cpu = cpu;
2109         data.dtpda_flow = dtp->dt_flow;
2110         data.dtpda_indent = dtp->dt_indent;
2111         data.dtpda_prefix = dtp->dt_prefix;

2113         for (offs = buf->dtbd_oldest; offs < buf->dtbd_size; ) {
2114                 dtrace_eprobedesc_t *epd;

2116                 /*
2117                  * We're guaranteed to have an ID.
```

```
2118                 */
2119                id = *(uint32_t *)((uintptr_t)buf->dtbd_data + offs);
2120
2121                if (id == DTRACE_EPIDNONE) {
2122                        /*
2123                         * This is filler to assure proper alignment of the
2124                         * next record; we simply ignore it.
2125                         */
2126                        offs += sizeof (id);
2127                        continue;
2128                }
2129
2130                if ((rval = dt_epid_lookup(dtp, id, &data.dtpda_edesc,
2131                    &data.dtpda_pdesc)) != 0)
2132                        return (rval);
2133
2134                epd = data.dtpda_edesc;
2135                data.dtpda_data = buf->dtbd_data + offs;
2136
2137                if (data.dtpda_edesc->dtepd_uarg != DT_ECB_DEFAULT) {
2138                        rval = dt_handle(dtp, &data);
2139
2140                        if (rval == DTRACE_CONSUME_NEXT)
2141                                goto nextepid;
2142
2143                        if (rval == DTRACE_CONSUME_ERROR)
2144                                return (-1);
2145                }
2146
2147                if (flow)
2148                        (void) dt_flowindent(dtp, &data, dtp->dt_last_epid,
2149                            buf, offs);
2150
2151                rval = (*efunc)(&data, arg);
2152
2153                if (flow) {
2154                        if (data.dtpda_flow == DTRACEFLOW_ENTRY)
2155                                data.dtpda_indent += 2;
2156                }
2157
2158                if (rval == DTRACE_CONSUME_NEXT)
2159                        goto nextepid;
2160
2161                if (rval == DTRACE_CONSUME_ABORT)
2162                        return (dt_set_errno(dtp, EDT_DIRABORT));
2163
2164                if (rval != DTRACE_CONSUME_THIS)
2165                        return (dt_set_errno(dtp, EDT_BADRVAL));
2166
2167                for (i = 0; i < epd->dtepd_nrecs; i++) {
2168                        caddr_t addr;
2169                        dtrace_recdesc_t *rec = &epd->dtepd_rec[i];
2170                        dtrace_actkind_t act = rec->dtrd_action;
2171
2172                        data.dtpda_data = buf->dtbd_data + offs +
2173                            rec->dtrd_offset;
2174                        addr = data.dtpda_data;
2175
2176                        if (act == DTRACEACT_LIBACT) {
2177                                uint64_t arg = rec->dtrd_arg;
2178                                dtrace_aggvarid_t id;
2179
2180                                switch (arg) {
2181                                case DT_ACT_CLEAR:
2182                                        /* LINTED - alignment */
2183                                        id = *((dtrace_aggvarid_t *)addr);
```



```
2184                                        (void) dtrace_aggregate_walk(dtp,
2185                                            dt_clear_agg, &id);
2186                                        continue;
2187
2188                                case DT_ACT_DENORMALIZE:
2189                                        /* LINTED - alignment */
2190                                        id = *((dtrace_aggvarid_t *)addr);
2191                                        (void) dtrace_aggregate_walk(dtp,
2192                                            dt_denormalize_agg, &id);
2193                                        continue;
2194
2195                                case DT_ACT_FTRUNCATE:
2196                                        if (fp == NULL)
2197                                                continue;
2198
2199                                        (void) fflush(fp);
2200                                        (void) ftruncate(fileno(fp), 0);
2201                                        (void) fseeko(fp, 0, SEEK_SET);
2202                                        continue;
2203
2204                                case DT_ACT_NORMALIZE:
2205                                        if (i == epd->dtepd_nrecs - 1)
2206                                                return (dt_set_errno(dtp,
2207                                                    EDT_BADNORMAL));
2208
2209                                        if (dt_normalize(dtp,
2210                                            buf->dtbd_data + offs, rec) != 0)
2211                                                return (-1);
2212
2213                                        i++;
2214                                        continue;
2215
2216                                case DT_ACT_SETOPT: {
2217                                        uint64_t *opts = dtp->dt_options;
2218                                        dtrace_recdesc_t *valrec;
2219                                        uint32_t valsize;
2220                                        caddr_t val;
2221                                        int rv;
2222
2223                                        if (i == epd->dtepd_nrecs - 1) {
2224                                                return (dt_set_errno(dtp,
2225                                                    EDT_BADSETOPT));
2226                                        }
2227
2228                                        valrec = &epd->dtepd_rec[++i];
2229                                        valsize = valrec->dtrd_size;
2230
2231                                        if (valrec->dtrd_action != act ||
2232                                            valrec->dtrd_arg != arg) {
2233                                                return (dt_set_errno(dtp,
2234                                                    EDT_BADSETOPT));
2235                                        }
2236
2237                                        if (valsize > sizeof (uint64_t)) {
2238                                                val = buf->dtbd_data + offs +
2239                                                    valrec->dtrd_offset;
2240                                        } else {
2241                                                val = "1";
2242                                        }
2243
2244                                        rv = dt_setopt(dtp, &data, addr, val);
2245
2246                                        if (rv != 0)
2247                                                return (-1);
2248
2249                                        flow = (opts[DTRACEOPT_FLOWINDENT] !=
```

```
2250                                        DTRACEOPT_UNSET);
2251                                quiet = (opts[DTRACEOPT_QUIET] !=
2252                                    DTRACEOPT_UNSET);
2253
2254                                continue;
2255                        }
2256
2257                        case DT_ACT_TRUNC:
2258                                if (i == epd->dtepd_nrecs - 1)
2259                                        return (dt_set_errno(dtp,
2260                                            EDT_BADTRUNC));
2261
2262                                if (dt_trunc(dtp,
2263                                    buf->dtbd_data + offs, rec) != 0)
2264                                        return (-1);
2265
2266                                i++;
2267                                continue;
2268
2269                        default:
2270                                continue;
2271                        }
2272                }
2273
2274                if (act == DTRACEACT_TRACEMEM_DYNSIZE &&
2275                    rec->dtrd_size == sizeof (uint64_t)) {
2276                        /* LINTED - alignment */
2277                        tracememsize = *((unsigned long long *)addr);
2278                        continue;
2279                }
2280
2281                rval = (*rfunc)(&data, rec, arg);
2282
2283                if (rval == DTRACE_CONSUME_NEXT)
2284                        continue;
2285
2286                if (rval == DTRACE_CONSUME_ABORT)
2287                        return (dt_set_errno(dtp, EDT_DIRABORT));
2288
2289                if (rval != DTRACE_CONSUME_THIS)
2290                        return (dt_set_errno(dtp, EDT_BADRVAL));
2291
2292                if (act == DTRACEACT_STACK) {
2293                        int depth = rec->dtrd_arg;
2294
2295                        if (dt_print_stack(dtp, fp, NULL, addr, depth,
2296                            rec->dtrd_size / depth) < 0)
2297                                return (-1);
2298                        goto nextrec;
2299                }
2300
2301                if (act == DTRACEACT_USTACK ||
2302                    act == DTRACEACT_JSTACK) {
2303                        if (dt_print_ustack(dtp, fp, NULL,
2304                            addr, rec->dtrd_arg) < 0)
2305                                return (-1);
2306                        goto nextrec;
2307                }
2308
2309                if (act == DTRACEACT_SYM) {
2310                        if (dt_print_sym(dtp, fp, NULL, addr) < 0)
2311                                return (-1);
2312                        goto nextrec;
2313                }
2314
2315                if (act == DTRACEACT_MOD) {
```



```
2316                        if (dt_print_mod(dtp, fp, NULL, addr) < 0)
2317                                return (-1);
2318                        goto nextrec;
2319                }
2320
2321                if (act == DTRACEACT_USYM || act == DTRACEACT_UADDR) {
2322                        if (dt_print_usym(dtp, fp, addr, act) < 0)
2323                                return (-1);
2324                        goto nextrec;
2325                }
2326
2327                if (act == DTRACEACT_UMOD) {
2328                        if (dt_print_umod(dtp, fp, NULL, addr) < 0)
2329                                return (-1);
2330                        goto nextrec;
2331                }
2332
2333                if (DTRACEACT_ISPRINTFLIKE(act)) {
2334                        void *fmtdata;
2335                        int (*func)(dtrace_hdl_t *, FILE *, void *,
2336                            const dtrace_probedata_t *,
2337                            const dtrace_recdesc_t *, uint_t,
2338                            const void *buf, size_t);
2339
2340                        if ((fmtdata = dt_format_lookup(dtp,
2341                            rec->dtrd_format)) == NULL)
2342                                goto nofmt;
2343
2344                        switch (act) {
2345                        case DTRACEACT_PRINTF:
2346                                func = dtrace_fprintf;
2347                                break;
2348                        case DTRACEACT_PRINTA:
2349                                func = dtrace_fprinta;
2350                                break;
2351                        case DTRACEACT_SYSTEM:
2352                                func = dtrace_system;
2353                                break;
2354                        case DTRACEACT_FREOPEN:
2355                                func = dtrace_freopen;
2356                                break;
2357                        }
2358
2359                        n = (*func)(dtp, fp, fmtdata, &data,
2360                            rec, epd->dtepd_nrecs - i,
2361                            (uchar_t *)buf->dtbd_data + offs,
2362                            buf->dtbd_size - offs);
2363
2364                        if (n < 0)
2365                                return (-1); /* errno is set for us */
2366
2367                        if (n > 0)
2368                                i += n - 1;
2369                        goto nextrec;
2370                }
2371
2372                /*
2373                 * If this is a DIF expression, and the record has a
2374                 * format set, this indicates we have a CTF type name
2375                 * associated with the data and we should try to print
2376                 * it out by type.
2377                 */
2378                if (act == DTRACEACT_DIFEXPR) {
2379                        const char *strdata = dt_strdata_lookup(dtp,
2380                            rec->dtrd_format);
2381                        if (strdata != NULL) {
```

```
2382                                    n = dtrace_print(dtp, fp, strdata,
2383                                        addr, rec->dtrd_size);

2385                                    /*
2386                                     * dtrace_print() will return -1 on
2387                                     * error, or return the number of bytes
2388                                     * consumed.  It will return 0 if the
2389                                     * type couldn't be determined, and we
2390                                     * should fall through to the normal
2391                                     * trace method.
2392                                     */
2393                                    if (n < 0)
2394                                            return (-1);

2396                                    if (n > 0)
2397                                            goto nextrec;
2398                            }
2399                    }

2401 nofmt:
2402                    if (act == DTRACEACT_PRINTA) {
2403                            dt_print_aggdata_t pd;
2404                            dtrace_aggvarid_t *aggvars;
2405                            int j, naggvars = 0;
2406                            size_t size = ((epd->dtepd_nrecs - i) *
2407                                sizeof (dtrace_aggvarid_t));

2409                            if ((aggvars = dt_alloc(dtp, size)) == NULL)
2410                                    return (-1);

2412                            /*
2413                             * This might be a printa() with multiple
2414                             * aggregation variables.  We need to scan
2415                             * forward through the records until we find
2416                             * a record from a different statement.
2417                             */
2418                            for (j = i; j < epd->dtepd_nrecs; j++) {
2419                                    dtrace_recdesc_t *nrec;
2420                                    caddr_t naddr;

2422                                    nrec = &epd->dtepd_rec[j];

2424                                    if (nrec->dtrd_uarg != rec->dtrd_uarg)
2425                                            break;

2427                                    if (nrec->dtrd_action != act) {
2428                                            return (dt_set_errno(dtp,
2429                                                EDT_BADAGG));
2430                                    }

2432                                    naddr = buf->dtbd_data + offs +
2433                                        nrec->dtrd_offset;

2435                                    aggvars[naggvars++] =
2436                                        /* LINTED - alignment */
2437                                        *((dtrace_aggvarid_t *)naddr);
2438                            }

2440                            i = j - 1;
2441                            bzero(&pd, sizeof (pd));
2442                            pd.dtpa_dtp = dtp;
2443                            pd.dtpa_fp = fp;

2445                            assert(naggvars >= 1);

2447                            if (naggvars == 1) {
```



```
2448                                    pd.dtpa_id = aggvars[0];
2449                                    dt_free(dtp, aggvars);

2451                                    if (dt_printf(dtp, fp, "\n") < 0 ||
2452                                        dtrace_aggregate_walk_sorted(dtp,
2453                                        dt_print_agg, &pd) < 0)
2454                                            return (-1);
2455                                    goto nextrec;
2456                            }

2458                            if (dt_printf(dtp, fp, "\n") < 0 ||
2459                                dtrace_aggregate_walk_joined(dtp, aggvars,
2460                                naggvars, dt_print_aggs, &pd) < 0) {
2461                                    dt_free(dtp, aggvars);
2462                                    return (-1);
2463                            }

2465                            dt_free(dtp, aggvars);
2466                            goto nextrec;
2467                    }

2469                    if (act == DTRACEACT_TRACEMEM) {
2470                            if (tracememsize == 0 ||
2471                                tracememsize > rec->dtrd_size) {
2472                                    tracememsize = rec->dtrd_size;
2473                            }

2475                            n = dt_print_bytes(dtp, fp, addr,
2476                                tracememsize, -33, quiet, 1);
 970                                tracememsize, 33, quiet, 1);

2478                            tracememsize = 0;

2480                            if (n < 0)
2481                                    return (-1);

2483                            goto nextrec;
2484                    }

2486                    switch (rec->dtrd_size) {
2487                    case sizeof (uint64_t):
2488                            n = dt_printf(dtp, fp,
2489                                quiet ? "%lld" : " %16lld",
2490                                /* LINTED - alignment */
2491                                *((unsigned long long *)addr));
2492                            break;
2493                    case sizeof (uint32_t):
2494                            n = dt_printf(dtp, fp, quiet ? "%d" : " %8d",
2495                                /* LINTED - alignment */
2496                                *((uint32_t *)addr));
2497                            break;
2498                    case sizeof (uint16_t):
2499                            n = dt_printf(dtp, fp, quiet ? "%d" : " %5d",
2500                                /* LINTED - alignment */
2501                                *((uint16_t *)addr));
2502                            break;
2503                    case sizeof (uint8_t):
2504                            n = dt_printf(dtp, fp, quiet ? "%d" : " %3d",
2505                                *((uint8_t *)addr));
2506                            break;
2507                    default:
2508                            n = dt_print_bytes(dtp, fp, addr,
2509                                rec->dtrd_size, -33, quiet, 0);
1003                                rec->dtrd_size, 33, quiet, 0);
2510                            break;
2511                    }
```

```
2513                        if (n < 0)
2514                                return (-1); /* errno is set for us */

2516 nextrec:
2517                        if (dt_buffered_flush(dtp, &data, rec, NULL, 0) < 0)
2518                                return (-1); /* errno is set for us */
2519                }

2521                /*
2522                 * Call the record callback with a NULL record to indicate
2523                 * that we're done processing this EPID.
2524                 */
2525                rval = (*rfunc)(&data, NULL, arg);
2526 nextepid:
2527                offs += epd->dtepd_size;
2528                dtp->dt_last_epid = id;
2529                if (just_one) {
2530                        buf->dtbd_oldest = offs;
2531                        break;
2532                }
2533        }

2535        dtp->dt_flow = data.dtpda_flow;
2536        dtp->dt_indent = data.dtpda_indent;
2537        dtp->dt_prefix = data.dtpda_prefix;

2539        if ((drops = buf->dtbd_drops) == 0)
2540                return (0);

2542        /*
2543         * Explicitly zero the drops to prevent us from processing them again.
2544         */
2545        buf->dtbd_drops = 0;

2547        return (dt_handle_cpudrop(dtp, cpu, DTRACEDROP_PRINCIPAL, drops));
2548 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   31247 Tue Jan 14 20:13:05 2014
new/usr/src/lib/libdtrace/common/dt_impl.h
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
______unchanged_portion_omitted_

 178 typedef struct dt_print_aggdata {
 179         dtrace_hdl_t *dtpa_dtp;         /* pointer to libdtrace handle */
 180         dtrace_aggvarid_t dtpa_id;      /* aggregation variable of interest */
 181         FILE *dtpa_fp;                  /* file pointer */
 182         int dtpa_allunprint;            /* print only unprinted aggregations */
 183         int dtpa_agghist;               /* print aggregation as histogram */
 184         int dtpa_agghisthdr;            /* aggregation histogram hdr printed */
 185         int dtpa_aggpack;               /* pack quantized aggregations */
 186 #endif /* ! codereview */
 187 } dt_print_aggdata_t;

 189 typedef struct dt_dirpath {
 190         dt_list_t dir_list;             /* linked-list forward/back pointers */
 191         char *dir_path;                 /* directory pathname */
 192 } dt_dirpath_t;

 194 typedef struct dt_lib_depend {
 195         dt_list_t dtld_deplist;         /* linked-list forward/back pointers */
 196         char *dtld_library;             /* library name */
 197         char *dtld_libpath;             /* library pathname */
 198         uint_t dtld_finish;             /* completion time in tsort for lib */
 199         uint_t dtld_start;              /* starting time in tsort for lib */
 200         uint_t dtld_loaded;             /* boolean: is this library loaded */
 201         dt_list_t dtld_dependencies;    /* linked-list of lib dependencies */
 202         dt_list_t dtld_dependents;      /* linked-list of lib dependents */
 203 } dt_lib_depend_t;

 205 typedef uint32_t dt_version_t;          /* encoded version (see below) */

 207 struct dtrace_hdl {
 208         const dtrace_vector_t *dt_vector; /* library vector, if vectored open */
 209         void *dt_varg;  /* vector argument, if vectored open */
 210         dtrace_conf_t dt_conf;  /* DTrace driver configuration profile */
 211         char dt_errmsg[BUFSIZ]; /* buffer for formatted syntax error msgs */
 212         const char *dt_errtag;  /* tag used with last call to dt_set_errmsg() */
 213         dt_pcb_t *dt_pcb;       /* pointer to current parsing control block */
 214         ulong_t dt_gen;         /* compiler generation number */
 215         dt_list_t dt_programs;  /* linked list of dtrace_prog_t's */
 216         dt_list_t dt_xlators;   /* linked list of dt_xlator_t's */
 217         struct dt_xlator **dt_xlatormap; /* dt_xlator_t's indexed by dx_id */
 218         id_t dt_xlatorid;       /* next dt_xlator_t id to assign */
 219         dt_ident_t *dt_externs; /* linked list of external symbol identifiers */
 220         dt_idhash_t *dt_macros; /* hash table of macro variable identifiers */
 221         dt_idhash_t *dt_aggs;   /* hash table of aggregation identifiers */
 222         dt_idhash_t *dt_globals; /* hash table of global identifiers */
 223         dt_idhash_t *dt_tls;    /* hash table of thread-local identifiers */
 224         dt_list_t dt_modlist;   /* linked list of dt_module_t's */
 225         dt_module_t **dt_mods;  /* hash table of dt_module_t's */
 226         uint_t dt_modbuckets;   /* number of module hash buckets */
 227         uint_t dt_nmods;        /* number of modules in hash and list */
 228         dt_provmod_t *dt_provmod; /* linked list of provider modules */
 229         dt_module_t *dt_exec;   /* pointer to executable module */
 230         dt_module_t *dt_rtld;   /* pointer to run-time linker module */
 231         dt_module_t *dt_cdefs;  /* pointer to C dynamic type module */
 232         dt_module_t *dt_ddefs;  /* pointer to D dynamic type module */
 233         dt_list_t dt_provlist;  /* linked list of dt_provider_t's */
 234         struct dt_provider **dt_provs; /* hash table of dt_provider_t's */
```



```
 235         uint_t dt_provbuckets;  /* number of provider hash buckets */
 236         uint_t dt_nprovs;       /* number of providers in hash and list */
 237         dt_proc_hash_t *dt_procs; /* hash table of grabbed process handles */
 238         char **dt_proc_env;     /* additional environment variables */
 239         dt_intdesc_t dt_ints[6]; /* cached integer type descriptions */
 240         ctf_id_t dt_type_func;  /* cached CTF identifier for function type */
 241         ctf_id_t dt_type_fptr;  /* cached CTF identifier for function pointer */
 242         ctf_id_t dt_type_str;   /* cached CTF identifier for string type */
 243         ctf_id_t dt_type_dyn;   /* cached CTF identifier for <DYN> type */
 244         ctf_id_t dt_type_stack; /* cached CTF identifier for stack type */
 245         ctf_id_t dt_type_symaddr; /* cached CTF identifier for _symaddr type */
 246         ctf_id_t dt_type_usymaddr; /* cached CTF ident. for _usymaddr type */
 247         size_t dt_maxprobe;     /* max enabled probe ID */
 248         dtrace_eprobedesc_t **dt_edesc; /* enabled probe descriptions */
 249         dtrace_probedesc_t **dt_pdesc; /* probe descriptions for enabled prbs */
 250         size_t dt_maxagg;       /* max aggregation ID */
 251         dtrace_aggdesc_t **dt_aggdesc; /* aggregation descriptions */
 252         int dt_maxformat;       /* max format ID */
 253         void **dt_formats;      /* pointer to format array */
 254         int dt_maxstrdata;      /* max strdata ID */
 255         char **dt_strdata;      /* pointer to strdata array */
 256         dt_aggregate_t dt_aggregate; /* aggregate */
 257         dt_pq_t *dt_bufq;       /* CPU-specific data queue */
 258         struct dt_pfdict *dt_pfdict; /* dictionary of printf conversions */
 259         dt_version_t dt_vmax;   /* optional ceiling on program API binding */
 260         dtrace_attribute_t dt_amin; /* optional floor on program attributes */
 261         char *dt_cpp_path;      /* pathname of cpp(1) to invoke if needed */
 262         char **dt_cpp_argv;     /* argument vector for exec'ing cpp(1) */
 263         int dt_cpp_argc;        /* count of initialized cpp(1) arguments */
 264         int dt_cpp_args;        /* size of dt_cpp_argv[] array */
 265         char *dt_ld_path;       /* pathname of ld(1) to invoke if needed */
 266         dt_list_t dt_lib_path;  /* linked-list forming library search path */
 267         uint_t dt_lazyload;     /* boolean:  set via -xlazyload */
 268         uint_t dt_droptags;     /* boolean:  set via -xdroptags */
 269         uint_t dt_active;       /* boolean:  set once tracing is active */
 270         uint_t dt_stopped;      /* boolean:  set once tracing is stopped */
 271         processorid_t dt_beganon; /* CPU that executed BEGIN probe (if any) */
 272         processorid_t dt_endedon; /* CPU that executed END probe (if any) */
 273         uint_t dt_oflags;       /* dtrace open-time options (see dtrace.h) */
 274         uint_t dt_cflags;       /* dtrace compile-time options (see dtrace.h) */
 275         uint_t dt_dflags;       /* dtrace link-time options (see dtrace.h) */
 276         uint_t dt_prcmode;      /* dtrace process create mode (see dt_proc.h) */
 277         uint_t dt_linkmode;     /* dtrace symbol linking mode (see below) */
 278         uint_t dt_linktype;     /* dtrace link output file type (see below) */
 279         uint_t dt_xlatemode;    /* dtrace translator linking mode (see below) */
 280         uint_t dt_stdcmode;     /* dtrace stdc compatibility mode (see below) */
 281         uint_t dt_encoding;     /* dtrace output encoding (see below) */
 282 #endif /* ! codereview */
 283         uint_t dt_treedump;     /* dtrace tree debug bitmap (see below) */
 284         uint64_t dt_options[DTRACEOPT_MAX]; /* dtrace run-time options */
 285         int dt_version;         /* library version requested by client */
 286         int dt_ctferr;          /* error resulting from last CTF failure */
 287         int dt_errno;           /* error resulting from last failed operation */
 288         int dt_fd;              /* file descriptor for dtrace pseudo-device */
 289         int dt_ftfd;            /* file descriptor for fasttrap pseudo-device */
 290         int dt_fterr;           /* saved errno from failed open of dt_ftfd */
 291         int dt_cdefs_fd;        /* file descriptor for C CTF debugging cache */
 292         int dt_ddefs_fd;        /* file descriptor for D CTF debugging cache */
 293         int dt_stdout_fd;       /* file descriptor for saved stdout */
 294         dtrace_handle_err_f *dt_errhdlr; /* error handler, if any */
 295         void *dt_errarg;        /* error handler argument */
 296         dtrace_prog_t *dt_errprog; /* error handler program, if any */
 297         dtrace_handle_drop_f *dt_drophdlr; /* drop handler, if any */
 298         void *dt_droparg;       /* drop handler argument */
 299         dtrace_handle_proc_f *dt_prochdlr; /* proc handler, if any */
 300         void *dt_procarg;       /* proc handler argument */
```

```
301         dtrace_handle_setopt_f *dt_setopthdlr; /* setopt handler, if any */
302         void *dt_setoptarg;     /* setopt handler argument */
303         dtrace_status_t dt_status[2]; /* status cache */
304         int dt_statusgen;       /* current status generation */
305         hrtime_t dt_laststatus; /* last status */
306         hrtime_t dt_lastswitch; /* last switch of buffer data */
307         hrtime_t dt_lastagg;    /* last snapshot of aggregation data */
308         char *dt_sprintf_buf;   /* buffer for dtrace_sprintf() */
309         int dt_sprintf_buflen;  /* length of dtrace_sprintf() buffer */
310         const char *dt_filetag; /* default filetag for dt_set_errmsg() */
311         char *dt_buffered_buf;  /* buffer for buffered output */
312         size_t dt_buffered_offs; /* current offset into buffered buffer */
313         size_t dt_buffered_size; /* size of buffered buffer */
314         dtrace_handle_buffered_f *dt_bufhdlr; /* buffered handler, if any */
315         void *dt_bufarg;        /* buffered handler argument */
316         dt_dof_t dt_dof;        /* DOF generation buffers (see dt_dof.c) */
317         struct utsname dt_uts;  /* uname(2) information for system */
318         dt_list_t dt_lib_dep;   /* scratch linked-list of lib dependencies */
319         dt_list_t dt_lib_dep_sorted;    /* dependency sorted library list */
320         dtrace_flowkind_t dt_flow;      /* flow kind */
321         const char *dt_prefix;  /* recommended flow prefix */
322         int dt_indent;          /* recommended flow indent */
323         dtrace_epid_t dt_last_epid;     /* most recently consumed EPID */
324         uint64_t dt_last_timestamp;     /* most recently consumed timestamp */
325 };

327 /*
328  * Values for the user arg of the ECB.
329  */
330 #define DT_ECB_DEFAULT          0
331 #define DT_ECB_ERROR            1

333 /*
334  * Values for the dt_linkmode property, which is used by the assembler when
335  * processing external symbol references.  User can set using -xlink=<mode>.
336  */
337 #define DT_LINK_KERNEL  0       /* kernel syms static, user syms dynamic */
338 #define DT_LINK_PRIMARY 1       /* primary kernel syms static, others dynamic */
339 #define DT_LINK_DYNAMIC 2       /* all symbols dynamic */
340 #define DT_LINK_STATIC  3       /* all symbols static */

342 /*
343  * Values for the dt_linktype property, which is used by dtrace_program_link()
344  * to determine the type of output file that is desired by the client.
345  */
346 #define DT_LTYP_ELF     0       /* produce ELF containing DOF */
347 #define DT_LTYP_DOF     1       /* produce stand-alone DOF */

349 /*
350  * Values for the dt_xlatemode property, which is used to determine whether
351  * references to dynamic translators are permitted.  Set using -xlate=<mode>.
352  */
353 #define DT_XL_STATIC    0       /* require xlators to be statically defined */
354 #define DT_XL_DYNAMIC   1       /* produce references to dynamic translators */

356 /*
357  * Values for the dt_stdcmode property, which is used by the compiler when
358  * running cpp to determine the presence and setting of the __STDC__ macro.
359  */
360 #define DT_STDC_XA      0       /* ISO C + K&R C compat w/o ISO: __STDC__=0 */
361 #define DT_STDC_XC      1       /* Strict ISO C: __STDC__=1 */
362 #define DT_STDC_XS      2       /* K&R C: __STDC__ not defined */
363 #define DT_STDC_XT      3       /* ISO C + K&R C compat with ISO: __STDC__=0 */

365 /*
366  * Values for the dt_encoding property, which is used to force a particular
```



```
367  * character encoding (overriding default behavior and/or automatic detection).
368  */
369 #define DT_ENCODING_UNSET       0
370 #define DT_ENCODING_ASCII       1
371 #define DT_ENCODING_UTF8        2

373 /*
374 #endif /* ! codereview */
375  * Macro to test whether a given pass bit is set in the dt_treedump bit-vector.
376  * If the bit for pass 'p' is set, the D compiler displays the parse tree for
377  * the program by printing it to stderr at the end of compiler pass 'p'.
378  */
379 #define DT_TREEDUMP_PASS(dtp, p)        ((dtp)->dt_treedump & (1 << ((p) - 1)))

381 /*
382  * Macros for accessing the cached CTF container and type ID for the common
383  * types "int", "string", and <DYN>, which we need to use frequently in the D
384  * compiler.  The DT_INT_* macro relies upon "int" being at index 0 in the
385  * _dtrace_ints_* tables in dt_open.c; the others are also set up there.
386  */
387 #define DT_INT_CTFP(dtp)        ((dtp)->dt_ints[0].did_ctfp)
388 #define DT_INT_TYPE(dtp)        ((dtp)->dt_ints[0].did_type)

390 #define DT_FUNC_CTFP(dtp)       ((dtp)->dt_ddefs->dm_ctfp)
391 #define DT_FUNC_TYPE(dtp)       ((dtp)->dt_type_func)

393 #define DT_FPTR_CTFP(dtp)       ((dtp)->dt_ddefs->dm_ctfp)
394 #define DT_FPTR_TYPE(dtp)       ((dtp)->dt_type_fptr)

396 #define DT_STR_CTFP(dtp)        ((dtp)->dt_ddefs->dm_ctfp)
397 #define DT_STR_TYPE(dtp)        ((dtp)->dt_type_str)

399 #define DT_DYN_CTFP(dtp)        ((dtp)->dt_ddefs->dm_ctfp)
400 #define DT_DYN_TYPE(dtp)        ((dtp)->dt_type_dyn)

402 #define DT_STACK_CTFP(dtp)      ((dtp)->dt_ddefs->dm_ctfp)
403 #define DT_STACK_TYPE(dtp)      ((dtp)->dt_type_stack)

405 #define DT_SYMADDR_CTFP(dtp)    ((dtp)->dt_ddefs->dm_ctfp)
406 #define DT_SYMADDR_TYPE(dtp)    ((dtp)->dt_type_symaddr)

408 #define DT_USYMADDR_CTFP(dtp)   ((dtp)->dt_ddefs->dm_ctfp)
409 #define DT_USYMADDR_TYPE(dtp)   ((dtp)->dt_type_usymaddr)

411 /*
412  * Actions and subroutines are both DT_NODE_FUNC nodes; to avoid confusing
413  * an action for a subroutine (or vice versa), we assure that the DT_ACT_*
414  * constants and the DIF_SUBR_* constants occupy non-overlapping ranges by
415  * starting the DT_ACT_* constants at DIF_SUBR_MAX + 1.
416  */
417 #define DT_ACT_BASE             DIF_SUBR_MAX + 1
418 #define DT_ACT(n)               (DT_ACT_BASE + (n))

420 #define DT_ACT_PRINTF           DT_ACT(0)       /* printf() action */
421 #define DT_ACT_TRACE            DT_ACT(1)       /* trace() action */
422 #define DT_ACT_TRACEMEM         DT_ACT(2)       /* tracemem() action */
423 #define DT_ACT_STACK            DT_ACT(3)       /* stack() action */
424 #define DT_ACT_STOP             DT_ACT(4)       /* stop() action */
425 #define DT_ACT_BREAKPOINT       DT_ACT(5)       /* breakpoint() action */
426 #define DT_ACT_PANIC            DT_ACT(6)       /* panic() action */
427 #define DT_ACT_SPECULATE        DT_ACT(7)       /* speculate() action */
428 #define DT_ACT_COMMIT           DT_ACT(8)       /* commit() action */
429 #define DT_ACT_DISCARD          DT_ACT(9)       /* discard() action */
430 #define DT_ACT_CHILL            DT_ACT(10)      /* chill() action */
431 #define DT_ACT_EXIT             DT_ACT(11)      /* exit() action */
432 #define DT_ACT_USTACK           DT_ACT(12)      /* ustack() action */
```

```
433 #define DT_ACT_PRINTA           DT_ACT(13)      /* printa() action */
434 #define DT_ACT_RAISE            DT_ACT(14)      /* raise() action */
435 #define DT_ACT_CLEAR            DT_ACT(15)      /* clear() action */
436 #define DT_ACT_NORMALIZE        DT_ACT(16)      /* normalize() action */
437 #define DT_ACT_DENORMALIZE      DT_ACT(17)      /* denormalize() action */
438 #define DT_ACT_TRUNC            DT_ACT(18)      /* trunc() action */
439 #define DT_ACT_SYSTEM           DT_ACT(19)      /* system() action */
440 #define DT_ACT_JSTACK           DT_ACT(20)      /* jstack() action */
441 #define DT_ACT_FTRUNCATE        DT_ACT(21)      /* ftruncate() action */
442 #define DT_ACT_FREOPEN          DT_ACT(22)      /* freopen() action */
443 #define DT_ACT_SYM              DT_ACT(23)      /* sym()/func() actions */
444 #define DT_ACT_MOD              DT_ACT(24)      /* mod() action */
445 #define DT_ACT_USYM             DT_ACT(25)      /* usym()/ufunc() actions */
446 #define DT_ACT_UMOD             DT_ACT(26)      /* umod() action */
447 #define DT_ACT_UADDR            DT_ACT(27)      /* uaddr() action */
448 #define DT_ACT_SETOPT           DT_ACT(28)      /* setopt() action */
449 #define DT_ACT_PRINT            DT_ACT(29)      /* print() action */

451 /*
452  * Sentinel to tell freopen() to restore the saved stdout.  This must not
453  * be ever valid for opening for write access via freopen(3C), which of
454  * course, "." never is.
455  */
456 #define DT_FREOPEN_RESTORE      "."

458 #define EDT_BASE        1000    /* base value for libdtrace errnos */

460 enum {
461         EDT_VERSION = EDT_BASE, /* client is requesting unsupported version */
462         EDT_VERSINVAL,          /* version string is invalid or overflows */
463         EDT_VERSUNDEF,          /* requested API version is not defined */
464         EDT_VERSREDUCED,        /* requested API version has been reduced */
465         EDT_CTF,                /* libctf called failed (dt_ctferr has more) */
466         EDT_COMPILER,           /* error in D program compilation */
467         EDT_NOTUPREG,           /* tuple register allocation failure */
468         EDT_NOMEM,              /* memory allocation failure */
469         EDT_INT2BIG,            /* integer limit exceeded */
470         EDT_STR2BIG,            /* string limit exceeded */
471         EDT_NOMOD,              /* unknown module name */
472         EDT_NOPROV,             /* unknown provider name */
473         EDT_NOPROBE,            /* unknown probe name */
474         EDT_NOSYM,              /* unknown symbol name */
475         EDT_NOSYMADDR,          /* no symbol corresponds to address */
476         EDT_NOTYPE,             /* unknown type name */
477         EDT_NOVAR,              /* unknown variable name */
478         EDT_NOAGG,              /* unknown aggregation name */
479         EDT_BADSCOPE,           /* improper use of type name scoping operator */
480         EDT_BADSPEC,            /* overspecified probe description */
481         EDT_BADSPCV,            /* bad macro variable in probe description */
482         EDT_BADID,              /* invalid probe identifier */
483         EDT_NOTLOADED,          /* module is not currently loaded */
484         EDT_NOCTF,              /* module does not contain any CTF data */
485         EDT_DATAMODEL,          /* module and program data models don't match */
486         EDT_DIFVERS,            /* library has newer DIF version than driver */
487         EDT_BADAGG,             /* unrecognized aggregating action */
488         EDT_FIO,                /* file i/o error */
489         EDT_DIFINVAL,           /* invalid DIF program */
490         EDT_DIFSIZE,            /* invalid DIF size */
491         EDT_DIFFAULT,           /* failed to copyin DIF program */
492         EDT_BADPROBE,           /* bad probe description */
493         EDT_BADPGLOB,           /* bad probe description globbing pattern */
494         EDT_NOSCOPE,            /* declaration scope stack underflow */
495         EDT_NODECL,             /* declaration stack underflow */
496         EDT_DMISMATCH,          /* record list does not match statement */
497         EDT_DOFFSET,            /* record data offset error */
498         EDT_DALIGN,             /* record data alignment error */
```



```
499         EDT_BADOPTNAME,         /* invalid dtrace_setopt option name */
500         EDT_BADOPTVAL,          /* invalid dtrace_setopt option value */
501         EDT_BADOPTCTX,          /* invalid dtrace_setopt option context */
502         EDT_CPPFORK,            /* failed to fork preprocessor */
503         EDT_CPPEXEC,            /* failed to exec preprocessor */
504         EDT_CPPENT,             /* preprocessor not found */
505         EDT_CPPERR,             /* unknown preprocessor error */
506         EDT_SYMOFLOW,           /* external symbol table overflow */
507         EDT_ACTIVE,             /* operation illegal when tracing is active */
508         EDT_DESTRUCTIVE,        /* destructive actions not allowed */
509         EDT_NOANON,             /* no anonymous tracing state */
510         EDT_ISANON,             /* can't claim anon state and enable probes */
511         EDT_ENDTOOBIG,          /* END enablings exceed size of prncpl buffer */
512         EDT_NOCONV,             /* failed to load type for printf conversion */
513         EDT_BADCONV,            /* incomplete printf conversion */
514         EDT_BADERROR,           /* invalid library ERROR action */
515         EDT_ERRABORT,           /* abort due to error */
516         EDT_DROPABORT,          /* abort due to drop */
517         EDT_DIRABORT,           /* abort explicitly directed */
518         EDT_BADRVAL,            /* invalid return value from callback */
519         EDT_BADNORMAL,          /* invalid normalization */
520         EDT_BUFTOOSMALL,        /* enabling exceeds size of buffer */
521         EDT_BADTRUNC,           /* invalid truncation */
522         EDT_BUSY,               /* device busy (active kernel debugger) */
523         EDT_ACCESS,             /* insufficient privileges to use DTrace */
524         EDT_NOENT,              /* dtrace device not available */
525         EDT_BRICKED,            /* abort due to systemic unresponsiveness */
526         EDT_HARDWIRE,           /* failed to load hard-wired definitions */
527         EDT_ELFVERSION,         /* libelf is out-of-date w.r.t libdtrace */
528         EDT_NOBUFFERED,         /* attempt to buffer output without handler */
529         EDT_UNSTABLE,           /* description matched unstable set of probes */
530         EDT_BADSETOPT,          /* invalid setopt library action */
531         EDT_BADSTACKPC,         /* invalid stack program counter size */
532         EDT_BADAGGVAR,          /* invalid aggregation variable identifier */
533         EDT_OVERSION,           /* client is requesting deprecated version */
534         EDT_ENABLING_ERR,       /* failed to enable probe */
535         EDT_NOPROBES,           /* no probes sites for declared provider */
536         EDT_CANTLOAD            /* failed to load a module */
537 };

539 /*
540  * Interfaces for parsing and comparing DTrace attribute tuples, which describe
541  * stability and architectural binding information.  The dtrace_attribute_t
542  * structure and associated constant definitions are found in <sys/dtrace.h>.
543  */
544 extern dtrace_attribute_t dt_attr_min(dtrace_attribute_t, dtrace_attribute_t);
545 extern dtrace_attribute_t dt_attr_max(dtrace_attribute_t, dtrace_attribute_t);
546 extern char *dt_attr_str(dtrace_attribute_t, char *, size_t);
547 extern int dt_attr_cmp(dtrace_attribute_t, dtrace_attribute_t);

549 /*
550  * Interfaces for parsing and handling DTrace version strings.  Version binding
551  * is a feature of the D compiler that is handled completely independently of
552  * the DTrace kernel infrastructure, so the definitions are here in libdtrace.
553  * Version strings are compiled into an encoded uint32_t which can be compared
554  * using C comparison operators.  Version definitions are found in dt_open.c.
555  */
556 #define DT_VERSION_STRMAX       16      /* enough for "255.4095.4095\0" */
557 #define DT_VERSION_MAJMAX       0xFF    /* maximum major version number */
558 #define DT_VERSION_MINMAX       0xFFF   /* maximum minor version number */
559 #define DT_VERSION_MICMAX       0xFFF   /* maximum micro version number */

561 #define DT_VERSION_NUMBER(M, m, u) \
562         ((((M) & 0xFF) << 24) | (((m) & 0xFFF) << 12) | ((u) & 0xFFF))

564 #define DT_VERSION_MAJOR(v)     (((v) & 0xFF000000) >> 24)
```

```
565 #define DT_VERSION_MINOR(v)     (((v) & 0x00FFF000) >> 12)
566 #define DT_VERSION_MICRO(v)     ((v) & 0x00000FFF)

568 extern char *dt_version_num2str(dt_version_t, char *, size_t);
569 extern int dt_version_str2num(const char *, dt_version_t *);
570 extern int dt_version_defined(dt_version_t);

572 /*
573  * Miscellaneous internal libdtrace interfaces.  The definitions below are for
574  * libdtrace routines that do not yet merit their own separate header file.
575  */
576 extern char *dt_cpp_add_arg(dtrace_hdl_t *, const char *);
577 extern char *dt_cpp_pop_arg(dtrace_hdl_t *);

579 extern int dt_set_errno(dtrace_hdl_t *, int);
580 extern void dt_set_errmsg(dtrace_hdl_t *, const char *, const char *,
581     const char *, int, const char *, va_list);

583 extern int dt_ioctl(dtrace_hdl_t *, int, void *);
584 extern int dt_status(dtrace_hdl_t *, processorid_t);
585 extern long dt_sysconf(dtrace_hdl_t *, int);
586 extern ssize_t dt_write(dtrace_hdl_t *, int, const void *, size_t);
587 extern int dt_printf(dtrace_hdl_t *, FILE *, const char *, ...);

589 extern void *dt_zalloc(dtrace_hdl_t *, size_t);
590 extern void *dt_alloc(dtrace_hdl_t *, size_t);
591 extern void dt_free(dtrace_hdl_t *, void *);
592 extern void dt_difo_free(dtrace_hdl_t *, dtrace_difo_t *);

594 extern int dt_gmatch(const char *, const char *);
595 extern char *dt_basename(char *);

597 extern ulong_t dt_popc(ulong_t);
598 extern ulong_t dt_popcb(const ulong_t *, ulong_t);

600 extern int dt_buffered_enable(dtrace_hdl_t *);
601 extern int dt_buffered_flush(dtrace_hdl_t *, dtrace_probedata_t *,
602     const dtrace_recdesc_t *, const dtrace_aggdata_t *, uint32_t flags);
603 extern void dt_buffered_disable(dtrace_hdl_t *);
604 extern void dt_buffered_destroy(dtrace_hdl_t *);

606 extern uint64_t dt_stddev(uint64_t *, uint64_t);

608 extern int dt_options_load(dtrace_hdl_t *);

610 extern void dt_dprintf(const char *, ...);

612 extern void dt_setcontext(dtrace_hdl_t *, dtrace_probedesc_t *);
613 extern void dt_endcontext(dtrace_hdl_t *);

615 extern void dt_pragma(dt_node_t *);
616 extern int dt_reduce(dtrace_hdl_t *, dt_version_t);
617 extern void dt_cg(dt_pcb_t *, dt_node_t *);
618 extern dtrace_difo_t *dt_as(dt_pcb_t *);
619 extern void dt_dis(const dtrace_difo_t *, FILE *);

621 extern int dt_aggregate_go(dtrace_hdl_t *);
622 extern int dt_aggregate_init(dtrace_hdl_t *);
623 extern void dt_aggregate_destroy(dtrace_hdl_t *);

625 extern int dt_epid_lookup(dtrace_hdl_t *, dtrace_epid_t,
626     dtrace_eprobedesc_t **, dtrace_probedesc_t **);
627 extern void dt_epid_destroy(dtrace_hdl_t *);
628 extern int dt_aggid_lookup(dtrace_hdl_t *, dtrace_aggid_t, dtrace_aggdesc_t **);
629 extern void dt_aggid_destroy(dtrace_hdl_t *);
```



```
631 extern void *dt_format_lookup(dtrace_hdl_t *, int);
632 extern void dt_format_destroy(dtrace_hdl_t *);

634 extern const char *dt_strdata_lookup(dtrace_hdl_t *, int);
635 extern void dt_strdata_destroy(dtrace_hdl_t *);

637 extern int dt_print_quantize(dtrace_hdl_t *, FILE *,
638     const void *, size_t, uint64_t);
639 extern int dt_print_lquantize(dtrace_hdl_t *, FILE *,
640     const void *, size_t, uint64_t);
641 extern int dt_print_llquantize(dtrace_hdl_t *, FILE *,
642     const void *, size_t, uint64_t);
643 extern int dt_print_agg(const dtrace_aggdata_t *, void *);

645 extern int dt_handle(dtrace_hdl_t *, dtrace_probedata_t *);
646 extern int dt_handle_liberr(dtrace_hdl_t *,
647     const dtrace_probedata_t *, const char *);
648 extern int dt_handle_cpudrop(dtrace_hdl_t *, processorid_t,
649     dtrace_dropkind_t, uint64_t);
650 extern int dt_handle_status(dtrace_hdl_t *,
651     dtrace_status_t *, dtrace_status_t *);
652 extern int dt_handle_setopt(dtrace_hdl_t *, dtrace_setoptdata_t *);

654 extern int dt_lib_depend_add(dtrace_hdl_t *, dt_list_t *, const char *);
655 extern dt_lib_depend_t *dt_lib_depend_lookup(dt_list_t *, const char *);

657 extern dt_pcb_t *yypcb;         /* pointer to current parser control block */
658 extern char yyintprefix;        /* int token prefix for macros (+/-) */
659 extern char yyintsuffix[4];     /* int token suffix ([uUlL]*) */
660 extern int yyintdecimal;        /* int token is decimal (1) or octal/hex (0) */
661 extern char yytext[];           /* lex input buffer */
662 extern int yylineno;            /* lex line number */
663 extern int yydebug;             /* lex debugging */
664 extern dt_node_t *yypragma;     /* lex token list for control lines */

666 extern const dtrace_attribute_t _dtrace_maxattr; /* maximum attributes */
667 extern const dtrace_attribute_t _dtrace_defattr; /* default attributes */
668 extern const dtrace_attribute_t _dtrace_symattr; /* symbol ref attributes */
669 extern const dtrace_attribute_t _dtrace_typattr; /* type ref attributes */
670 extern const dtrace_attribute_t _dtrace_prvattr; /* provider attributes */
671 extern const dtrace_pattr_t _dtrace_prvdesc;     /* provider attribute bundle */

673 extern const dt_version_t _dtrace_versions[];    /* array of valid versions */
674 extern const char *const _dtrace_version;        /* current version string */

676 extern int _dtrace_strbuckets;          /* number of hash buckets for strings */
677 extern int _dtrace_intbuckets;          /* number of hash buckets for ints */
678 extern uint_t _dtrace_stkindent;        /* default indent for stack/ustack */
679 extern uint_t _dtrace_pidbuckets;       /* number of hash buckets for pids */
680 extern uint_t _dtrace_pidlrulim;        /* number of proc handles to cache */
681 extern int _dtrace_debug;               /* debugging messages enabled */
682 extern size_t _dtrace_bufsize;          /* default dt_buf_create() size */
683 extern int _dtrace_argmax;              /* default maximum probe arguments */

685 extern const char *_dtrace_libdir;      /* default library directory */
686 extern const char *_dtrace_moddir;      /* default kernel module directory */

688 #ifdef  __cplusplus
689 }
690 #endif

692 #endif  /* _DT_IMPL_H */
```

```
**********************************************************
   54383 Tue Jan 14 20:13:05 2014
new/usr/src/lib/libdtrace/common/dt_open.c
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
______unchanged_portion_omitted_

  82 /*
  83  * The version number should be increased for every customer visible release
  84  * of DTrace. The major number should be incremented when a fundamental
  85  * change has been made that would affect all consumers, and would reflect
  86  * sweeping changes to DTrace or the D language. The minor number should be
  87  * incremented when a change is introduced that could break scripts that had
  88  * previously worked; for example, adding a new built-in variable could break
  89  * a script which was already using that identifier. The micro number should
  90  * be changed when introducing functionality changes or major bug fixes that
  91  * do not affect backward compatibility -- this is merely to make capabilities
  92  * easily determined from the version number. Minor bugs do not require any
  93  * modification to the version number.
  94  */
  95 #define DT_VERS_1_0     DT_VERSION_NUMBER(1, 0, 0)
  96 #define DT_VERS_1_1     DT_VERSION_NUMBER(1, 1, 0)
  97 #define DT_VERS_1_2     DT_VERSION_NUMBER(1, 2, 0)
  98 #define DT_VERS_1_2_1   DT_VERSION_NUMBER(1, 2, 1)
  99 #define DT_VERS_1_2_2   DT_VERSION_NUMBER(1, 2, 2)
 100 #define DT_VERS_1_3     DT_VERSION_NUMBER(1, 3, 0)
 101 #define DT_VERS_1_4     DT_VERSION_NUMBER(1, 4, 0)
 102 #define DT_VERS_1_4_1   DT_VERSION_NUMBER(1, 4, 1)
 103 #define DT_VERS_1_5     DT_VERSION_NUMBER(1, 5, 0)
 104 #define DT_VERS_1_6     DT_VERSION_NUMBER(1, 6, 0)
 105 #define DT_VERS_1_6_1   DT_VERSION_NUMBER(1, 6, 1)
 106 #define DT_VERS_1_6_2   DT_VERSION_NUMBER(1, 6, 2)
 107 #define DT_VERS_1_6_3   DT_VERSION_NUMBER(1, 6, 3)
 108 #define DT_VERS_1_7     DT_VERSION_NUMBER(1, 7, 0)
 109 #define DT_VERS_1_7_1   DT_VERSION_NUMBER(1, 7, 1)
 110 #define DT_VERS_1_8     DT_VERSION_NUMBER(1, 8, 0)
 111 #define DT_VERS_1_8_1   DT_VERSION_NUMBER(1, 8, 1)
 112 #define DT_VERS_1_9     DT_VERSION_NUMBER(1, 9, 0)
 113 #define DT_VERS_1_9_1   DT_VERSION_NUMBER(1, 9, 1)
 114 #define DT_VERS_1_10    DT_VERSION_NUMBER(1, 10, 0)
 115 #define DT_VERS_1_11    DT_VERSION_NUMBER(1, 11, 0)
 116 #define DT_VERS_1_12    DT_VERSION_NUMBER(1, 12, 0)
 117 #define DT_VERS_1_12_1  DT_VERSION_NUMBER(1, 12, 1)
 118 #define DT_VERS_LATEST  DT_VERS_1_12_1
 119 #define DT_VERS_STRING  "Sun D 1.12.1"
 117 #define DT_VERS_LATEST  DT_VERS_1_12
 118 #define DT_VERS_STRING  "Sun D 1.12"

 121 const dt_version_t _dtrace_versions[] = {
 122         DT_VERS_1_0,    /* D API 1.0.0 (PSARC 2001/466) Solaris 10 FCS */
 123         DT_VERS_1_1,    /* D API 1.1.0 Solaris Express 6/05 */
 124         DT_VERS_1_2,    /* D API 1.2.0 Solaris 10 Update 1 */
 125         DT_VERS_1_2_1,  /* D API 1.2.1 Solaris Express 4/06 */
 126         DT_VERS_1_2_2,  /* D API 1.2.2 Solaris Express 6/06 */
 127         DT_VERS_1_3,    /* D API 1.3 Solaris Express 10/06 */
 128         DT_VERS_1_4,    /* D API 1.4 Solaris Express 2/07 */
 129         DT_VERS_1_4_1,  /* D API 1.4.1 Solaris Express 4/07 */
 130         DT_VERS_1_5,    /* D API 1.5 Solaris Express 7/07 */
 131         DT_VERS_1_6,    /* D API 1.6 */
 132         DT_VERS_1_6_1,  /* D API 1.6.1 */
 133         DT_VERS_1_6_2,  /* D API 1.6.2 */
 134         DT_VERS_1_6_3,  /* D API 1.6.3 */
 135         DT_VERS_1_7,    /* D API 1.7 */
 136         DT_VERS_1_7_1,  /* D API 1.7.1 */
```



```
 137         DT_VERS_1_8,    /* D API 1.8 */
 138         DT_VERS_1_8_1,  /* D API 1.8.1 */
 139         DT_VERS_1_9,    /* D API 1.9 */
 140         DT_VERS_1_9_1,  /* D API 1.9.1 */
 141         DT_VERS_1_10,   /* D API 1.10 */
 142         DT_VERS_1_11,   /* D API 1.11 */
 143         DT_VERS_1_12,   /* D API 1.12 */
 144         DT_VERS_1_12_1, /* D API 1.12.1 */
 145 #endif /* ! codereview */
 146         0
 147 };

 149 /*
 150  * Table of global identifiers.  This is used to populate the global identifier
 151  * hash when a new dtrace client open occurs.  For more info see dt_ident.h.
 152  * The global identifiers that represent functions use the dt_idops_func ops
 153  * and specify the private data pointer as a prototype string which is parsed
 154  * when the identifier is first encountered.  These prototypes look like ANSI
 155  * C function prototypes except that the special symbol "@" can be used as a
 156  * wildcard to represent a single parameter of any type (i.e. any dt_node_t).
 157  * The standard "..." notation can also be used to represent varargs.  An empty
 158  * parameter list is taken to mean void (that is, no arguments are permitted).
 159  * A parameter enclosed in square brackets (e.g. "[int]") denotes an optional
 160  * argument.
 161  */
 162 static const dt_ident_t _dtrace_globals[] = {
 163 { "alloca", DT_IDENT_FUNC, 0, DIF_SUBR_ALLOCA, DT_ATTR_STABCMN, DT_VERS_1_0,
 164         &dt_idops_func, "void *(size_t)" },
 165 { "arg0", DT_IDENT_SCALAR, 0, DIF_VAR_ARG0, DT_ATTR_STABCMN, DT_VERS_1_0,
 166         &dt_idops_type, "int64_t" },
 167 { "arg1", DT_IDENT_SCALAR, 0, DIF_VAR_ARG1, DT_ATTR_STABCMN, DT_VERS_1_0,
 168         &dt_idops_type, "int64_t" },
 169 { "arg2", DT_IDENT_SCALAR, 0, DIF_VAR_ARG2, DT_ATTR_STABCMN, DT_VERS_1_0,
 170         &dt_idops_type, "int64_t" },
 171 { "arg3", DT_IDENT_SCALAR, 0, DIF_VAR_ARG3, DT_ATTR_STABCMN, DT_VERS_1_0,
 172         &dt_idops_type, "int64_t" },
 173 { "arg4", DT_IDENT_SCALAR, 0, DIF_VAR_ARG4, DT_ATTR_STABCMN, DT_VERS_1_0,
 174         &dt_idops_type, "int64_t" },
 175 { "arg5", DT_IDENT_SCALAR, 0, DIF_VAR_ARG5, DT_ATTR_STABCMN, DT_VERS_1_0,
 176         &dt_idops_type, "int64_t" },
 177 { "arg6", DT_IDENT_SCALAR, 0, DIF_VAR_ARG6, DT_ATTR_STABCMN, DT_VERS_1_0,
 178         &dt_idops_type, "int64_t" },
 179 { "arg7", DT_IDENT_SCALAR, 0, DIF_VAR_ARG7, DT_ATTR_STABCMN, DT_VERS_1_0,
 180         &dt_idops_type, "int64_t" },
 181 { "arg8", DT_IDENT_SCALAR, 0, DIF_VAR_ARG8, DT_ATTR_STABCMN, DT_VERS_1_0,
 182         &dt_idops_type, "int64_t" },
 183 { "arg9", DT_IDENT_SCALAR, 0, DIF_VAR_ARG9, DT_ATTR_STABCMN, DT_VERS_1_0,
 184         &dt_idops_type, "int64_t" },
 185 { "args", DT_IDENT_ARRAY, 0, DIF_VAR_ARGS, DT_ATTR_STABCMN, DT_VERS_1_0,
 186         &dt_idops_args, NULL },
 187 { "avg", DT_IDENT_AGGFUNC, 0, DTRACEAGG_AVG, DT_ATTR_STABCMN, DT_VERS_1_0,
 188         &dt_idops_func, "void(@)" },
 189 { "basename", DT_IDENT_FUNC, 0, DIF_SUBR_BASENAME, DT_ATTR_STABCMN, DT_VERS_1_0,
 190         &dt_idops_func, "string(const char *)" },
 191 { "bcopy", DT_IDENT_FUNC, 0, DIF_SUBR_BCOPY, DT_ATTR_STABCMN, DT_VERS_1_0,
 192         &dt_idops_func, "void(void *, void *, size_t)" },
 193 { "breakpoint", DT_IDENT_ACTFUNC, 0, DT_ACT_BREAKPOINT,
 194         DT_ATTR_STABCMN, DT_VERS_1_0,
 195         &dt_idops_func, "void()" },
 196 { "caller", DT_IDENT_SCALAR, 0, DIF_VAR_CALLER, DT_ATTR_STABCMN, DT_VERS_1_0,
 197         &dt_idops_type, "uintptr_t" },
 198 { "chill", DT_IDENT_ACTFUNC, 0, DT_ACT_CHILL, DT_ATTR_STABCMN, DT_VERS_1_0,
 199         &dt_idops_func, "void(int)" },
 200 { "cleanpath", DT_IDENT_FUNC, 0, DIF_SUBR_CLEANPATH, DT_ATTR_STABCMN,
 201         DT_VERS_1_0, &dt_idops_func, "string(const char *)" },
 202 { "clear", DT_IDENT_ACTFUNC, 0, DT_ACT_CLEAR, DT_ATTR_STABCMN, DT_VERS_1_0,
```

```
203         &dt_idops_func, "void(...)" },
204 { "commit", DT_IDENT_ACTFUNC, 0, DT_ACT_COMMIT, DT_ATTR_STABCMN, DT_VERS_1_0,
205         &dt_idops_func, "void(int)" },
206 { "copyin", DT_IDENT_FUNC, 0, DIF_SUBR_COPYIN, DT_ATTR_STABCMN, DT_VERS_1_0,
207         &dt_idops_func, "void *(uintptr_t, size_t)" },
208 { "copyinstr", DT_IDENT_FUNC, 0, DIF_SUBR_COPYINSTR,
209         DT_ATTR_STABCMN, DT_VERS_1_0,
210         &dt_idops_func, "string(uintptr_t, [size_t])" },
211 { "copyinto", DT_IDENT_FUNC, 0, DIF_SUBR_COPYINTO, DT_ATTR_STABCMN,
212         DT_VERS_1_0, &dt_idops_func, "void(uintptr_t, size_t, void *)" },
213 { "copyout", DT_IDENT_FUNC, 0, DIF_SUBR_COPYOUT, DT_ATTR_STABCMN, DT_VERS_1_0,
214         &dt_idops_func, "void(void *, uintptr_t, size_t)" },
215 { "copyoutstr", DT_IDENT_FUNC, 0, DIF_SUBR_COPYOUTSTR,
216         DT_ATTR_STABCMN, DT_VERS_1_0,
217         &dt_idops_func, "void(char *, uintptr_t, size_t)" },
218 { "count", DT_IDENT_AGGFUNC, 0, DTRACEAGG_COUNT, DT_ATTR_STABCMN, DT_VERS_1_0,
219         &dt_idops_func, "void()" },
220 { "curthread", DT_IDENT_SCALAR, 0, DIF_VAR_CURTHREAD,
221         { DTRACE_STABILITY_STABLE, DTRACE_STABILITY_PRIVATE,
222         DTRACE_CLASS_COMMON }, DT_VERS_1_0,
223         &dt_idops_type, "genunix`kthread_t *" },
224 { "ddi_pathname", DT_IDENT_FUNC, 0, DIF_SUBR_DDI_PATHNAME,
225         DT_ATTR_EVOLCMN, DT_VERS_1_0,
226         &dt_idops_func, "string(void *, int64_t)" },
227 { "denormalize", DT_IDENT_ACTFUNC, 0, DT_ACT_DENORMALIZE, DT_ATTR_STABCMN,
228         DT_VERS_1_0, &dt_idops_func, "void(...)" },
229 { "dirname", DT_IDENT_FUNC, 0, DIF_SUBR_DIRNAME, DT_ATTR_STABCMN, DT_VERS_1_0,
230         &dt_idops_func, "string(const char *)" },
231 { "discard", DT_IDENT_ACTFUNC, 0, DT_ACT_DISCARD, DT_ATTR_STABCMN, DT_VERS_1_0,
232         &dt_idops_func, "void(int)" },
233 { "epid", DT_IDENT_SCALAR, 0, DIF_VAR_EPID, DT_ATTR_STABCMN, DT_VERS_1_0,
234         &dt_idops_type, "uint_t" },
235 { "errno", DT_IDENT_SCALAR, 0, DIF_VAR_ERRNO, DT_ATTR_STABCMN, DT_VERS_1_0,
236         &dt_idops_type, "int" },
237 { "execname", DT_IDENT_SCALAR, 0, DIF_VAR_EXECNAME,
238         DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
239 { "exit", DT_IDENT_ACTFUNC, 0, DT_ACT_EXIT, DT_ATTR_STABCMN, DT_VERS_1_0,
240         &dt_idops_func, "void(int)" },
241 { "freopen", DT_IDENT_ACTFUNC, 0, DT_ACT_FREOPEN, DT_ATTR_STABCMN,
242         DT_VERS_1_1, &dt_idops_func, "void(@, ...)" },
243 { "ftruncate", DT_IDENT_ACTFUNC, 0, DT_ACT_FTRUNCATE, DT_ATTR_STABCMN,
244         DT_VERS_1_0, &dt_idops_func, "void()" },
245 { "func", DT_IDENT_ACTFUNC, 0, DT_ACT_SYM, DT_ATTR_STABCMN,
246         DT_VERS_1_2, &dt_idops_func, "_symaddr(uintptr_t)" },
247 { "getmajor", DT_IDENT_FUNC, 0, DIF_SUBR_GETMAJOR,
248         DT_ATTR_EVOLCMN, DT_VERS_1_0,
249         &dt_idops_func, "genunix`major_t(genunix`dev_t)" },
250 { "getminor", DT_IDENT_FUNC, 0, DIF_SUBR_GETMINOR,
251         DT_ATTR_EVOLCMN, DT_VERS_1_0,
252         &dt_idops_func, "genunix`minor_t(genunix`dev_t)" },
253 { "htonl", DT_IDENT_FUNC, 0, DIF_SUBR_HTONL, DT_ATTR_EVOLCMN, DT_VERS_1_3,
254         &dt_idops_func, "uint32_t(uint32_t)" },
255 { "htonll", DT_IDENT_FUNC, 0, DIF_SUBR_HTONLL, DT_ATTR_EVOLCMN, DT_VERS_1_3,
256         &dt_idops_func, "uint64_t(uint64_t)" },
257 { "htons", DT_IDENT_FUNC, 0, DIF_SUBR_HTONS, DT_ATTR_EVOLCMN, DT_VERS_1_3,
258         &dt_idops_func, "uint16_t(uint16_t)" },
259 { "getf", DT_IDENT_FUNC, 0, DIF_SUBR_GETF, DT_ATTR_STABCMN, DT_VERS_1_10,
260         &dt_idops_func, "file_t *(int)" },
261 { "gid", DT_IDENT_SCALAR, 0, DIF_VAR_GID, DT_ATTR_STABCMN, DT_VERS_1_0,
262         &dt_idops_type, "gid_t" },
263 { "id", DT_IDENT_SCALAR, 0, DIF_VAR_ID, DT_ATTR_STABCMN, DT_VERS_1_0,
264         &dt_idops_type, "uint_t" },
265 { "index", DT_IDENT_FUNC, 0, DIF_SUBR_INDEX, DT_ATTR_STABCMN, DT_VERS_1_1,
266         &dt_idops_func, "int(const char *, const char *, [int])" },
267 { "inet_ntoa", DT_IDENT_FUNC, 0, DIF_SUBR_INET_NTOA, DT_ATTR_STABCMN,
268         DT_VERS_1_5, &dt_idops_func, "string(ipaddr_t *)" },
```



```
269 { "inet_ntoa6", DT_IDENT_FUNC, 0, DIF_SUBR_INET_NTOA6, DT_ATTR_STABCMN,
270         DT_VERS_1_5, &dt_idops_func, "string(in6_addr_t *)" },
271 { "inet_ntop", DT_IDENT_FUNC, 0, DIF_SUBR_INET_NTOP, DT_ATTR_STABCMN,
272         DT_VERS_1_5, &dt_idops_func, "string(int, void *)" },
273 { "ipl", DT_IDENT_SCALAR, 0, DIF_VAR_IPL, DT_ATTR_STABCMN, DT_VERS_1_0,
274         &dt_idops_type, "uint_t" },
275 { "json", DT_IDENT_FUNC, 0, DIF_SUBR_JSON, DT_ATTR_STABCMN, DT_VERS_1_11,
276         &dt_idops_func, "string(const char *, const char *)" },
277 { "jstack", DT_IDENT_ACTFUNC, 0, DT_ACT_JSTACK, DT_ATTR_STABCMN, DT_VERS_1_0,
278         &dt_idops_func, "stack(...)" },
279 { "lltostr", DT_IDENT_FUNC, 0, DIF_SUBR_LLTOSTR, DT_ATTR_STABCMN, DT_VERS_1_0,
280         &dt_idops_func, "string(int64_t, [int])" },
281 { "llquantize", DT_IDENT_AGGFUNC, 0, DTRACEAGG_LLQUANTIZE, DT_ATTR_STABCMN,
282         DT_VERS_1_7, &dt_idops_func,
283         "void(@, int32_t, int32_t, int32_t, int32_t, ...)" },
284 { "lquantize", DT_IDENT_AGGFUNC, 0, DTRACEAGG_LQUANTIZE,
285         DT_ATTR_STABCMN, DT_VERS_1_0,
286         &dt_idops_func, "void(@, int32_t, int32_t, ...)" },
287 { "max", DT_IDENT_AGGFUNC, 0, DTRACEAGG_MAX, DT_ATTR_STABCMN, DT_VERS_1_0,
288         &dt_idops_func, "void(@)" },
289 { "min", DT_IDENT_AGGFUNC, 0, DTRACEAGG_MIN, DT_ATTR_STABCMN, DT_VERS_1_0,
290         &dt_idops_func, "void(@)" },
291 { "mod", DT_IDENT_ACTFUNC, 0, DT_ACT_MOD, DT_ATTR_STABCMN,
292         DT_VERS_1_2, &dt_idops_func, "_symaddr(uintptr_t)" },
293 { "msgdsize", DT_IDENT_FUNC, 0, DIF_SUBR_MSGDSIZE,
294         DT_ATTR_STABCMN, DT_VERS_1_0,
295         &dt_idops_func, "size_t(mblk_t *)" },
296 { "msgsize", DT_IDENT_FUNC, 0, DIF_SUBR_MSGSIZE,
297         DT_ATTR_STABCMN, DT_VERS_1_0,
298         &dt_idops_func, "size_t(mblk_t *)" },
299 { "mutex_owned", DT_IDENT_FUNC, 0, DIF_SUBR_MUTEX_OWNED,
300         DT_ATTR_EVOLCMN, DT_VERS_1_0,
301         &dt_idops_func, "int(genunix`kmutex_t *)" },
302 { "mutex_owner", DT_IDENT_FUNC, 0, DIF_SUBR_MUTEX_OWNER,
303         DT_ATTR_EVOLCMN, DT_VERS_1_0,
304         &dt_idops_func, "genunix`kthread_t *(genunix`kmutex_t *)" },
305 { "mutex_type_adaptive", DT_IDENT_FUNC, 0, DIF_SUBR_MUTEX_TYPE_ADAPTIVE,
306         DT_ATTR_EVOLCMN, DT_VERS_1_0,
307         &dt_idops_func, "int(genunix`kmutex_t *)" },
308 { "mutex_type_spin", DT_IDENT_FUNC, 0, DIF_SUBR_MUTEX_TYPE_SPIN,
309         DT_ATTR_EVOLCMN, DT_VERS_1_0,
310         &dt_idops_func, "int(genunix`kmutex_t *)" },
311 { "ntohl", DT_IDENT_FUNC, 0, DIF_SUBR_NTOHL, DT_ATTR_EVOLCMN, DT_VERS_1_3,
312         &dt_idops_func, "uint32_t(uint32_t)" },
313 { "ntohll", DT_IDENT_FUNC, 0, DIF_SUBR_NTOHLL, DT_ATTR_EVOLCMN, DT_VERS_1_3,
314         &dt_idops_func, "uint64_t(uint64_t)" },
315 { "ntohs", DT_IDENT_FUNC, 0, DIF_SUBR_NTOHS, DT_ATTR_EVOLCMN, DT_VERS_1_3,
316         &dt_idops_func, "uint16_t(uint16_t)" },
317 { "normalize", DT_IDENT_ACTFUNC, 0, DT_ACT_NORMALIZE, DT_ATTR_STABCMN,
318         DT_VERS_1_0, &dt_idops_func, "void(...)" },
319 { "panic", DT_IDENT_ACTFUNC, 0, DT_ACT_PANIC, DT_ATTR_STABCMN, DT_VERS_1_0,
320         &dt_idops_func, "void()" },
321 { "pid", DT_IDENT_SCALAR, 0, DIF_VAR_PID, DT_ATTR_STABCMN, DT_VERS_1_0,
322         &dt_idops_type, "pid_t" },
323 { "ppid", DT_IDENT_SCALAR, 0, DIF_VAR_PPID, DT_ATTR_STABCMN, DT_VERS_1_0,
324         &dt_idops_type, "pid_t" },
325 { "print", DT_IDENT_ACTFUNC, 0, DT_ACT_PRINT, DT_ATTR_STABCMN, DT_VERS_1_9,
326         &dt_idops_func, "void(@)" },
327 { "printa", DT_IDENT_ACTFUNC, 0, DT_ACT_PRINTA, DT_ATTR_STABCMN, DT_VERS_1_0,
328         &dt_idops_func, "void(@, ...)" },
329 { "printf", DT_IDENT_ACTFUNC, 0, DT_ACT_PRINTF, DT_ATTR_STABCMN, DT_VERS_1_0,
330         &dt_idops_func, "void(@, ...)" },
331 { "probefunc", DT_IDENT_SCALAR, 0, DIF_VAR_PROBEFUNC,
332         DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
333 { "probemod", DT_IDENT_SCALAR, 0, DIF_VAR_PROBEMOD,
334         DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
```

```
{ "probename", DT_IDENT_SCALAR, 0, DIF_VAR_PROBENAME,
        DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
{ "probeprov", DT_IDENT_SCALAR, 0, DIF_VAR_PROBEPROV,
        DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
{ "progenyof", DT_IDENT_FUNC, 0, DIF_SUBR_PROGENYOF,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "int(pid_t)" },
{ "quantize", DT_IDENT_AGGFUNC, 0, DTRACEAGG_QUANTIZE,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@, ...)" },
{ "raise", DT_IDENT_ACTFUNC, 0, DT_ACT_RAISE, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(int)" },
{ "rand", DT_IDENT_FUNC, 0, DIF_SUBR_RAND, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "int()" },
{ "rindex", DT_IDENT_FUNC, 0, DIF_SUBR_RINDEX, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "int(const char *, const char *, [int])" },
{ "rw_iswriter", DT_IDENT_FUNC, 0, DIF_SUBR_RW_ISWRITER,
        DT_ATTR_EVOLCMN, DT_VERS_1_0,
        &dt_idops_func, "int(genunix`krwlock_t *)" },
{ "rw_read_held", DT_IDENT_FUNC, 0, DIF_SUBR_RW_READ_HELD,
        DT_ATTR_EVOLCMN, DT_VERS_1_0,
        &dt_idops_func, "int(genunix`krwlock_t *)" },
{ "rw_write_held", DT_IDENT_FUNC, 0, DIF_SUBR_RW_WRITE_HELD,
        DT_ATTR_EVOLCMN, DT_VERS_1_0,
        &dt_idops_func, "int(genunix`krwlock_t *)" },
{ "self", DT_IDENT_PTR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "void" },
{ "setopt", DT_IDENT_ACTFUNC, 0, DT_ACT_SETOPT, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "void(const char *, [const char *])" },
{ "speculate", DT_IDENT_ACTFUNC, 0, DT_ACT_SPECULATE,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(int)" },
{ "speculation", DT_IDENT_FUNC, 0, DIF_SUBR_SPECULATION,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "int()" },
{ "stack", DT_IDENT_ACTFUNC, 0, DT_ACT_STACK, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "stack(...)" },
{ "stackdepth", DT_IDENT_SCALAR, 0, DIF_VAR_STACKDEPTH,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "uint32_t" },
{ "stddev", DT_IDENT_AGGFUNC, 0, DTRACEAGG_STDDEV, DT_ATTR_STABCMN,
        DT_VERS_1_6, &dt_idops_func, "void(@)" },
{ "stop", DT_IDENT_ACTFUNC, 0, DT_ACT_STOP, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void()" },
{ "strchr", DT_IDENT_FUNC, 0, DIF_SUBR_STRCHR, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "string(const char *, char)" },
{ "strlen", DT_IDENT_FUNC, 0, DIF_SUBR_STRLEN, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "size_t(const char *)" },
{ "strjoin", DT_IDENT_FUNC, 0, DIF_SUBR_STRJOIN, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "string(const char *, const char *)" },
{ "strrchr", DT_IDENT_FUNC, 0, DIF_SUBR_STRRCHR, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "string(const char *, char)" },
{ "strstr", DT_IDENT_FUNC, 0, DIF_SUBR_STRSTR, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "string(const char *, const char *)" },
{ "strtok", DT_IDENT_FUNC, 0, DIF_SUBR_STRTOK, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "string(const char *, const char *)" },
{ "strtoll", DT_IDENT_FUNC, 0, DIF_SUBR_STRTOLL, DT_ATTR_STABCMN, DT_VERS_1_11,
        &dt_idops_func, "int64_t(const char *, [int])" },
{ "substr", DT_IDENT_FUNC, 0, DIF_SUBR_SUBSTR, DT_ATTR_STABCMN, DT_VERS_1_1,
        &dt_idops_func, "string(const char *, int, [int])" },
{ "sum", DT_IDENT_AGGFUNC, 0, DTRACEAGG_SUM, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@)" },
{ "sym", DT_IDENT_ACTFUNC, 0, DT_ACT_SYM, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "_symaddr(uintptr_t)" },
{ "system", DT_IDENT_ACTFUNC, 0, DT_ACT_SYSTEM, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@, ...)" },
```



```
{ "this", DT_IDENT_PTR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "void" },
{ "tid", DT_IDENT_SCALAR, 0, DIF_VAR_TID, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "id_t" },
{ "timestamp", DT_IDENT_SCALAR, 0, DIF_VAR_TIMESTAMP,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "uint64_t" },
{ "tolower", DT_IDENT_FUNC, 0, DIF_SUBR_TOLOWER, DT_ATTR_STABCMN, DT_VERS_1_8,
        &dt_idops_func, "string(const char *)" },
{ "toupper", DT_IDENT_FUNC, 0, DIF_SUBR_TOUPPER, DT_ATTR_STABCMN, DT_VERS_1_8,
        &dt_idops_func, "string(const char *)" },
{ "trace", DT_IDENT_ACTFUNC, 0, DT_ACT_TRACE, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@)" },
{ "tracemem", DT_IDENT_ACTFUNC, 0, DT_ACT_TRACEMEM,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@, size_t, ...)" },
{ "trunc", DT_IDENT_ACTFUNC, 0, DT_ACT_TRUNC, DT_ATTR_STABCMN,
        DT_VERS_1_0, &dt_idops_func, "void(...)" },
{ "uaddr", DT_IDENT_ACTFUNC, 0, DT_ACT_UADDR, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "_usymaddr(uintptr_t)" },
{ "ucaller", DT_IDENT_SCALAR, 0, DIF_VAR_UCALLER, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_type, "uint64_t" },
{ "ufunc", DT_IDENT_ACTFUNC, 0, DT_ACT_USYM, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "_usymaddr(uintptr_t)" },
{ "uid", DT_IDENT_SCALAR, 0, DIF_VAR_UID, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "uid_t" },
{ "umod", DT_IDENT_ACTFUNC, 0, DT_ACT_UMOD, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "_usymaddr(uintptr_t)" },
{ "uregs", DT_IDENT_ARRAY, 0, DIF_VAR_UREGS, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_regs, NULL },
{ "ustack", DT_IDENT_ACTFUNC, 0, DT_ACT_USTACK, DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "stack(...)" },
{ "ustackdepth", DT_IDENT_SCALAR, 0, DIF_VAR_USTACKDEPTH,
        DT_ATTR_STABCMN, DT_VERS_1_2,
        &dt_idops_type, "uint32_t" },
{ "usym", DT_IDENT_ACTFUNC, 0, DT_ACT_USYM, DT_ATTR_STABCMN,
        DT_VERS_1_2, &dt_idops_func, "_usymaddr(uintptr_t)" },
{ "vmregs", DT_IDENT_ARRAY, 0, DIF_VAR_VMREGS, DT_ATTR_STABCMN, DT_VERS_1_7,
        &dt_idops_regs, NULL },
{ "vtimestamp", DT_IDENT_SCALAR, 0, DIF_VAR_VTIMESTAMP,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "uint64_t" },
{ "walltimestamp", DT_IDENT_SCALAR, 0, DIF_VAR_WALLTIMESTAMP,
        DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_type, "int64_t" },
{ "zonename", DT_IDENT_SCALAR, 0, DIF_VAR_ZONENAME,
        DT_ATTR_STABCMN, DT_VERS_1_0, &dt_idops_type, "string" },
{ NULL, 0, 0, 0, { 0, 0, 0 }, 0, NULL, NULL }
};

/*
 * Tables of ILP32 intrinsic integer and floating-point type templates to use
 * to populate the dynamic "C" CTF type container.
 */
static const dt_intrinsic_t _dtrace_intrinsics_32[] = {
{ "void", { CTF_INT_SIGNED, 0, 0 }, CTF_K_INTEGER },
{ "signed", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "unsigned", { 0, 0, 32 }, CTF_K_INTEGER },
{ "char", { CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "short", { CTF_INT_SIGNED, 0, 16 }, CTF_K_INTEGER },
{ "int", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "long", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "long long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "signed char", { CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "signed short", { CTF_INT_SIGNED, 0, 16 }, CTF_K_INTEGER },
{ "signed int", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
```

```
{ "signed long", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "signed long long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "unsigned char", { CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "unsigned short", { 0, 0, 16 }, CTF_K_INTEGER },
{ "unsigned int", { 0, 0, 32 }, CTF_K_INTEGER },
{ "unsigned long", { 0, 0, 32 }, CTF_K_INTEGER },
{ "unsigned long long", { 0, 0, 64 }, CTF_K_INTEGER },
{ "_Bool", { CTF_INT_BOOL, 0, 8 }, CTF_K_INTEGER },
{ "float", { CTF_FP_SINGLE, 0, 32 }, CTF_K_FLOAT },
{ "double", { CTF_FP_DOUBLE, 0, 64 }, CTF_K_FLOAT },
{ "long double", { CTF_FP_LDOUBLE, 0, 128 }, CTF_K_FLOAT },
{ "float imaginary", { CTF_FP_IMAGRY, 0, 32 }, CTF_K_FLOAT },
{ "double imaginary", { CTF_FP_DIMAGRY, 0, 64 }, CTF_K_FLOAT },
{ "long double imaginary", { CTF_FP_LDIMAGRY, 0, 128 }, CTF_K_FLOAT },
{ "float complex", { CTF_FP_CPLX, 0, 64 }, CTF_K_FLOAT },
{ "double complex", { CTF_FP_DCPLX, 0, 128 }, CTF_K_FLOAT },
{ "long double complex", { CTF_FP_LDCPLX, 0, 256 }, CTF_K_FLOAT },
{ NULL, { 0, 0, 0 }, 0 }
};

/*
 * Tables of LP64 intrinsic integer and floating-point type templates to use
 * to populate the dynamic "C" CTF type container.
 */
static const dt_intrinsic_t _dtrace_intrinsics_64[] = {
{ "void", { CTF_INT_SIGNED, 0, 0 }, CTF_K_INTEGER },
{ "signed", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "unsigned", { 0, 0, 32 }, CTF_K_INTEGER },
{ "char", { CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "short", { CTF_INT_SIGNED, 0, 16 }, CTF_K_INTEGER },
{ "int", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "long long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "signed char", { CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "signed short", { CTF_INT_SIGNED, 0, 16 }, CTF_K_INTEGER },
{ "signed int", { CTF_INT_SIGNED, 0, 32 }, CTF_K_INTEGER },
{ "signed long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "signed long long", { CTF_INT_SIGNED, 0, 64 }, CTF_K_INTEGER },
{ "unsigned char", { CTF_INT_CHAR, 0, 8 }, CTF_K_INTEGER },
{ "unsigned short", { 0, 0, 16 }, CTF_K_INTEGER },
{ "unsigned int", { 0, 0, 32 }, CTF_K_INTEGER },
{ "unsigned long", { 0, 0, 64 }, CTF_K_INTEGER },
{ "unsigned long long", { 0, 0, 64 }, CTF_K_INTEGER },
{ "_Bool", { CTF_INT_BOOL, 0, 8 }, CTF_K_INTEGER },
{ "float", { CTF_FP_SINGLE, 0, 32 }, CTF_K_FLOAT },
{ "double", { CTF_FP_DOUBLE, 0, 64 }, CTF_K_FLOAT },
{ "long double", { CTF_FP_LDOUBLE, 0, 128 }, CTF_K_FLOAT },
{ "float imaginary", { CTF_FP_IMAGRY, 0, 32 }, CTF_K_FLOAT },
{ "double imaginary", { CTF_FP_DIMAGRY, 0, 64 }, CTF_K_FLOAT },
{ "long double imaginary", { CTF_FP_LDIMAGRY, 0, 128 }, CTF_K_FLOAT },
{ "float complex", { CTF_FP_CPLX, 0, 64 }, CTF_K_FLOAT },
{ "double complex", { CTF_FP_DCPLX, 0, 128 }, CTF_K_FLOAT },
{ "long double complex", { CTF_FP_LDCPLX, 0, 256 }, CTF_K_FLOAT },
{ NULL, { 0, 0, 0 }, 0 }
};

/*
 * Tables of ILP32 typedefs to use to populate the dynamic "D" CTF container.
 * These aliases ensure that D definitions can use typical <sys/types.h> names.
 */
static const dt_typedef_t _dtrace_typedefs_32[] = {
{ "char", "int8_t" },
{ "short", "int16_t" },
{ "int", "int32_t" },
{ "long long", "int64_t" },
{ "int", "intptr_t" },
```



```
{ "int", "ssize_t" },
{ "unsigned char", "uint8_t" },
{ "unsigned short", "uint16_t" },
{ "unsigned", "uint32_t" },
{ "unsigned long long", "uint64_t" },
{ "unsigned char", "uchar_t" },
{ "unsigned short", "ushort_t" },
{ "unsigned", "uint_t" },
{ "unsigned long", "ulong_t" },
{ "unsigned long long", "u_longlong_t" },
{ "int", "ptrdiff_t" },
{ "unsigned", "uintptr_t" },
{ "unsigned", "size_t" },
{ "long", "id_t" },
{ "long", "pid_t" },
{ NULL, NULL }
};

/*
 * Tables of LP64 typedefs to use to populate the dynamic "D" CTF container.
 * These aliases ensure that D definitions can use typical <sys/types.h> names.
 */
static const dt_typedef_t _dtrace_typedefs_64[] = {
{ "char", "int8_t" },
{ "short", "int16_t" },
{ "int", "int32_t" },
{ "long", "int64_t" },
{ "long", "intptr_t" },
{ "long", "ssize_t" },
{ "unsigned char", "uint8_t" },
{ "unsigned short", "uint16_t" },
{ "unsigned", "uint32_t" },
{ "unsigned long", "uint64_t" },
{ "unsigned char", "uchar_t" },
{ "unsigned short", "ushort_t" },
{ "unsigned", "uint_t" },
{ "unsigned long", "ulong_t" },
{ "unsigned long long", "u_longlong_t" },
{ "long", "ptrdiff_t" },
{ "unsigned long", "uintptr_t" },
{ "unsigned long", "size_t" },
{ "int", "id_t" },
{ "int", "pid_t" },
{ NULL, NULL }
};

/*
 * Tables of ILP32 integer type templates used to populate the dtp->dt_ints[]
 * cache when a new dtrace client open occurs.  Values are set by dtrace_open().
 */
static const dt_intdesc_t _dtrace_ints_32[] = {
{ "int", NULL, CTF_ERR, 0x7fffffffULL },
{ "unsigned int", NULL, CTF_ERR, 0xffffffffULL },
{ "long", NULL, CTF_ERR, 0x7fffffffULL },
{ "unsigned long", NULL, CTF_ERR, 0xffffffffULL },
{ "long long", NULL, CTF_ERR, 0x7fffffffffffffffULL },
{ "unsigned long long", NULL, CTF_ERR, 0xffffffffffffffffULL }
};

/*
 * Tables of LP64 integer type templates used to populate the dtp->dt_ints[]
 * cache when a new dtrace client open occurs.  Values are set by dtrace_open().
 */
static const dt_intdesc_t _dtrace_ints_64[] = {
{ "int", NULL, CTF_ERR, 0x7fffffffULL },
{ "unsigned int", NULL, CTF_ERR, 0xffffffffULL },
```

```
599 { "long", NULL, CTF_ERR, 0x7fffffffffffffffULL },
600 { "unsigned long", NULL, CTF_ERR, 0xffffffffffffffffULL },
601 { "long long", NULL, CTF_ERR, 0x7fffffffffffffffULL },
602 { "unsigned long long", NULL, CTF_ERR, 0xffffffffffffffffULL }
603 };

605 /*
606  * Table of macro variable templates used to populate the macro identifier hash
607  * when a new dtrace client open occurs.  Values are set by dtrace_update().
608  */
609 static const dt_ident_t _dtrace_macros[] = {
610 { "egid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
611 { "euid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
612 { "gid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
613 { "pid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
614 { "pgid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
615 { "ppid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
616 { "projid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
617 { "sid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
618 { "taskid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
619 { "target", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
620 { "uid", DT_IDENT_SCALAR, 0, 0, DT_ATTR_STABCMN, DT_VERS_1_0 },
621 { NULL, 0, 0, 0, { 0, 0, 0 }, 0 }
622 };

624 /*
625  * Hard-wired definition string to be compiled and cached every time a new
626  * DTrace library handle is initialized.  This string should only be used to
627  * contain definitions that should be present regardless of DTRACE_O_NOLIBS.
628  */
629 static const char _dtrace_hardwire[] = "\
630 inline long NULL = 0; \n\
631 #pragma D binding \"1.0\" NULL\n\
632 ";

634 /*
635  * Default DTrace configuration to use when opening libdtrace DTRACE_O_NODEV.
636  * If DTRACE_O_NODEV is not set, we load the configuration from the kernel.
637  * The use of CTF_MODEL_NATIVE is more subtle than it might appear: we are
638  * relying on the fact that when running dtrace(1M), isaexec will invoke the
639  * binary with the same bitness as the kernel, which is what we want by default
640  * when generating our DIF.  The user can override the choice using oflags.
641  */
642 static const dtrace_conf_t _dtrace_conf = {
643         DIF_VERSION,            /* dtc_difversion */
644         DIF_DIR_NREGS,          /* dtc_difintregs */
645         DIF_DTR_NREGS,          /* dtc_diftupregs */
646         CTF_MODEL_NATIVE        /* dtc_ctfmodel */
647 };

649 const dtrace_attribute_t _dtrace_maxattr = {
650         DTRACE_STABILITY_MAX,
651         DTRACE_STABILITY_MAX,
652         DTRACE_CLASS_MAX
653 };

655 const dtrace_attribute_t _dtrace_defattr = {
656         DTRACE_STABILITY_STABLE,
657         DTRACE_STABILITY_STABLE,
658         DTRACE_CLASS_COMMON
659 };

661 const dtrace_attribute_t _dtrace_symattr = {
662         DTRACE_STABILITY_PRIVATE,
663         DTRACE_STABILITY_PRIVATE,
664         DTRACE_CLASS_UNKNOWN
```



```
665 };

667 const dtrace_attribute_t _dtrace_typattr = {
668         DTRACE_STABILITY_PRIVATE,
669         DTRACE_STABILITY_PRIVATE,
670         DTRACE_CLASS_UNKNOWN
671 };

673 const dtrace_attribute_t _dtrace_prvattr = {
674         DTRACE_STABILITY_PRIVATE,
675         DTRACE_STABILITY_PRIVATE,
676         DTRACE_CLASS_UNKNOWN
677 };

679 const dtrace_pattr_t _dtrace_prvdesc = {
680 { DTRACE_STABILITY_UNSTABLE, DTRACE_STABILITY_UNSTABLE, DTRACE_CLASS_COMMON },
681 { DTRACE_STABILITY_UNSTABLE, DTRACE_STABILITY_UNSTABLE, DTRACE_CLASS_COMMON },
682 { DTRACE_STABILITY_UNSTABLE, DTRACE_STABILITY_UNSTABLE, DTRACE_CLASS_COMMON },
683 { DTRACE_STABILITY_UNSTABLE, DTRACE_STABILITY_UNSTABLE, DTRACE_CLASS_COMMON },
684 { DTRACE_STABILITY_UNSTABLE, DTRACE_STABILITY_UNSTABLE, DTRACE_CLASS_COMMON },
685 };

687 const char *_dtrace_defcpp = "/usr/ccs/lib/cpp"; /* default cpp(1) to invoke */
688 const char *_dtrace_defld = "/usr/ccs/bin/ld";   /* default ld(1) to invoke */

690 const char *_dtrace_libdir = "/usr/lib/dtrace"; /* default library directory */
691 const char *_dtrace_provdir = "/dev/dtrace/provider"; /* provider directory */

693 int _dtrace_strbuckets = 211;   /* default number of hash buckets (prime) */
694 int _dtrace_intbuckets = 256;   /* default number of integer buckets (Pof2) */
695 uint_t _dtrace_strsize = 256;   /* default size of string intrinsic type */
696 uint_t _dtrace_stkindent = 14;  /* default whitespace indent for stack/ustack */
697 uint_t _dtrace_pidbuckets = 64; /* default number of pid hash buckets */
698 uint_t _dtrace_pidlrulim = 8;   /* default number of pid handles to cache */
699 size_t _dtrace_bufsize = 512;   /* default dt_buf_create() size */
700 int _dtrace_argmax = 32;        /* default maximum number of probe arguments */

702 int _dtrace_debug = 0;          /* debug messages enabled (off) */
703 const char *const _dtrace_version = DT_VERS_STRING; /* API version string */
704 int _dtrace_rdvers = RD_VERSION; /* rtld_db feature version */

706 typedef struct dt_fdlist {
707         int *df_fds;            /* array of provider driver file descriptors */
708         uint_t df_ents;         /* number of valid elements in df_fds[] */
709         uint_t df_size;         /* size of df_fds[] */
710 } dt_fdlist_t;

712 #pragma init(_dtrace_init)
713 void
714 _dtrace_init(void)
715 {
716         _dtrace_debug = getenv("DTRACE_DEBUG") != NULL;

718         for (; _dtrace_rdvers > 0; _dtrace_rdvers--) {
719                 if (rd_init(_dtrace_rdvers) == RD_OK)
720                         break;
721         }
722 }

724 static dtrace_hdl_t *
725 set_open_errno(dtrace_hdl_t *dtp, int *errp, int err)
726 {
727         if (dtp != NULL)
728                 dtrace_close(dtp);
729         if (errp != NULL)
730                 *errp = err;
```

```
731         return (NULL);
732 }
733
734 static void
735 dt_provmod_open(dt_provmod_t **provmod, dt_fdlist_t *dfp)
736 {
737         dt_provmod_t *prov;
738         char path[PATH_MAX];
739         struct dirent *dp, *ep;
740         DIR *dirp;
741         int fd;
742
743         if ((dirp = opendir(_dtrace_provdir)) == NULL)
744                 return; /* failed to open directory; just skip it */
745
746         ep = alloca(sizeof (struct dirent) + PATH_MAX + 1);
747         bzero(ep, sizeof (struct dirent) + PATH_MAX + 1);
748
749         while (readdir_r(dirp, ep, &dp) == 0 && dp != NULL) {
750                 if (dp->d_name[0] == '.')
751                         continue; /* skip "." and ".." */
752
753                 if (dfp->df_ents == dfp->df_size) {
754                         uint_t size = dfp->df_size ? dfp->df_size * 2 : 16;
755                         int *fds = realloc(dfp->df_fds, size * sizeof (int));
756
757                         if (fds == NULL)
758                                 break; /* skip the rest of this directory */
759
760                         dfp->df_fds = fds;
761                         dfp->df_size = size;
762                 }
763
764                 (void) snprintf(path, sizeof (path), "%s/%s",
765                     _dtrace_provdir, dp->d_name);
766
767                 if ((fd = open(path, O_RDONLY)) == -1)
768                         continue; /* failed to open driver; just skip it */
769
770                 if (((prov = malloc(sizeof (dt_provmod_t))) == NULL) ||
771                     (prov->dp_name = malloc(strlen(dp->d_name) + 1)) == NULL) {
772                         free(prov);
773                         (void) close(fd);
774                         break;
775                 }
776
777                 (void) strcpy(prov->dp_name, dp->d_name);
778                 prov->dp_next = *provmod;
779                 *provmod = prov;
780
781                 dt_dprintf("opened provider %s\n", dp->d_name);
782                 dfp->df_fds[dfp->df_ents++] = fd;
783         }
784
785         (void) closedir(dirp);
786 }
787
788 static void
789 dt_provmod_destroy(dt_provmod_t **provmod)
790 {
791         dt_provmod_t *next, *current;
792
793         for (current = *provmod; current != NULL; current = next) {
794                 next = current->dp_next;
795                 free(current->dp_name);
796                 free(current);
```



```
797         }
798
799         *provmod = NULL;
800 }
801
802 static const char *
803 dt_get_sysinfo(int cmd, char *buf, size_t len)
804 {
805         ssize_t rv = sysinfo(cmd, buf, len);
806         char *p = buf;
807
808         if (rv < 0 || rv > len)
809                 (void) snprintf(buf, len, "%s", "Unknown");
810
811         while ((p = strchr(p, '.')) != NULL)
812                 *p++ = '_';
813
814         return (buf);
815 }
816
817 static dtrace_hdl_t *
818 dt_vopen(int version, int flags, int *errp,
819     const dtrace_vector_t *vector, void *arg)
820 {
821         dtrace_hdl_t *dtp = NULL;
822         int dtfd = -1, ftfd = -1, fterr = 0;
823         dtrace_prog_t *pgp;
824         dt_module_t *dmp;
825         dt_provmod_t *provmod = NULL;
826         int i, err;
827         struct rlimit rl;
828
829         const dt_intrinsic_t *dinp;
830         const dt_typedef_t *dtyp;
831         const dt_ident_t *idp;
832
833         dtrace_typeinfo_t dtt;
834         ctf_funcinfo_t ctc;
835         ctf_arinfo_t ctr;
836
837         dt_fdlist_t df = { NULL, 0, 0 };
838
839         char isadef[32], utsdef[32];
840         char s1[64], s2[64];
841
842         if (version <= 0)
843                 return (set_open_errno(dtp, errp, EINVAL));
844
845         if (version > DTRACE_VERSION)
846                 return (set_open_errno(dtp, errp, EDT_VERSION));
847
848         if (version < DTRACE_VERSION) {
849                 /*
850                  * Currently, increasing the library version number is used to
851                  * denote a binary incompatible change.  That is, a consumer
852                  * of the library cannot run on a version of the library with
853                  * a higher DTRACE_VERSION number than the consumer compiled
854                  * against.  Once the library API has been committed to,
855                  * backwards binary compatibility will be required; at that
856                  * time, this check should change to return EDT_OVERSION only
857                  * if the specified version number is less than the version
858                  * number at the time of interface commitment.
859                  */
860                 return (set_open_errno(dtp, errp, EDT_OVERSION));
861         }
```

```
863         if (flags & ~DTRACE_O_MASK)
864                 return (set_open_errno(dtp, errp, EINVAL));
865
866         if ((flags & DTRACE_O_LP64) && (flags & DTRACE_O_ILP32))
867                 return (set_open_errno(dtp, errp, EINVAL));
868
869         if (vector == NULL && arg != NULL)
870                 return (set_open_errno(dtp, errp, EINVAL));
871
872         if (elf_version(EV_CURRENT) == EV_NONE)
873                 return (set_open_errno(dtp, errp, EDT_ELFVERSION));
874
875         if (vector != NULL || (flags & DTRACE_O_NODEV))
876                 goto alloc; /* do not attempt to open dtrace device */
877
878         /*
879          * Before we get going, crank our limit on file descriptors up to the
880          * hard limit.  This is to allow for the fact that libproc keeps file
881          * descriptors to objects open for the lifetime of the proc handle;
882          * without raising our hard limit, we would have an acceptably small
883          * bound on the number of processes that we could concurrently
884          * instrument with the pid provider.
885          */
886         if (getrlimit(RLIMIT_NOFILE, &rl) == 0) {
887                 rl.rlim_cur = rl.rlim_max;
888                 (void) setrlimit(RLIMIT_NOFILE, &rl);
889         }
890
891         /*
892          * Get the device path of each of the providers.  We hold them open
893          * in the df.df_fds list until we open the DTrace driver itself,
894          * allowing us to see all of the probes provided on this system.  Once
895          * we have the DTrace driver open, we can safely close all the providers
896          * now that they have registered with the framework.
897          */
898         dt_provmod_open(&provmod, &df);
899
900         dtfd = open("/dev/dtrace/dtrace", O_RDWR);
901         err = errno; /* save errno from opening dtfd */
902
903         ftfd = open("/dev/dtrace/provider/fasttrap", O_RDWR);
904         fterr = ftfd == -1 ? errno : 0; /* save errno from open ftfd */
905
906         while (df.df_ents-- != 0)
907                 (void) close(df.df_fds[df.df_ents]);
908
909         free(df.df_fds);
910
911         /*
912          * If we failed to open the dtrace device, fail dtrace_open().
913          * We convert some kernel errnos to custom libdtrace errnos to
914          * improve the resulting message from the usual strerror().
915          */
916         if (dtfd == -1) {
917                 dt_provmod_destroy(&provmod);
918                 switch (err) {
919                 case ENOENT:
920                         err = EDT_NOENT;
921                         break;
922                 case EBUSY:
923                         err = EDT_BUSY;
924                         break;
925                 case EACCES:
926                         err = EDT_ACCESS;
927                         break;
928                 }
```



```
929                 return (set_open_errno(dtp, errp, err));
930         }
931
932         (void) fcntl(dtfd, F_SETFD, FD_CLOEXEC);
933         (void) fcntl(ftfd, F_SETFD, FD_CLOEXEC);
934
935 alloc:
936         if ((dtp = malloc(sizeof (dtrace_hdl_t))) == NULL)
937                 return (set_open_errno(dtp, errp, EDT_NOMEM));
938
939         bzero(dtp, sizeof (dtrace_hdl_t));
940         dtp->dt_oflags = flags;
941         dtp->dt_prcmode = DT_PROC_STOP_PREINIT;
942         dtp->dt_linkmode = DT_LINK_KERNEL;
943         dtp->dt_linktype = DT_LTYP_ELF;
944         dtp->dt_xlatemode = DT_XL_STATIC;
945         dtp->dt_stdcmode = DT_STDC_XA;
946         dtp->dt_encoding = DT_ENCODING_UNSET;
947 #endif /* ! codereview */
948         dtp->dt_version = version;
949         dtp->dt_fd = dtfd;
950         dtp->dt_ftfd = ftfd;
951         dtp->dt_fterr = fterr;
952         dtp->dt_cdefs_fd = -1;
953         dtp->dt_ddefs_fd = -1;
954         dtp->dt_stdout_fd = -1;
955         dtp->dt_modbuckets = _dtrace_strbuckets;
956         dtp->dt_mods = calloc(dtp->dt_modbuckets, sizeof (dt_module_t *));
957         dtp->dt_provbuckets = _dtrace_strbuckets;
958         dtp->dt_provs = calloc(dtp->dt_provbuckets, sizeof (dt_provider_t *));
959         dt_proc_init(dtp);
960         dtp->dt_vmax = DT_VERS_LATEST;
961         dtp->dt_cpp_path = strdup(_dtrace_defcpp);
962         dtp->dt_cpp_argv = malloc(sizeof (char *));
963         dtp->dt_cpp_argc = 1;
964         dtp->dt_cpp_args = 1;
965         dtp->dt_ld_path = strdup(_dtrace_defld);
966         dtp->dt_provmod = provmod;
967         dtp->dt_vector = vector;
968         dtp->dt_varg = arg;
969         dt_dof_init(dtp);
970         (void) uname(&dtp->dt_uts);
971
972         if (dtp->dt_mods == NULL || dtp->dt_provs == NULL ||
973             dtp->dt_procs == NULL || dtp->dt_proc_env == NULL ||
974             dtp->dt_ld_path == NULL || dtp->dt_cpp_path == NULL ||
975             dtp->dt_cpp_argv == NULL)
976                 return (set_open_errno(dtp, errp, EDT_NOMEM));
977
978         for (i = 0; i < DTRACEOPT_MAX; i++)
979                 dtp->dt_options[i] = DTRACEOPT_UNSET;
980
981         dtp->dt_cpp_argv[0] = (char *)strbasename(dtp->dt_cpp_path);
982
983         (void) snprintf(isadef, sizeof (isadef), "-D__SUNW_D_%u",
984             (uint_t)(sizeof (void *) * NBBY));
985
986         (void) snprintf(utsdef, sizeof (utsdef), "-D__%s_%s",
987             dt_get_sysinfo(SI_SYSNAME, s1, sizeof (s1)),
988             dt_get_sysinfo(SI_RELEASE, s2, sizeof (s2)));
989
990         if (dt_cpp_add_arg(dtp, "-D__sun") == NULL ||
991             dt_cpp_add_arg(dtp, "-D__unix") == NULL ||
992             dt_cpp_add_arg(dtp, "-D__SVR4") == NULL ||
993             dt_cpp_add_arg(dtp, "-D__SUNW_D=1") == NULL ||
994             dt_cpp_add_arg(dtp, isadef) == NULL ||
```

```
 995             dt_cpp_add_arg(dtp, utsdef) == NULL)
 996                 return (set_open_errno(dtp, errp, EDT_NOMEM));

 998         if (flags & DTRACE_O_NODEV)
 999                 bcopy(&_dtrace_conf, &dtp->dt_conf, sizeof (_dtrace_conf));
1000         else if (dt_ioctl(dtp, DTRACEIOC_CONF, &dtp->dt_conf) != 0)
1001                 return (set_open_errno(dtp, errp, errno));

1003         if (flags & DTRACE_O_LP64)
1004                 dtp->dt_conf.dtc_ctfmodel = CTF_MODEL_LP64;
1005         else if (flags & DTRACE_O_ILP32)
1006                 dtp->dt_conf.dtc_ctfmodel = CTF_MODEL_ILP32;

1008 #ifdef __sparc
1009         /*
1010          * On SPARC systems, __sparc is always defined for <sys/isa_defs.h>
1011          * and __sparcv9 is defined if we are doing a 64-bit compile.
1012          */
1013         if (dt_cpp_add_arg(dtp, "-D__sparc") == NULL)
1014                 return (set_open_errno(dtp, errp, EDT_NOMEM));

1016         if (dtp->dt_conf.dtc_ctfmodel == CTF_MODEL_LP64 &&
1017             dt_cpp_add_arg(dtp, "-D__sparcv9") == NULL)
1018                 return (set_open_errno(dtp, errp, EDT_NOMEM));
1019 #endif

1021 #ifdef __x86
1022         /*
1023          * On x86 systems, __i386 is defined for <sys/isa_defs.h> for 32-bit
1024          * compiles and __amd64 is defined for 64-bit compiles.  Unlike SPARC,
1025          * they are defined exclusive of one another (see PSARC 2004/619).
1026          */
1027         if (dtp->dt_conf.dtc_ctfmodel == CTF_MODEL_LP64) {
1028                 if (dt_cpp_add_arg(dtp, "-D__amd64") == NULL)
1029                         return (set_open_errno(dtp, errp, EDT_NOMEM));
1030         } else {
1031                 if (dt_cpp_add_arg(dtp, "-D__i386") == NULL)
1032                         return (set_open_errno(dtp, errp, EDT_NOMEM));
1033         }
1034 #endif

1036         if (dtp->dt_conf.dtc_difversion < DIF_VERSION)
1037                 return (set_open_errno(dtp, errp, EDT_DIFVERS));

1039         if (dtp->dt_conf.dtc_ctfmodel == CTF_MODEL_ILP32)
1040                 bcopy(_dtrace_ints_32, dtp->dt_ints, sizeof (_dtrace_ints_32));
1041         else
1042                 bcopy(_dtrace_ints_64, dtp->dt_ints, sizeof (_dtrace_ints_64));

1044         dtp->dt_macros = dt_idhash_create("macro", NULL, 0, UINT_MAX);
1045         dtp->dt_aggs = dt_idhash_create("aggregation", NULL,
1046             DTRACE_AGGVARIDNONE + 1, UINT_MAX);

1048         dtp->dt_globals = dt_idhash_create("global", _dtrace_globals,
1049             DIF_VAR_OTHER_UBASE, DIF_VAR_OTHER_MAX);

1051         dtp->dt_tls = dt_idhash_create("thread local", NULL,
1052             DIF_VAR_OTHER_UBASE, DIF_VAR_OTHER_MAX);

1054         if (dtp->dt_macros == NULL || dtp->dt_aggs == NULL ||
1055             dtp->dt_globals == NULL || dtp->dt_tls == NULL)
1056                 return (set_open_errno(dtp, errp, EDT_NOMEM));

1058         /*
1059          * Populate the dt_macros identifier hash table by hand: we can't use
1060          * the dt_idhash_populate() mechanism because we're not yet compiling
```



```
1061          * and dtrace_update() needs to immediately reference these idents.
1062          */
1063         for (idp = _dtrace_macros; idp->di_name != NULL; idp++) {
1064                 if (dt_idhash_insert(dtp->dt_macros, idp->di_name,
1065                     idp->di_kind, idp->di_flags, idp->di_id, idp->di_attr,
1066                     idp->di_vers, idp->di_ops ? idp->di_ops : &dt_idops_thaw,
1067                     idp->di_iarg, 0) == NULL)
1068                         return (set_open_errno(dtp, errp, EDT_NOMEM));
1069         }

1071         /*
1072          * Update the module list using /system/object and load the values for
1073          * the macro variable definitions according to the current process.
1074          */
1075         dtrace_update(dtp);

1077         /*
1078          * Select the intrinsics and typedefs we want based on the data model.
1079          * The intrinsics are under "C".  The typedefs are added under "D".
1080          */
1081         if (dtp->dt_conf.dtc_ctfmodel == CTF_MODEL_ILP32) {
1082                 dinp = _dtrace_intrinsics_32;
1083                 dtyp = _dtrace_typedefs_32;
1084         } else {
1085                 dinp = _dtrace_intrinsics_64;
1086                 dtyp = _dtrace_typedefs_64;
1087         }

1089         /*
1090          * Create a dynamic CTF container under the "C" scope for intrinsic
1091          * types and types defined in ANSI-C header files that are included.
1092          */
1093         if ((dmp = dtp->dt_cdefs = dt_module_create(dtp, "C")) == NULL)
1094                 return (set_open_errno(dtp, errp, EDT_NOMEM));

1096         if ((dmp->dm_ctfp = ctf_create(&dtp->dt_ctferr)) == NULL)
1097                 return (set_open_errno(dtp, errp, EDT_CTF));

1099         dt_dprintf("created CTF container for %s (%p)\n",
1100             dmp->dm_name, (void *)dmp->dm_ctfp);

1102         (void) ctf_setmodel(dmp->dm_ctfp, dtp->dt_conf.dtc_ctfmodel);
1103         ctf_setspecific(dmp->dm_ctfp, dmp);

1105         dmp->dm_flags = DT_DM_LOADED; /* fake up loaded bit */
1106         dmp->dm_modid = -1; /* no module ID */

1108         /*
1109          * Fill the dynamic "C" CTF container with all of the intrinsic
1110          * integer and floating-point types appropriate for this data model.
1111          */
1112         for (; dinp->din_name != NULL; dinp++) {
1113                 if (dinp->din_kind == CTF_K_INTEGER) {
1114                         err = ctf_add_integer(dmp->dm_ctfp, CTF_ADD_ROOT,
1115                             dinp->din_name, &dinp->din_data);
1116                 } else {
1117                         err = ctf_add_float(dmp->dm_ctfp, CTF_ADD_ROOT,
1118                             dinp->din_name, &dinp->din_data);
1119                 }

1121                 if (err == CTF_ERR) {
1122                         dt_dprintf("failed to add %s to C container: %s\n",
1123                             dinp->din_name, ctf_errmsg(
1124                             ctf_errno(dmp->dm_ctfp)));
1125                         return (set_open_errno(dtp, errp, EDT_CTF));
1126                 }
```

```
1127         }
1129         if (ctf_update(dmp->dm_ctfp) != 0) {
1130                 dt_dprintf("failed to update C container: %s\n",
1131                     ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1132                 return (set_open_errno(dtp, errp, EDT_CTF));
1133         }
1135         /*
1136          * Add intrinsic pointer types that are needed to initialize printf
1137          * format dictionary types (see table in dt_printf.c).
1138          */
1139         (void) ctf_add_pointer(dmp->dm_ctfp, CTF_ADD_ROOT,
1140             ctf_lookup_by_name(dmp->dm_ctfp, "void"));
1142         (void) ctf_add_pointer(dmp->dm_ctfp, CTF_ADD_ROOT,
1143             ctf_lookup_by_name(dmp->dm_ctfp, "char"));
1145         (void) ctf_add_pointer(dmp->dm_ctfp, CTF_ADD_ROOT,
1146             ctf_lookup_by_name(dmp->dm_ctfp, "int"));
1148         if (ctf_update(dmp->dm_ctfp) != 0) {
1149                 dt_dprintf("failed to update C container: %s\n",
1150                     ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1151                 return (set_open_errno(dtp, errp, EDT_CTF));
1152         }
1154         /*
1155          * Create a dynamic CTF container under the "D" scope for types that
1156          * are defined by the D program itself or on-the-fly by the D compiler.
1157          * The "D" CTF container is a child of the "C" CTF container.
1158          */
1159         if ((dmp = dtp->dt_ddefs = dt_module_create(dtp, "D")) == NULL)
1160                 return (set_open_errno(dtp, errp, EDT_NOMEM));
1162         if ((dmp->dm_ctfp = ctf_create(&dtp->dt_ctferr)) == NULL)
1163                 return (set_open_errno(dtp, errp, EDT_CTF));
1165         dt_dprintf("created CTF container for %s (%p)\n",
1166             dmp->dm_name, (void *)dmp->dm_ctfp);
1168         (void) ctf_setmodel(dmp->dm_ctfp, dtp->dt_conf.dtc_ctfmodel);
1169         ctf_setspecific(dmp->dm_ctfp, dmp);
1171         dmp->dm_flags = DT_DM_LOADED; /* fake up loaded bit */
1172         dmp->dm_modid = -1; /* no module ID */
1174         if (ctf_import(dmp->dm_ctfp, dtp->dt_cdefs->dm_ctfp) == CTF_ERR) {
1175                 dt_dprintf("failed to import D parent container: %s\n",
1176                     ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1177                 return (set_open_errno(dtp, errp, EDT_CTF));
1178         }
1180         /*
1181          * Fill the dynamic "D" CTF container with all of the built-in typedefs
1182          * that we need to use for our D variable and function definitions.
1183          * This ensures that basic inttypes.h names are always available to us.
1184          */
1185         for (; dtyp->dty_src != NULL; dtyp++) {
1186                 if (ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1187                     dtyp->dty_dst, ctf_lookup_by_name(dmp->dm_ctfp,
1188                     dtyp->dty_src)) == CTF_ERR) {
1189                         dt_dprintf("failed to add typedef %s %s to D "
1190                             "container: %s", dtyp->dty_src, dtyp->dty_dst,
1191                             ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1192                         return (set_open_errno(dtp, errp, EDT_CTF));
```



```
1193                 }
1194         }
1196         /*
1197          * Insert a CTF ID corresponding to a pointer to a type of kind
1198          * CTF_K_FUNCTION we can use in the compiler for function pointers.
1199          * CTF treats all function pointers as "int (*)()" so we only need one.
1200          */
1201         ctc.ctc_return = ctf_lookup_by_name(dmp->dm_ctfp, "int");
1202         ctc.ctc_argc = 0;
1203         ctc.ctc_flags = 0;
1205         dtp->dt_type_func = ctf_add_function(dmp->dm_ctfp,
1206             CTF_ADD_ROOT, &ctc, NULL);
1208         dtp->dt_type_fptr = ctf_add_pointer(dmp->dm_ctfp,
1209             CTF_ADD_ROOT, dtp->dt_type_func);
1211         /*
1212          * We also insert CTF definitions for the special D intrinsic types
1213          * string and <DYN> into the D container.  The string type is added
1214          * as a typedef of char[n].  The <DYN> type is an alias for void.
1215          * We compare types to these special CTF ids throughout the compiler.
1216          */
1217         ctr.ctr_contents = ctf_lookup_by_name(dmp->dm_ctfp, "char");
1218         ctr.ctr_index = ctf_lookup_by_name(dmp->dm_ctfp, "long");
1219         ctr.ctr_nelems = _dtrace_strsize;
1221         dtp->dt_type_str = ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1222             "string", ctf_add_array(dmp->dm_ctfp, CTF_ADD_ROOT, &ctr));
1224         dtp->dt_type_dyn = ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1225             "<DYN>", ctf_lookup_by_name(dmp->dm_ctfp, "void"));
1227         dtp->dt_type_stack = ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1228             "stack", ctf_lookup_by_name(dmp->dm_ctfp, "void"));
1230         dtp->dt_type_symaddr = ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1231             "_symaddr", ctf_lookup_by_name(dmp->dm_ctfp, "void"));
1233         dtp->dt_type_usymaddr = ctf_add_typedef(dmp->dm_ctfp, CTF_ADD_ROOT,
1234             "_usymaddr", ctf_lookup_by_name(dmp->dm_ctfp, "void"));
1236         if (dtp->dt_type_func == CTF_ERR || dtp->dt_type_fptr == CTF_ERR ||
1237             dtp->dt_type_str == CTF_ERR || dtp->dt_type_dyn == CTF_ERR ||
1238             dtp->dt_type_stack == CTF_ERR || dtp->dt_type_symaddr == CTF_ERR ||
1239             dtp->dt_type_usymaddr == CTF_ERR) {
1240                 dt_dprintf("failed to add intrinsic to D container: %s\n",
1241                     ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1242                 return (set_open_errno(dtp, errp, EDT_CTF));
1243         }
1245         if (ctf_update(dmp->dm_ctfp) != 0) {
1246                 dt_dprintf("failed update D container: %s\n",
1247                     ctf_errmsg(ctf_errno(dmp->dm_ctfp)));
1248                 return (set_open_errno(dtp, errp, EDT_CTF));
1249         }
1251         /*
1252          * Initialize the integer description table used to convert integer
1253          * constants to the appropriate types.  Refer to the comments above
1254          * dt_node_int() for a complete description of how this table is used.
1255          */
1256         for (i = 0; i < sizeof (dtp->dt_ints) / sizeof (dtp->dt_ints[0]); i++) {
1257                 if (dtrace_lookup_by_type(dtp, DTRACE_OBJ_EVERY,
1258                     dtp->dt_ints[i].did_name, &dtt) != 0) {
```

```
                        dt_dprintf("failed to lookup integer type %s: %s\n",
                            dtp->dt_ints[i].did_name,
                            dtrace_errmsg(dtp, dtrace_errno(dtp)));
                        return (set_open_errno(dtp, errp, dtp->dt_errno));
                }
                dtp->dt_ints[i].did_ctfp = dtt.dtt_ctfp;
                dtp->dt_ints[i].did_type = dtt.dtt_type;
        }

        /*
         * Now that we've created the "C" and "D" containers, move them to the
         * start of the module list so that these types and symbols are found
         * first (for stability) when iterating through the module list.
         */
        dt_list_delete(&dtp->dt_modlist, dtp->dt_ddefs);
        dt_list_prepend(&dtp->dt_modlist, dtp->dt_ddefs);

        dt_list_delete(&dtp->dt_modlist, dtp->dt_cdefs);
        dt_list_prepend(&dtp->dt_modlist, dtp->dt_cdefs);

        if (dt_pfdict_create(dtp) == -1)
                return (set_open_errno(dtp, errp, dtp->dt_errno));

        /*
         * If we are opening libdtrace DTRACE_O_NODEV enable C_ZDEFS by default
         * because without /dev/dtrace open, we will not be able to load the
         * names and attributes of any providers or probes from the kernel.
         */
        if (flags & DTRACE_O_NODEV)
                dtp->dt_cflags |= DTRACE_C_ZDEFS;

        /*
         * Load hard-wired inlines into the definition cache by calling the
         * compiler on the raw definition string defined above.
         */
        if ((pgp = dtrace_program_strcompile(dtp, _dtrace_hardwire,
            DTRACE_PROBESPEC_NONE, DTRACE_C_EMPTY, 0, NULL)) == NULL) {
                dt_dprintf("failed to load hard-wired definitions: %s\n",
                    dtrace_errmsg(dtp, dtrace_errno(dtp)));
                return (set_open_errno(dtp, errp, EDT_HARDWIRE));
        }

        dt_program_destroy(dtp, pgp);

        /*
         * Set up the default DTrace library path.  Once set, the next call to
         * dt_compile() will compile all the libraries.  We intentionally defer
         * library processing to improve overhead for clients that don't ever
         * compile, and to provide better error reporting (because the full
         * reporting of compiler errors requires dtrace_open() to succeed).
         */
        if (dtrace_setopt(dtp, "libdir", _dtrace_libdir) != 0)
                return (set_open_errno(dtp, errp, dtp->dt_errno));

        return (dtp);
}

dtrace_hdl_t *
dtrace_open(int version, int flags, int *errp)
{
        return (dt_vopen(version, flags, errp, NULL, NULL));
}

dtrace_hdl_t *
dtrace_vopen(int version, int flags, int *errp,
    const dtrace_vector_t *vector, void *arg)
```



```
{
        return (dt_vopen(version, flags, errp, vector, arg));
}

void
dtrace_close(dtrace_hdl_t *dtp)
{
        dt_ident_t *idp, *ndp;
        dt_module_t *dmp;
        dt_provider_t *pvp;
        dtrace_prog_t *pgp;
        dt_xlator_t *dxp;
        dt_dirpath_t *dirp;
        int i;

        if (dtp->dt_procs != NULL)
                dt_proc_fini(dtp);

        while ((pgp = dt_list_next(&dtp->dt_programs)) != NULL)
                dt_program_destroy(dtp, pgp);

        while ((dxp = dt_list_next(&dtp->dt_xlators)) != NULL)
                dt_xlator_destroy(dtp, dxp);

        dt_free(dtp, dtp->dt_xlatormap);

        for (idp = dtp->dt_externs; idp != NULL; idp = ndp) {
                ndp = idp->di_next;
                dt_ident_destroy(idp);
        }

        if (dtp->dt_macros != NULL)
                dt_idhash_destroy(dtp->dt_macros);
        if (dtp->dt_aggs != NULL)
                dt_idhash_destroy(dtp->dt_aggs);
        if (dtp->dt_globals != NULL)
                dt_idhash_destroy(dtp->dt_globals);
        if (dtp->dt_tls != NULL)
                dt_idhash_destroy(dtp->dt_tls);

        while ((dmp = dt_list_next(&dtp->dt_modlist)) != NULL)
                dt_module_destroy(dtp, dmp);

        while ((pvp = dt_list_next(&dtp->dt_provlist)) != NULL)
                dt_provider_destroy(dtp, pvp);

        if (dtp->dt_fd != -1)
                (void) close(dtp->dt_fd);
        if (dtp->dt_ftfd != -1)
                (void) close(dtp->dt_ftfd);
        if (dtp->dt_cdefs_fd != -1)
                (void) close(dtp->dt_cdefs_fd);
        if (dtp->dt_ddefs_fd != -1)
                (void) close(dtp->dt_ddefs_fd);
        if (dtp->dt_stdout_fd != -1)
                (void) close(dtp->dt_stdout_fd);

        dt_epid_destroy(dtp);
        dt_aggid_destroy(dtp);
        dt_format_destroy(dtp);
        dt_strdata_destroy(dtp);
        dt_buffered_destroy(dtp);
        dt_aggregate_destroy(dtp);
        dt_pfdict_destroy(dtp);
        dt_provmod_destroy(&dtp->dt_provmod);
        dt_dof_fini(dtp);
```

```
1392         for (i = 1; i < dtp->dt_cpp_argc; i++)
1393                 free(dtp->dt_cpp_argv[i]);

1395         while ((dirp = dt_list_next(&dtp->dt_lib_path)) != NULL) {
1396                 dt_list_delete(&dtp->dt_lib_path, dirp);
1397                 free(dirp->dir_path);
1398                 free(dirp);
1399         }

1401         free(dtp->dt_cpp_argv);
1402         free(dtp->dt_cpp_path);
1403         free(dtp->dt_ld_path);

1405         free(dtp->dt_mods);
1406         free(dtp->dt_provs);
1407         free(dtp);
1408 }

1410 int
1411 dtrace_provider_modules(dtrace_hdl_t *dtp, const char **mods, int nmods)
1412 {
1413         dt_provmod_t *prov;
1414         int i = 0;

1416         for (prov = dtp->dt_provmod; prov != NULL; prov = prov->dp_next, i++) {
1417                 if (i < nmods)
1418                         mods[i] = prov->dp_name;
1419         }

1421         return (i);
1422 }

1424 int
1425 dtrace_ctlfd(dtrace_hdl_t *dtp)
1426 {
1427         return (dtp->dt_fd);
1428 }
```

```
**********************************************************
   24534 Tue Jan 14 20:13:06 2014
new/usr/src/lib/libdtrace/common/dt_options.c
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2013, Joyent, Inc. All rights reserved.
#endif /* ! codereview */
 * Copyright (c) 2012 by Delphix. All rights reserved.
 */

#include <sys/resource.h>
#include <sys/mman.h>
#include <sys/types.h>

#include <strings.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>
#include <alloca.h>
#include <errno.h>
#include <fcntl.h>

#include <dt_impl.h>
#include <dt_string.h>

static int
dt_opt_agg(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        dt_aggregate_t *agp = &dtp->dt_aggregate;

        if (arg != NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        agp->dtat_flags |= option;
        return (0);
}
```



```
/*ARGSUSED*/
static int
dt_opt_amin(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        char str[DTRACE_ATTR2STR_MAX];
        dtrace_attribute_t attr;

        if (arg == NULL || dtrace_str2attr(arg, &attr) == -1)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        dt_dprintf("set compiler attribute minimum to %s\n",
            dtrace_attr2str(attr, str, sizeof (str)));

        if (dtp->dt_pcb != NULL) {
                dtp->dt_pcb->pcb_cflags |= DTRACE_C_EATTR;
                dtp->dt_pcb->pcb_amin = attr;
        } else {
                dtp->dt_cflags |= DTRACE_C_EATTR;
                dtp->dt_amin = attr;
        }

        return (0);
}

static void
dt_coredump(void)
{
        const char msg[] = "libdtrace DEBUG: [ forcing coredump ]\n";

        struct sigaction act;
        struct rlimit lim;

        (void) write(STDERR_FILENO, msg, sizeof (msg) - 1);

        act.sa_handler = SIG_DFL;
        act.sa_flags = 0;

        (void) sigemptyset(&act.sa_mask);
        (void) sigaction(SIGABRT, &act, NULL);

        lim.rlim_cur = RLIM_INFINITY;
        lim.rlim_max = RLIM_INFINITY;

        (void) setrlimit(RLIMIT_CORE, &lim);
        abort();
}

/*ARGSUSED*/
static int
dt_opt_core(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        static int enabled = 0;

        if (arg != NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        if (enabled++ || atexit(dt_coredump) == 0)
                return (0);

        return (dt_set_errno(dtp, errno));
}

/*ARGSUSED*/
static int
dt_opt_cpp_hdrs(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
```

```
126 {
127         if (arg != NULL)
128                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
129
130         if (dtp->dt_pcb != NULL)
131                 return (dt_set_errno(dtp, EDT_BADOPTCTX));
132
133         if (dt_cpp_add_arg(dtp, "-H") == NULL)
134                 return (dt_set_errno(dtp, EDT_NOMEM));
135
136         return (0);
137 }
138
139 /*ARGSUSED*/
140 static int
141 dt_opt_cpp_path(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
142 {
143         char *cpp;
144
145         if (arg == NULL)
146                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
147
148         if (dtp->dt_pcb != NULL)
149                 return (dt_set_errno(dtp, EDT_BADOPTCTX));
150
151         if ((cpp = strdup(arg)) == NULL)
152                 return (dt_set_errno(dtp, EDT_NOMEM));
153
154         dtp->dt_cpp_argv[0] = (char *)strbasename(cpp);
155         free(dtp->dt_cpp_path);
156         dtp->dt_cpp_path = cpp;
157
158         return (0);
159 }
160
161 static int
162 dt_opt_cpp_opts(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
163 {
164         char *buf;
165         size_t len;
166         const char *opt = (const char *)option;
167
168         if (opt == NULL || arg == NULL)
169                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
170
171         if (dtp->dt_pcb != NULL)
172                 return (dt_set_errno(dtp, EDT_BADOPTCTX));
173
174         len = strlen(opt) + strlen(arg) + 1;
175         buf = alloca(len);
176
177         (void) strcpy(buf, opt);
178         (void) strcat(buf, arg);
179
180         if (dt_cpp_add_arg(dtp, buf) == NULL)
181                 return (dt_set_errno(dtp, EDT_NOMEM));
182
183         return (0);
184 }
185
186 /*ARGSUSED*/
187 static int
188 dt_opt_ctypes(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
189 {
190         int fd;
```



```
192         if (arg == NULL)
193                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
194
195         if ((fd = open64(arg, O_CREAT | O_WRONLY, 0666)) == -1)
196                 return (dt_set_errno(dtp, errno));
197
198         (void) close(dtp->dt_cdefs_fd);
199         dtp->dt_cdefs_fd = fd;
200         return (0);
201 }
202
203 /*ARGSUSED*/
204 static int
205 dt_opt_droptags(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
206 {
207         dtp->dt_droptags = 1;
208         return (0);
209 }
210
211 /*ARGSUSED*/
212 static int
213 dt_opt_dtypes(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
214 {
215         int fd;
216
217         if (arg == NULL)
218                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
219
220         if ((fd = open64(arg, O_CREAT | O_WRONLY, 0666)) == -1)
221                 return (dt_set_errno(dtp, errno));
222
223         (void) close(dtp->dt_ddefs_fd);
224         dtp->dt_ddefs_fd = fd;
225         return (0);
226 }
227
228 /*ARGSUSED*/
229 static int
230 dt_opt_debug(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
231 {
232         if (arg != NULL)
233                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
234
235         _dtrace_debug = 1;
236         return (0);
237 }
238
239 /*ARGSUSED*/
240 static int
241 dt_opt_iregs(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
242 {
243         int n;
244
245         if (arg == NULL || (n = atoi(arg)) <= 0)
246                 return (dt_set_errno(dtp, EDT_BADOPTVAL));
247
248         dtp->dt_conf.dtc_difintregs = n;
249         return (0);
250 }
251
252 /*ARGSUSED*/
253 static int
254 dt_opt_lazyload(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
255 {
256         dtp->dt_lazyload = 1;
```

```
258         return (0);
259 }

261 /*ARGSUSED*/
262 static int
263 dt_opt_ld_path(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
264 {
265         char *ld;

267         if (arg == NULL)
268                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

270         if (dtp->dt_pcb != NULL)
271                 return (dt_set_errno(dtp, EDT_BADOPTCTX));

273         if ((ld = strdup(arg)) == NULL)
274                 return (dt_set_errno(dtp, EDT_NOMEM));

276         free(dtp->dt_ld_path);
277         dtp->dt_ld_path = ld;

279         return (0);
280 }

282 /*ARGSUSED*/
283 static int
284 dt_opt_libdir(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
285 {
286         dt_dirpath_t *dp;

288         if (arg == NULL)
289                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

291         if ((dp = malloc(sizeof (dt_dirpath_t))) == NULL ||
292             (dp->dir_path = strdup(arg)) == NULL) {
293                 free(dp);
294                 return (dt_set_errno(dtp, EDT_NOMEM));
295         }

297         dt_list_append(&dtp->dt_lib_path, dp);
298         return (0);
299 }

301 /*ARGSUSED*/
302 static int
303 dt_opt_linkmode(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
304 {
305         if (arg == NULL)
306                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

308         if (strcmp(arg, "kernel") == 0)
309                 dtp->dt_linkmode = DT_LINK_KERNEL;
310         else if (strcmp(arg, "primary") == 0)
311                 dtp->dt_linkmode = DT_LINK_PRIMARY;
312         else if (strcmp(arg, "dynamic") == 0)
313                 dtp->dt_linkmode = DT_LINK_DYNAMIC;
314         else if (strcmp(arg, "static") == 0)
315                 dtp->dt_linkmode = DT_LINK_STATIC;
316         else
317                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

319         return (0);
320 }

322 /*ARGSUSED*/
323 static int
```



```
324 dt_opt_linktype(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
325 {
326         if (arg == NULL)
327                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

329         if (strcasecmp(arg, "elf") == 0)
330                 dtp->dt_linktype = DT_LTYP_ELF;
331         else if (strcasecmp(arg, "dof") == 0)
332                 dtp->dt_linktype = DT_LTYP_DOF;
333         else
334                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

336         return (0);
337 }

339 /*ARGSUSED*/
340 static int
341 dt_opt_encoding(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
342 {
343         if (arg == NULL)
344                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

346         if (strcmp(arg, "ascii") == 0)
347                 dtp->dt_encoding = DT_ENCODING_ASCII;
348         else if (strcmp(arg, "utf8") == 0)
349                 dtp->dt_encoding = DT_ENCODING_UTF8;
350         else
351                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

353         return (0);
354 }

356 /*ARGSUSED*/
357 static int
358 #endif /* ! codereview */
359 dt_opt_evaltime(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
360 {
361         if (arg == NULL)
362                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

364         if (strcmp(arg, "exec") == 0)
365                 dtp->dt_prcmode = DT_PROC_STOP_CREATE;
366         else if (strcmp(arg, "preinit") == 0)
367                 dtp->dt_prcmode = DT_PROC_STOP_PREINIT;
368         else if (strcmp(arg, "postinit") == 0)
369                 dtp->dt_prcmode = DT_PROC_STOP_POSTINIT;
370         else if (strcmp(arg, "main") == 0)
371                 dtp->dt_prcmode = DT_PROC_STOP_MAIN;
372         else
373                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

375         return (0);
376 }

378 /*ARGSUSED*/
379 static int
380 dt_opt_pgmax(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
381 {
382         int n;

384         if (arg == NULL || (n = atoi(arg)) < 0)
385                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

387         dtp->dt_procs->dph_lrulim = n;
388         return (0);
389 }
```

```
static int
dt_opt_setenv(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        char **p;
        char *var;
        int i;

        /*
         * We can't effectively set environment variables from #pragma lines
         * since the processes have already been spawned.
         */
        if (dtp->dt_pcb != NULL)
                return (dt_set_errno(dtp, EDT_BADOPTCTX));

        if (arg == NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        if (!option && strchr(arg, '=') != NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        for (i = 1, p = dtp->dt_proc_env; *p != NULL; i++, p++)
                continue;

        for (p = dtp->dt_proc_env; *p != NULL; p++) {
                var = strchr(*p, '=');
                if (var == NULL)
                        var = *p + strlen(*p);
                if (strncmp(*p, arg, var - *p) == 0) {
                        dt_free(dtp, *p);
                        *p = dtp->dt_proc_env[i - 1];
                        dtp->dt_proc_env[i - 1] = NULL;
                        i--;
                }
        }

        if (option) {
                if ((var = strdup(arg)) == NULL)
                        return (dt_set_errno(dtp, EDT_NOMEM));

                if ((p = dt_alloc(dtp, sizeof (char *) * (i + 1))) == NULL) {
                        dt_free(dtp, var);
                        return (dt_set_errno(dtp, EDT_NOMEM));
                }

                bcopy(dtp->dt_proc_env, p, sizeof (char *) * i);
                dt_free(dtp, dtp->dt_proc_env);
                dtp->dt_proc_env = p;

                dtp->dt_proc_env[i - 1] = var;
                dtp->dt_proc_env[i] = NULL;
        }

        return (0);
}

/*ARGSUSED*/
static int
dt_opt_stdc(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        if (arg == NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        if (dtp->dt_pcb != NULL)
                return (dt_set_errno(dtp, EDT_BADOPTCTX));
```



```
        if (strcmp(arg, "a") == 0)
                dtp->dt_stdcmode = DT_STDC_XA;
        else if (strcmp(arg, "c") == 0)
                dtp->dt_stdcmode = DT_STDC_XC;
        else if (strcmp(arg, "s") == 0)
                dtp->dt_stdcmode = DT_STDC_XS;
        else if (strcmp(arg, "t") == 0)
                dtp->dt_stdcmode = DT_STDC_XT;
        else
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        return (0);
}

/*ARGSUSED*/
static int
dt_opt_syslibdir(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        dt_dirpath_t *dp = dt_list_next(&dtp->dt_lib_path);
        char *path;

        if (arg == NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        if ((path = strdup(arg)) == NULL)
                return (dt_set_errno(dtp, EDT_NOMEM));

        free(dp->dir_path);
        dp->dir_path = path;

        return (0);
}

/*ARGSUSED*/
static int
dt_opt_tree(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        int m;

        if (arg == NULL || (m = atoi(arg)) <= 0)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        dtp->dt_treedump = m;
        return (0);
}

/*ARGSUSED*/
static int
dt_opt_tregs(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        int n;

        if (arg == NULL || (n = atoi(arg)) <= 0)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));

        dtp->dt_conf.dtc_diftupregs = n;
        return (0);
}

/*ARGSUSED*/
static int
dt_opt_xlate(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
{
        if (arg == NULL)
                return (dt_set_errno(dtp, EDT_BADOPTVAL));
```

```
522         if (strcmp(arg, "dynamic") == 0)
523                 dtp->dt_xlatemode = DT_XL_DYNAMIC;
524         else if (strcmp(arg, "static") == 0)
525                 dtp->dt_xlatemode = DT_XL_STATIC;
526         else
527                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

529         return (0);
530 }

532 /*ARGSUSED*/
533 static int
534 dt_opt_cflags(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
535 {
536         if (arg != NULL)
537                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

539         if (dtp->dt_pcb != NULL)
540                 dtp->dt_pcb->pcb_cflags |= option;
541         else
542                 dtp->dt_cflags |= option;

544         return (0);
545 }

547 static int
548 dt_opt_dflags(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
549 {
550         if (arg != NULL)
551                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

553         dtp->dt_dflags |= option;
554         return (0);
555 }

557 static int
558 dt_opt_invcflags(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
559 {
560         if (arg != NULL)
561                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

563         if (dtp->dt_pcb != NULL)
564                 dtp->dt_pcb->pcb_cflags &= ~option;
565         else
566                 dtp->dt_cflags &= ~option;

568         return (0);
569 }

571 /*ARGSUSED*/
572 static int
573 dt_opt_version(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
574 {
575         dt_version_t v;

577         if (arg == NULL)
578                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

580         if (dt_version_str2num(arg, &v) == -1)
581                 return (dt_set_errno(dtp, EDT_VERSINVAL));

583         if (!dt_version_defined(v))
584                 return (dt_set_errno(dtp, EDT_VERSUNDEF));

586         return (dt_reduce(dtp, v));
587 }
```



```
589 static int
590 dt_opt_runtime(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
591 {
592         char *end;
593         dtrace_optval_t val = 0;
594         int i;

596         const struct {
597                 char *positive;
598                 char *negative;
599         } couples[] = {
600                 { "yes",        "no" },
601                 { "enable",     "disable" },
602                 { "enabled",    "disabled" },
603                 { "true",       "false" },
604                 { "on",         "off" },
605                 { "set",        "unset" },
606                 { NULL }
607         };

609         if (arg != NULL) {
610                 if (arg[0] == '\0') {
611                         val = DTRACEOPT_UNSET;
612                         goto out;
613                 }

615                 for (i = 0; couples[i].positive != NULL; i++) {
616                         if (strcasecmp(couples[i].positive, arg) == 0) {
617                                 val = 1;
618                                 goto out;
619                         }

621                         if (strcasecmp(couples[i].negative, arg) == 0) {
622                                 val = DTRACEOPT_UNSET;
623                                 goto out;
624                         }
625                 }

627                 errno = 0;
628                 val = strtoull(arg, &end, 0);

630                 if (*end != '\0' || errno != 0 || val < 0)
631                         return (dt_set_errno(dtp, EDT_BADOPTVAL));
632         }

634 out:
635         dtp->dt_options[option] = val;
636         return (0);
637 }

639 static int
640 dt_optval_parse(const char *arg, dtrace_optval_t *rval)
641 {
642         dtrace_optval_t mul = 1;
643         size_t len;
644         char *end;

646         len = strlen(arg);
647         errno = 0;

649         switch (arg[len - 1]) {
650         case 't':
651         case 'T':
652                 mul *= 1024;
653                 /*FALLTHRU*/
```

```
654         case 'g':
655         case 'G':
656                 mul *= 1024;
657                 /*FALLTHRU*/
658         case 'm':
659         case 'M':
660                 mul *= 1024;
661                 /*FALLTHRU*/
662         case 'k':
663         case 'K':
664                 mul *= 1024;
665                 /*FALLTHRU*/
666         default:
667                 break;
668         }

670         errno = 0;
671         *rval = strtoull(arg, &end, 0) * mul;

673         if ((mul > 1 && end != &arg[len - 1]) || (mul == 1 && *end != '\0') ||
674             *rval < 0 || errno != 0)
675                 return (-1);

677         return (0);
678 }

680 static int
681 dt_opt_size(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
682 {
683         dtrace_optval_t val = 0;

685         if (arg != NULL && dt_optval_parse(arg, &val) != 0)
686                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

688         dtp->dt_options[option] = val;
689         return (0);
690 }

692 static int
693 dt_opt_rate(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
694 {
695         char *end;
696         int i;
697         dtrace_optval_t mul = 1, val = 0;

699         const struct {
700                 char *name;
701                 hrtime_t mul;
702         } suffix[] = {
703                 { "ns",         NANOSEC / NANOSEC },
704                 { "nsec",       NANOSEC / NANOSEC },
705                 { "us",         NANOSEC / MICROSEC },
706                 { "usec",       NANOSEC / MICROSEC },
707                 { "ms",         NANOSEC / MILLISEC },
708                 { "msec",       NANOSEC / MILLISEC },
709                 { "s",          NANOSEC / SEC },
710                 { "sec",        NANOSEC / SEC },
711                 { "m",          NANOSEC * (hrtime_t)60 },
712                 { "min",        NANOSEC * (hrtime_t)60 },
713                 { "h",          NANOSEC * (hrtime_t)60 * (hrtime_t)60 },
714                 { "hour",       NANOSEC * (hrtime_t)60 * (hrtime_t)60 },
715                 { "d",          NANOSEC * (hrtime_t)(24 * 60 * 60) },
716                 { "day",        NANOSEC * (hrtime_t)(24 * 60 * 60) },
717                 { "hz",         0 },
718                 { NULL }
719         };
```



```
721         if (arg != NULL) {
722                 errno = 0;
723                 val = strtoull(arg, &end, 0);

725                 for (i = 0; suffix[i].name != NULL; i++) {
726                         if (strcasecmp(suffix[i].name, end) == 0) {
727                                 mul = suffix[i].mul;
728                                 break;
729                         }
730                 }

732                 if (suffix[i].name == NULL && *end != '\0' || val < 0)
733                         return (dt_set_errno(dtp, EDT_BADOPTVAL));

735                 if (mul == 0) {
736                         /*
737                          * The rate has been specified in frequency-per-second.
738                          */
739                         if (val != 0)
740                                 val = NANOSEC / val;
741                 } else {
742                         val *= mul;
743                 }
744         }

746         dtp->dt_options[option] = val;
747         return (0);
748 }

750 /*
751  * When setting the strsize option, set the option in the dt_options array
752  * using dt_opt_size() as usual, and then update the definition of the CTF
753  * type for the D intrinsic "string" to be an array of the corresponding size.
754  * If any errors occur, reset dt_options[option] to its previous value.
755  */
756 static int
757 dt_opt_strsize(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
758 {
759         dtrace_optval_t val = dtp->dt_options[option];
760         ctf_file_t *fp = DT_STR_CTFP(dtp);
761         ctf_id_t type = ctf_type_resolve(fp, DT_STR_TYPE(dtp));
762         ctf_arinfo_t r;

764         if (dt_opt_size(dtp, arg, option) != 0)
765                 return (-1); /* dt_errno is set for us */

767         if (dtp->dt_options[option] > UINT_MAX) {
768                 dtp->dt_options[option] = val;
769                 return (dt_set_errno(dtp, EOVERFLOW));
770         }

772         if (ctf_array_info(fp, type, &r) == CTF_ERR) {
773                 dtp->dt_options[option] = val;
774                 dtp->dt_ctferr = ctf_errno(fp);
775                 return (dt_set_errno(dtp, EDT_CTF));
776         }

778         r.ctr_nelems = (uint_t)dtp->dt_options[option];

780         if (ctf_set_array(fp, type, &r) == CTF_ERR ||
781             ctf_update(fp) == CTF_ERR) {
782                 dtp->dt_options[option] = val;
783                 dtp->dt_ctferr = ctf_errno(fp);
784                 return (dt_set_errno(dtp, EDT_CTF));
785         }
```

```
787         return (0);
788 }

790 static const struct {
791         const char *dtbp_name;
792         int dtbp_policy;
793 } _dtrace_bufpolicies[] = {
794         { "ring", DTRACEOPT_BUFPOLICY_RING },
795         { "fill", DTRACEOPT_BUFPOLICY_FILL },
796         { "switch", DTRACEOPT_BUFPOLICY_SWITCH },
797         { NULL, 0 }
798 };

800 /*ARGSUSED*/
801 static int
802 dt_opt_bufpolicy(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
803 {
804         dtrace_optval_t policy = DTRACEOPT_UNSET;
805         int i;

807         if (arg == NULL)
808                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

810         for (i = 0; _dtrace_bufpolicies[i].dtbp_name != NULL; i++) {
811                 if (strcmp(_dtrace_bufpolicies[i].dtbp_name, arg) == 0) {
812                         policy = _dtrace_bufpolicies[i].dtbp_policy;
813                         break;
814                 }
815         }

817         if (policy == DTRACEOPT_UNSET)
818                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

820         dtp->dt_options[DTRACEOPT_BUFPOLICY] = policy;

822         return (0);
823 }

825 static const struct {
826         const char *dtbr_name;
827         int dtbr_policy;
828 } _dtrace_bufresize[] = {
829         { "auto", DTRACEOPT_BUFRESIZE_AUTO },
830         { "manual", DTRACEOPT_BUFRESIZE_MANUAL },
831         { NULL, 0 }
832 };

834 /*ARGSUSED*/
835 static int
836 dt_opt_bufresize(dtrace_hdl_t *dtp, const char *arg, uintptr_t option)
837 {
838         dtrace_optval_t policy = DTRACEOPT_UNSET;
839         int i;

841         if (arg == NULL)
842                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

844         for (i = 0; _dtrace_bufresize[i].dtbr_name != NULL; i++) {
845                 if (strcmp(_dtrace_bufresize[i].dtbr_name, arg) == 0) {
846                         policy = _dtrace_bufresize[i].dtbr_policy;
847                         break;
848                 }
849         }

851         if (policy == DTRACEOPT_UNSET)
```



```
852                 return (dt_set_errno(dtp, EDT_BADOPTVAL));

854         dtp->dt_options[DTRACEOPT_BUFRESIZE] = policy;

856         return (0);
857 }

859 int
860 dt_options_load(dtrace_hdl_t *dtp)
861 {
862         dof_hdr_t hdr, *dof;
863         dof_sec_t *sec;
864         size_t offs;
865         int i;

867         /*
868          * To load the option values, we need to ask the kernel to provide its
869          * DOF, which we'll sift through to look for OPTDESC sections.
870          */
871         bzero(&hdr, sizeof (dof_hdr_t));
872         hdr.dofh_loadsz = sizeof (dof_hdr_t);

874         if (dt_ioctl(dtp, DTRACEIOC_DOFGET, &hdr) == -1)
875                 return (dt_set_errno(dtp, errno));

877         if (hdr.dofh_loadsz < sizeof (dof_hdr_t))
878                 return (dt_set_errno(dtp, EINVAL));

880         dof = alloca(hdr.dofh_loadsz);
881         bzero(dof, sizeof (dof_hdr_t));
882         dof->dofh_loadsz = hdr.dofh_loadsz;

884         for (i = 0; i < DTRACEOPT_MAX; i++)
885                 dtp->dt_options[i] = DTRACEOPT_UNSET;

887         if (dt_ioctl(dtp, DTRACEIOC_DOFGET, dof) == -1)
888                 return (dt_set_errno(dtp, errno));

890         for (i = 0; i < dof->dofh_secnum; i++) {
891                 sec = (dof_sec_t *)(uintptr_t)((uintptr_t)dof +
892                     dof->dofh_secoff + i * dof->dofh_secsize);

894                 if (sec->dofs_type != DOF_SECT_OPTDESC)
895                         continue;

897                 break;
898         }

900         for (offs = 0; offs < sec->dofs_size; offs += sec->dofs_entsize) {
901                 dof_optdesc_t *opt = (dof_optdesc_t *)(uintptr_t)
902                     ((uintptr_t)dof + sec->dofs_offset + offs);

904                 if (opt->dofo_strtab != DOF_SECIDX_NONE)
905                         continue;

907                 if (opt->dofo_option >= DTRACEOPT_MAX)
908                         continue;

910                 dtp->dt_options[opt->dofo_option] = opt->dofo_value;
911         }

913         return (0);
914 }

916 typedef struct dt_option {
917         const char *o_name;
```

```
        int (*o_func)(dtrace_hdl_t *, const char *, uintptr_t);
        uintptr_t o_option;
} dt_option_t;

/*
 * Compile-time options.
 */
static const dt_option_t _dtrace_ctoptions[] = {
        { "aggpercpu", dt_opt_agg, DTRACE_A_PERCPU },
        { "amin", dt_opt_amin },
        { "argref", dt_opt_cflags, DTRACE_C_ARGREF },
        { "core", dt_opt_core },
        { "cpp", dt_opt_cflags, DTRACE_C_CPP },
        { "cpphdrs", dt_opt_cpp_hdrs },
        { "cpppath", dt_opt_cpp_path },
        { "ctypes", dt_opt_ctypes },
        { "defaultargs", dt_opt_cflags, DTRACE_C_DEFARG },
        { "dtypes", dt_opt_dtypes },
        { "debug", dt_opt_debug },
        { "define", dt_opt_cpp_opts, (uintptr_t)"-D" },
        { "droptags", dt_opt_droptags },
        { "empty", dt_opt_cflags, DTRACE_C_EMPTY },
        { "encoding", dt_opt_encoding },
#endif /* ! codereview */
        { "errtags", dt_opt_cflags, DTRACE_C_ETAGS },
        { "evaltime", dt_opt_evaltime },
        { "incdir", dt_opt_cpp_opts, (uintptr_t)"-I" },
        { "iregs", dt_opt_iregs },
        { "kdefs", dt_opt_invcflags, DTRACE_C_KNODEF },
        { "knodefs", dt_opt_cflags, DTRACE_C_KNODEF },
        { "late", dt_opt_xlate },
        { "lazyload", dt_opt_lazyload },
        { "ldpath", dt_opt_ld_path },
        { "libdir", dt_opt_libdir },
        { "linkmode", dt_opt_linkmode },
        { "linktype", dt_opt_linktype },
        { "nolibs", dt_opt_cflags, DTRACE_C_NOLIBS },
        { "pgmax", dt_opt_pgmax },
        { "pspec", dt_opt_cflags, DTRACE_C_PSPEC },
        { "setenv", dt_opt_setenv, 1 },
        { "stdc", dt_opt_stdc },
        { "strip", dt_opt_dflags, DTRACE_D_STRIP },
        { "syslibdir", dt_opt_syslibdir },
        { "tree", dt_opt_tree },
        { "tregs", dt_opt_tregs },
        { "udefs", dt_opt_invcflags, DTRACE_C_UNODEF },
        { "undef", dt_opt_cpp_opts, (uintptr_t)"-U" },
        { "unodefs", dt_opt_cflags, DTRACE_C_UNODEF },
        { "unsetenv", dt_opt_setenv, 0 },
        { "verbose", dt_opt_cflags, DTRACE_C_DIFV },
        { "version", dt_opt_version },
        { "zdefs", dt_opt_cflags, DTRACE_C_ZDEFS },
        { NULL }
};

/*
 * Run-time options.
 */
static const dt_option_t _dtrace_rtoptions[] = {
        { "aggsize", dt_opt_size, DTRACEOPT_AGGSIZE },
        { "bufsize", dt_opt_size, DTRACEOPT_BUFSIZE },
        { "bufpolicy", dt_opt_bufpolicy, DTRACEOPT_BUFPOLICY },
        { "bufresize", dt_opt_bufresize, DTRACEOPT_BUFRESIZE },
        { "cleanrate", dt_opt_rate, DTRACEOPT_CLEANRATE },
        { "cpu", dt_opt_runtime, DTRACEOPT_CPU },
        { "destructive", dt_opt_runtime, DTRACEOPT_DESTRUCTIVE },
```



```
        { "dynvarsize", dt_opt_size, DTRACEOPT_DYNVARSIZE },
        { "grabanon", dt_opt_runtime, DTRACEOPT_GRABANON },
        { "jstackframes", dt_opt_runtime, DTRACEOPT_JSTACKFRAMES },
        { "jstackstrsize", dt_opt_size, DTRACEOPT_JSTACKSTRSIZE },
        { "nspec", dt_opt_runtime, DTRACEOPT_NSPEC },
        { "specsize", dt_opt_size, DTRACEOPT_SPECSIZE },
        { "stackframes", dt_opt_runtime, DTRACEOPT_STACKFRAMES },
        { "statusrate", dt_opt_rate, DTRACEOPT_STATUSRATE },
        { "strsize", dt_opt_strsize, DTRACEOPT_STRSIZE },
        { "ustackframes", dt_opt_runtime, DTRACEOPT_USTACKFRAMES },
        { "temporal", dt_opt_runtime, DTRACEOPT_TEMPORAL },
        { NULL }
};

/*
 * Dynamic run-time options.
 */
static const dt_option_t _dtrace_drtoptions[] = {
        { "agghist", dt_opt_runtime, DTRACEOPT_AGGHIST },
        { "aggpack", dt_opt_runtime, DTRACEOPT_AGGPACK },
#endif /* ! codereview */
        { "aggrate", dt_opt_rate, DTRACEOPT_AGGRATE },
        { "aggsortkey", dt_opt_runtime, DTRACEOPT_AGGSORTKEY },
        { "aggsortkeypos", dt_opt_runtime, DTRACEOPT_AGGSORTKEYPOS },
        { "aggsortpos", dt_opt_runtime, DTRACEOPT_AGGSORTPOS },
        { "aggsortrev", dt_opt_runtime, DTRACEOPT_AGGSORTREV },
        { "aggzoom", dt_opt_runtime, DTRACEOPT_AGGZOOM },
#endif /* ! codereview */
        { "flowindent", dt_opt_runtime, DTRACEOPT_FLOWINDENT },
        { "quiet", dt_opt_runtime, DTRACEOPT_QUIET },
        { "rawbytes", dt_opt_runtime, DTRACEOPT_RAWBYTES },
        { "stackindent", dt_opt_runtime, DTRACEOPT_STACKINDENT },
        { "switchrate", dt_opt_rate, DTRACEOPT_SWITCHRATE },
        { NULL }
};

int
dtrace_getopt(dtrace_hdl_t *dtp, const char *opt, dtrace_optval_t *val)
{
        const dt_option_t *op;

        if (opt == NULL)
                return (dt_set_errno(dtp, EINVAL));

        /*
         * We only need to search the run-time options -- it's not legal
         * to get the values of compile-time options.
         */
        for (op = _dtrace_rtoptions; op->o_name != NULL; op++) {
                if (strcmp(op->o_name, opt) == 0) {
                        *val = dtp->dt_options[op->o_option];
                        return (0);
                }
        }

        for (op = _dtrace_drtoptions; op->o_name != NULL; op++) {
                if (strcmp(op->o_name, opt) == 0) {
                        *val = dtp->dt_options[op->o_option];
                        return (0);
                }
        }

        return (dt_set_errno(dtp, EDT_BADOPTNAME));
}

int
```

```
1050 dtrace_setopt(dtrace_hdl_t *dtp, const char *opt, const char *val)
1051 {
1052         const dt_option_t *op;

1054         if (opt == NULL)
1055                 return (dt_set_errno(dtp, EINVAL));

1057         for (op = _dtrace_ctoptions; op->o_name != NULL; op++) {
1058                 if (strcmp(op->o_name, opt) == 0)
1059                         return (op->o_func(dtp, val, op->o_option));
1060         }

1062         for (op = _dtrace_drtoptions; op->o_name != NULL; op++) {
1063                 if (strcmp(op->o_name, opt) == 0)
1064                         return (op->o_func(dtp, val, op->o_option));
1065         }

1067         for (op = _dtrace_rtoptions; op->o_name != NULL; op++) {
1068                 if (strcmp(op->o_name, opt) == 0) {
1069                         /*
1070                          * Only dynamic run-time options may be set while
1071                          * tracing is active.
1072                          */
1073                         if (dtp->dt_active)
1074                                 return (dt_set_errno(dtp, EDT_ACTIVE));

1076                         return (op->o_func(dtp, val, op->o_option));
1077                 }
1078         }

1080         return (dt_set_errno(dtp, EDT_BADOPTNAME));
1081 }
```

```
**********************************************************
   23816 Tue Jan 14 20:13:06 2014
new/usr/src/lib/libdtrace/common/dtrace.h
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
______unchanged_portion_omitted_

 345 typedef int dtrace_handle_setopt_f(const dtrace_setoptdata_t *, void *);
 346 extern int dtrace_handle_setopt(dtrace_hdl_t *,
 347     dtrace_handle_setopt_f *, void *);

 349 /*
 350  * DTrace Aggregate Interface
 351  */

 353 #define DTRACE_A_PERCPU         0x0001
 354 #define DTRACE_A_KEEPDELTA      0x0002
 355 #define DTRACE_A_ANONYMOUS      0x0004
 356 #define DTRACE_A_TOTAL          0x0008
 357 #define DTRACE_A_MINMAXBIN      0x0010
 358 #define DTRACE_A_HASNEGATIVES   0x0020
 359 #define DTRACE_A_HASPOSITIVES   0x0040

 361 #define DTRACE_AGGZOOM_MAX              0.95    /* height of max bar */
 362 #endif /* ! codereview */

 364 #define DTRACE_AGGWALK_ERROR            -1      /* error while processing */
 365 #define DTRACE_AGGWALK_NEXT             0       /* proceed to next element */
 366 #define DTRACE_AGGWALK_ABORT            1       /* abort aggregation walk */
 367 #define DTRACE_AGGWALK_CLEAR            2       /* clear this element */
 368 #define DTRACE_AGGWALK_NORMALIZE        3       /* normalize this element */
 369 #define DTRACE_AGGWALK_DENORMALIZE      4       /* denormalize this element */
 370 #define DTRACE_AGGWALK_REMOVE           5       /* remove this element */

 372 struct dtrace_aggdata {
 373         dtrace_hdl_t *dtada_handle;             /* handle to DTrace library */
 374         dtrace_aggdesc_t *dtada_desc;           /* aggregation description */
 375         dtrace_eprobedesc_t *dtada_edesc;       /* enabled probe description */
 376         dtrace_probedesc_t *dtada_pdesc;        /* probe description */
 377         caddr_t dtada_data;                     /* pointer to raw data */
 378         uint64_t dtada_normal;                  /* the normal -- 1 for denorm */
 379         size_t dtada_size;                      /* total size of the data */
 380         caddr_t dtada_delta;                    /* delta data, if available */
 381         caddr_t *dtada_percpu;                  /* per CPU data, if avail */
 382         caddr_t *dtada_percpu_delta;            /* per CPU delta, if avail */
 383         int64_t dtada_total;                    /* per agg total, if avail */
 384         uint16_t dtada_minbin;                  /* minimum bin, if avail */
 385         uint16_t dtada_maxbin;                  /* maximum bin, if avail */
 386         uint32_t dtada_flags;                   /* flags */
 387 #endif /* ! codereview */
 388 };

 390 typedef int dtrace_aggregate_f(const dtrace_aggdata_t *, void *);
 391 typedef int dtrace_aggregate_walk_f(dtrace_hdl_t *,
 392     dtrace_aggregate_f *, void *);
 393 typedef int dtrace_aggregate_walk_joined_f(const dtrace_aggdata_t **,
 394     const int, void *);

 396 extern void dtrace_aggregate_clear(dtrace_hdl_t *);
 397 extern int dtrace_aggregate_snap(dtrace_hdl_t *);
 398 extern int dtrace_aggregate_print(dtrace_hdl_t *, FILE *,
 399     dtrace_aggregate_walk_f *);

 401 extern int dtrace_aggregate_walk(dtrace_hdl_t *, dtrace_aggregate_f *, void *);
```



```
 403 extern int dtrace_aggregate_walk_joined(dtrace_hdl_t *,
 404     dtrace_aggvarid_t *, int, dtrace_aggregate_walk_joined_f *, void *);

 406 extern int dtrace_aggregate_walk_sorted(dtrace_hdl_t *,
 407     dtrace_aggregate_f *, void *);

 409 extern int dtrace_aggregate_walk_keysorted(dtrace_hdl_t *,
 410     dtrace_aggregate_f *, void *);

 412 extern int dtrace_aggregate_walk_valsorted(dtrace_hdl_t *,
 413     dtrace_aggregate_f *, void *);

 415 extern int dtrace_aggregate_walk_keyvarsorted(dtrace_hdl_t *,
 416     dtrace_aggregate_f *, void *);

 418 extern int dtrace_aggregate_walk_valvarsorted(dtrace_hdl_t *,
 419     dtrace_aggregate_f *, void *);

 421 extern int dtrace_aggregate_walk_keyrevsorted(dtrace_hdl_t *,
 422     dtrace_aggregate_f *, void *);

 424 extern int dtrace_aggregate_walk_valrevsorted(dtrace_hdl_t *,
 425     dtrace_aggregate_f *, void *);

 427 extern int dtrace_aggregate_walk_keyvarrevsorted(dtrace_hdl_t *,
 428     dtrace_aggregate_f *, void *);

 430 extern int dtrace_aggregate_walk_valvarrevsorted(dtrace_hdl_t *,
 431     dtrace_aggregate_f *, void *);

 433 #define DTRACE_AGD_PRINTED      0x1     /* aggregation printed in program */

 435 /*
 436  * DTrace Process Control Interface
 437  *
 438  * Library clients who wish to have libdtrace create or grab processes for
 439  * monitoring of their symbol table changes may use these interfaces to
 440  * request that libdtrace obtain control of the process using libproc.
 441  */

 443 extern struct ps_prochandle *dtrace_proc_create(dtrace_hdl_t *,
 444     const char *, char *const *);

 446 extern struct ps_prochandle *dtrace_proc_grab(dtrace_hdl_t *, pid_t, int);
 447 extern void dtrace_proc_release(dtrace_hdl_t *, struct ps_prochandle *);
 448 extern void dtrace_proc_continue(dtrace_hdl_t *, struct ps_prochandle *);

 450 /*
 451  * DTrace Object, Symbol, and Type Interfaces
 452  *
 453  * Library clients can use libdtrace to perform symbol and C type information
 454  * lookups by symbol name, symbol address, or C type name, or to lookup meta-
 455  * information cached for each of the program objects in use by DTrace.  The
 456  * resulting struct contain pointers to arbitrary-length strings, including
 457  * object, symbol, and type names, that are persistent until the next call to
 458  * dtrace_update().  Once dtrace_update() is called, any cached values must
 459  * be flushed and not used subsequently by the client program.
 460  */

 462 #define DTRACE_OBJ_EXEC  ((const char *)0L)     /* primary executable file */
 463 #define DTRACE_OBJ_RTLD  ((const char *)1L)     /* run-time link-editor */
 464 #define DTRACE_OBJ_CDEFS ((const char *)2L)     /* C include definitions */
 465 #define DTRACE_OBJ_DDEFS ((const char *)3L)     /* D program definitions */
 466 #define DTRACE_OBJ_EVERY ((const char *)-1L)    /* all known objects */
 467 #define DTRACE_OBJ_KMODS ((const char *)-2L)    /* all kernel objects */
```

```
#define DTRACE_OBJ_UMODS ((const char *)-3L)    /* all user objects */

typedef struct dtrace_objinfo {
        const char *dto_name;                   /* object file scope name */
        const char *dto_file;                   /* object file path (if any) */
        int dto_id;                             /* object file id (if any) */
        uint_t dto_flags;                       /* object flags (see below) */
        GElf_Addr dto_text_va;                  /* address of text section */
        GElf_Xword dto_text_size;               /* size of text section */
        GElf_Addr dto_data_va;                  /* address of data section */
        GElf_Xword dto_data_size;               /* size of data section */
        GElf_Addr dto_bss_va;                   /* address of BSS */
        GElf_Xword dto_bss_size;                /* size of BSS */
} dtrace_objinfo_t;

#define DTRACE_OBJ_F_KERNEL     0x1             /* object is a kernel module */
#define DTRACE_OBJ_F_PRIMARY    0x2             /* object is a primary module */

typedef int dtrace_obj_f(dtrace_hdl_t *, const dtrace_objinfo_t *, void *);

extern int dtrace_object_iter(dtrace_hdl_t *, dtrace_obj_f *, void *);
extern int dtrace_object_info(dtrace_hdl_t *, const char *, dtrace_objinfo_t *);

typedef struct dtrace_syminfo {
        const char *dts_object;                 /* object name */
        const char *dts_name;                   /* symbol name */
        ulong_t dts_id;                         /* symbol id */
} dtrace_syminfo_t;

extern int dtrace_lookup_by_name(dtrace_hdl_t *, const char *, const char *,
    GElf_Sym *, dtrace_syminfo_t *);

extern int dtrace_lookup_by_addr(dtrace_hdl_t *, GElf_Addr addr,
    GElf_Sym *, dtrace_syminfo_t *);

typedef struct dtrace_typeinfo {
        const char *dtt_object;                 /* object containing type */
        ctf_file_t *dtt_ctfp;                   /* CTF container handle */
        ctf_id_t dtt_type;                      /* CTF type identifier */
        uint_t dtt_flags;                       /* Misc. flags */
} dtrace_typeinfo_t;

#define DTT_FL_USER     0x1                     /* user type */

extern int dtrace_lookup_by_type(dtrace_hdl_t *, const char *, const char *,
    dtrace_typeinfo_t *);

extern int dtrace_symbol_type(dtrace_hdl_t *, const GElf_Sym *,
    const dtrace_syminfo_t *, dtrace_typeinfo_t *);

extern int dtrace_type_strcompile(dtrace_hdl_t *,
    const char *, dtrace_typeinfo_t *);

extern int dtrace_type_fcompile(dtrace_hdl_t *,
    FILE *, dtrace_typeinfo_t *);

/*
 * DTrace Probe Interface
 *
 * Library clients can use these functions to iterate over the set of available
 * probe definitions and inquire as to their attributes.  The probe iteration
 * interfaces report probes that are declared as well as those from dtrace(7D).
 */
typedef struct dtrace_probeinfo {
        dtrace_attribute_t dtp_attr;            /* name attributes */
        dtrace_attribute_t dtp_arga;            /* arg attributes */
```



```
        const dtrace_typeinfo_t *dtp_argv;      /* arg types */
        int dtp_argc;                           /* arg count */
} dtrace_probeinfo_t;

typedef int dtrace_probe_f(dtrace_hdl_t *, const dtrace_probedesc_t *, void *);

extern int dtrace_probe_iter(dtrace_hdl_t *,
    const dtrace_probedesc_t *pdp, dtrace_probe_f *, void *);

extern int dtrace_probe_info(dtrace_hdl_t *,
    const dtrace_probedesc_t *, dtrace_probeinfo_t *);

/*
 * DTrace Vector Interface
 *
 * The DTrace library normally speaks directly to dtrace(7D).  However,
 * this communication may be vectored elsewhere.  Consumers who wish to
 * perform a vectored open must fill in the vector, and use the dtrace_vopen()
 * entry point to obtain a library handle.
 */
struct dtrace_vector {
        int (*dtv_ioctl)(void *, int, void *);
        int (*dtv_lookup_by_addr)(void *, GElf_Addr, GElf_Sym *,
            dtrace_syminfo_t *);
        int (*dtv_status)(void *, processorid_t);
        long (*dtv_sysconf)(void *, int);
};

/*
 * DTrace Utility Functions
 *
 * Library clients can use these functions to convert addresses strings, to
 * convert between string and integer probe descriptions and the
 * dtrace_probedesc_t representation, and to perform similar conversions on
 * stability attributes.
 */
extern int dtrace_addr2str(dtrace_hdl_t *, uint64_t, char *, int);
extern int dtrace_uaddr2str(dtrace_hdl_t *, pid_t, uint64_t, char *, int);

extern int dtrace_xstr2desc(dtrace_hdl_t *, dtrace_probespec_t,
    const char *, int, char *const [], dtrace_probedesc_t *);

extern int dtrace_str2desc(dtrace_hdl_t *, dtrace_probespec_t,
    const char *, dtrace_probedesc_t *);

extern int dtrace_id2desc(dtrace_hdl_t *, dtrace_id_t, dtrace_probedesc_t *);

#define DTRACE_DESC2STR_MAX     1024    /* min buf size for dtrace_desc2str() */

extern char *dtrace_desc2str(const dtrace_probedesc_t *, char *, size_t);

#define DTRACE_ATTR2STR_MAX     64      /* min buf size for dtrace_attr2str() */

extern char *dtrace_attr2str(dtrace_attribute_t, char *, size_t);
extern int dtrace_str2attr(const char *, dtrace_attribute_t *);

extern const char *dtrace_stability_name(dtrace_stability_t);
extern const char *dtrace_class_name(dtrace_class_t);

extern int dtrace_provider_modules(dtrace_hdl_t *, const char **, int);

extern const char *const _dtrace_version;
extern int _dtrace_debug;

#ifdef  __cplusplus
}
```

```
600 #endif

602 #endif  /* _DTRACE_H */
```

```
**********************************************************
  123163 Tue Jan 14 20:13:07 2014
new/usr/src/pkg/manifests/system-dtrace-tests.mf
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright (c) 2012 by Delphix. All rights reserved.
  25 #

  27 set name=pkg.fmri value=pkg:/system/dtrace/tests@$(PKGVERS)
  28 set name=pkg.description value="DTrace Test Suite Internal Distribution"
  29 set name=pkg.summary value="DTrace Test Suite"
  30 set name=info.classification \
  31     value=org.opensolaris.category.2008:Development/System
  32 set name=variant.arch value=$(ARCH)
  33 dir path=opt/SUNWdtrt group=sys
  34 dir path=opt/SUNWdtrt/bin
  35 dir path=opt/SUNWdtrt/bin/$(ARCH32)
  36 dir path=opt/SUNWdtrt/bin/$(ARCH64)
  37 dir path=opt/SUNWdtrt/lib
  38 dir path=opt/SUNWdtrt/lib/java
  39 dir path=opt/SUNWdtrt/tst
  40 dir path=opt/SUNWdtrt/tst/$(ARCH)
  41 dir path=opt/SUNWdtrt/tst/$(ARCH)/arrays
  42 $(i386_ONLY)dir path=opt/SUNWdtrt/tst/$(ARCH)/funcs
  43 dir path=opt/SUNWdtrt/tst/$(ARCH)/pid
  44 $(sparc_ONLY)dir path=opt/SUNWdtrt/tst/$(ARCH)/usdt
  45 dir path=opt/SUNWdtrt/tst/$(ARCH)/ustack
  46 dir path=opt/SUNWdtrt/tst/common
  47 dir path=opt/SUNWdtrt/tst/common/aggs
  48 dir path=opt/SUNWdtrt/tst/common/arithmetic
  49 dir path=opt/SUNWdtrt/tst/common/arrays
  50 dir path=opt/SUNWdtrt/tst/common/assocs
  51 dir path=opt/SUNWdtrt/tst/common/begin
  52 dir path=opt/SUNWdtrt/tst/common/bitfields
  53 dir path=opt/SUNWdtrt/tst/common/buffering
  54 dir path=opt/SUNWdtrt/tst/common/builtinvar
  55 dir path=opt/SUNWdtrt/tst/common/cg
  56 dir path=opt/SUNWdtrt/tst/common/clauses
  57 dir path=opt/SUNWdtrt/tst/common/cpc
  58 dir path=opt/SUNWdtrt/tst/common/decls
  59 dir path=opt/SUNWdtrt/tst/common/drops
```



```
  60 dir path=opt/SUNWdtrt/tst/common/dtraceUtil
  61 dir path=opt/SUNWdtrt/tst/common/end
  62 dir path=opt/SUNWdtrt/tst/common/enum
  63 dir path=opt/SUNWdtrt/tst/common/env
  64 dir path=opt/SUNWdtrt/tst/common/error
  65 dir path=opt/SUNWdtrt/tst/common/exit
  66 dir path=opt/SUNWdtrt/tst/common/fbtprovider
  67 dir path=opt/SUNWdtrt/tst/common/funcs
  68 dir path=opt/SUNWdtrt/tst/common/grammar
  69 dir path=opt/SUNWdtrt/tst/common/include
  70 dir path=opt/SUNWdtrt/tst/common/inline
  71 dir path=opt/SUNWdtrt/tst/common/io
  72 dir path=opt/SUNWdtrt/tst/common/ip
  73 dir path=opt/SUNWdtrt/tst/common/java_api
  74 dir path=opt/SUNWdtrt/tst/common/json
  75 dir path=opt/SUNWdtrt/tst/common/lexer
  76 dir path=opt/SUNWdtrt/tst/common/llquantize
  77 dir path=opt/SUNWdtrt/tst/common/mdb
  78 dir path=opt/SUNWdtrt/tst/common/mib
  79 dir path=opt/SUNWdtrt/tst/common/misc
  80 dir path=opt/SUNWdtrt/tst/common/multiaggs
  81 dir path=opt/SUNWdtrt/tst/common/nfs
  82 dir path=opt/SUNWdtrt/tst/common/offsetof
  83 dir path=opt/SUNWdtrt/tst/common/operators
  84 dir path=opt/SUNWdtrt/tst/common/pid
  85 dir path=opt/SUNWdtrt/tst/common/plockstat
  86 dir path=opt/SUNWdtrt/tst/common/pointers
  87 dir path=opt/SUNWdtrt/tst/common/pragma
  88 dir path=opt/SUNWdtrt/tst/common/predicates
  89 dir path=opt/SUNWdtrt/tst/common/preprocessor
  90 dir path=opt/SUNWdtrt/tst/common/print
  91 dir path=opt/SUNWdtrt/tst/common/printa
  92 dir path=opt/SUNWdtrt/tst/common/printf
  93 dir path=opt/SUNWdtrt/tst/common/privs
  94 dir path=opt/SUNWdtrt/tst/common/probes
  95 dir path=opt/SUNWdtrt/tst/common/proc
  96 dir path=opt/SUNWdtrt/tst/common/profile-n
  97 dir path=opt/SUNWdtrt/tst/common/providers
  98 dir path=opt/SUNWdtrt/tst/common/raise
  99 dir path=opt/SUNWdtrt/tst/common/rates
 100 dir path=opt/SUNWdtrt/tst/common/safety
 101 dir path=opt/SUNWdtrt/tst/common/scalars
 102 dir path=opt/SUNWdtrt/tst/common/sched
 103 dir path=opt/SUNWdtrt/tst/common/scripting
 104 dir path=opt/SUNWdtrt/tst/common/sdt
 105 dir path=opt/SUNWdtrt/tst/common/sizeof
 106 dir path=opt/SUNWdtrt/tst/common/speculation
 107 dir path=opt/SUNWdtrt/tst/common/stability
 108 dir path=opt/SUNWdtrt/tst/common/stack
 109 dir path=opt/SUNWdtrt/tst/common/stackdepth
 110 dir path=opt/SUNWdtrt/tst/common/stop
 111 dir path=opt/SUNWdtrt/tst/common/strlen
 112 dir path=opt/SUNWdtrt/tst/common/strtoll
 113 dir path=opt/SUNWdtrt/tst/common/struct
 114 dir path=opt/SUNWdtrt/tst/common/syscall
 115 dir path=opt/SUNWdtrt/tst/common/sysevent
 116 dir path=opt/SUNWdtrt/tst/common/tick-n
 117 dir path=opt/SUNWdtrt/tst/common/trace
 118 dir path=opt/SUNWdtrt/tst/common/tracemem
 119 dir path=opt/SUNWdtrt/tst/common/translators
 120 dir path=opt/SUNWdtrt/tst/common/typedef
 121 dir path=opt/SUNWdtrt/tst/common/types
 122 dir path=opt/SUNWdtrt/tst/common/uctf
 123 dir path=opt/SUNWdtrt/tst/common/union
 124 dir path=opt/SUNWdtrt/tst/common/usdt
 125 dir path=opt/SUNWdtrt/tst/common/ustack
```

```
dir path=opt/SUNWdtrt/tst/common/vars
dir path=opt/SUNWdtrt/tst/common/version
$(i386_ONLY)dir path=opt/SUNWdtrt/tst/i86xpv
$(i386_ONLY)dir path=opt/SUNWdtrt/tst/i86xpv/xdt
file path=opt/SUNWdtrt/README mode=0444
file path=opt/SUNWdtrt/bin/$(ARCH32)/chkargs mode=0555
file path=opt/SUNWdtrt/bin/$(ARCH64)/chkargs mode=0555
file path=opt/SUNWdtrt/bin/baddof mode=0555
file path=opt/SUNWdtrt/bin/badioctl mode=0555
file path=opt/SUNWdtrt/bin/chkargs mode=0555
file path=opt/SUNWdtrt/bin/dstyle mode=0555
file path=opt/SUNWdtrt/bin/dtest mode=0555
file path=opt/SUNWdtrt/bin/dtfailures mode=0555
file path=opt/SUNWdtrt/bin/exception.lst mode=0444
file path=opt/SUNWdtrt/bin/jdtrace mode=0555
file path=opt/SUNWdtrt/lib/java/jdtrace.jar
file path=opt/SUNWdtrt/tst/$(ARCH)/arrays/tst.uregsarray.d mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/funcs/tst.badcopyin.d mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/funcs/tst.badcopyinstr.d \
    mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/funcs/tst.badcopyout.d \
    mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/funcs/tst.badcopyoutstr.d \
    mode=0444
$(sparc_ONLY)file \
    path=opt/SUNWdtrt/tst/$(ARCH)/pid/err.D_PROC_ALIGN.misaligned.d mode=0444
$(sparc_ONLY)file \
    path=opt/SUNWdtrt/tst/$(ARCH)/pid/err.D_PROC_ALIGN.misaligned.exe \
    mode=0555
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.badinstr.d mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.badinstr.exe mode=0555
$(sparc_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.br.d mode=0444
$(sparc_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.br.d.out mode=0444
$(sparc_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.br.exe mode=0555
file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.branch.d mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.branch.exe mode=0555
file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.embedded.d mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.embedded.exe mode=0555
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.ret.d mode=0444
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.ret.exe mode=0555
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.retlist.exe mode=0555
$(i386_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/pid/tst.retlist.ksh mode=0444
$(sparc_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/usdt/tst.tailcall.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.annotated.d mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.annotated.d.out mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.annotated.exe mode=0555
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.circstack.d mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.circstack.exe mode=0555
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.helper.d mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.helper.d.out mode=0444
file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.helper.exe mode=0555
$(sparc_ONLY)file path=opt/SUNWdtrt/tst/$(ARCH)/ustack/tst.trapstat.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_FUNC.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_MDIM.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_NULL.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_REDEF.redef.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.avgtoofew.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.maxnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.mintoofew.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.quantizetoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.stddevtoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_AGG_SCALAR.sumtoofew.d mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/aggs/err.D_CLEAR_AGGARG.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_CLEAR_PROTO.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_FUNC_IDENT.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_FUNC_UNDEF.badaggfunc.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_IDENT_UNDEF.badexpr.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_IDENT_UNDEF.badkey3.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_IDENT_UNDEF.noeffect.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_KEY_TYPE.badkey1.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_KEY_TYPE.badkey2.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_KEY_TYPE.badkey4.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_BASETYPE.lqbad1.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_BASETYPE.lqshort.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_BASEVAL.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_LIMTYPE.lqbad1.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_LIMVAL.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MATCHBASE.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MATCHBASE.order.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MATCHLIM.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MATCHLIM.order.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MATCHSTEP.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_MISMATCH.lqbadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_STEPLARGE.lqtoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_STEPSMALL.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_STEPTYPE.lqbadinc.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_LQUANT_STEPVAL.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_NORMALIZE_AGGARG.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_NORMALIZE_PROTO.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_NORMALIZE_SCALAR.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_ARG.lquantizetoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.avgnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.avgtoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.counttoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.lquantizenoarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.lquantizetoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.maxnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.maxtoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.minnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.mintoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.quantizenoarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.quantizetoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.stddevnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.stddevtoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.sumnoarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_PROTO_LEN.sumtoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_TRUNC_AGGARG.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_TRUNC_PROTO.badmany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_TRUNC_PROTO.badnone.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/err.D_TRUNC_SCALAR.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggencoding.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggencoding.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.agghist.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.agghist.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpack.d mode=0444
```

```
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpack.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpackbanner.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpackbanner.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpackzoom.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggpackzoom.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggzoom.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.aggzoom.d.out mode=0444
#endif /* ! codereview */
file path=opt/SUNWdtrt/tst/common/aggs/tst.allquant.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.allquant.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.avg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.avg.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.avg_neg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.avg_neg.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clear.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clear.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearavg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearavg.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearavg2.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearavg2.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.cleardenormalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.cleardenormalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearlquantize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearlquantize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearnormalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearnormalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearstddev.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.clearstddev.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.count.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.count.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.count2.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.count2.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.count3.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.denormalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.denormalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.denormalizeonly.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.denormalizeonly.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.fmtnormalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.fmtnormalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.forms.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.forms.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.goodkey.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.keysort.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.keysort.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantnormal.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantnormal.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantrange.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantrange.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantround.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantround.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantzero.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.lquantzero.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.max.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.max.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.max_neg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.max_neg.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.min.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.min.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.min_neg.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.min_neg.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multiaggs1.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multiaggs2.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multiaggs2.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multiaggs3.d mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/aggs/tst.multiaggs3.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multinormalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.multinormalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.neglquant.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.neglquant.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negorder.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negorder.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negquant.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negquant.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negtrunc.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negtrunc.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negtruncquant.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.negtruncquant.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.normalize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.normalize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.order.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.order.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantize.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantmany.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantmany.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantround.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantround.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantzero.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.quantzero.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.signature.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.signedkeys.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.signedkeys.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.signedkeyspos.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.signedkeyspos.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.sizedkeys.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.sizedkeys.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.stddev.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.stddev.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.subr.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.sum.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.sum.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.trunc.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.trunc.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.trunc0.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.trunc0.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.truncquant.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.truncquant.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.valsortkeypos.d mode=0444
file path=opt/SUNWdtrt/tst/common/aggs/tst.valsortkeypos.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_DIV_ZERO.divby0.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_DIV_ZERO.divby0_1.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_DIV_ZERO.divby0_2.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_DIV_ZERO.modby0.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_SYNTAX.addmin.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_SYNTAX.divmin.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_SYNTAX.muladd.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/err.D_SYNTAX.muldiv.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.basics.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.basics.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.compcast.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.compcast.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.compnarrowassign.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.compnarrowassign.d.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.execcast.d mode=0444
file path=opt/SUNWdtrt/tst/common/arithmetic/tst.execcast.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/arrays/err.D_ARR_BADREF.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/arrays/err.D_DECL_ARRBIG.toobig.d mode=0444
```

```
390 file path=opt/SUNWdtrt/tst/common/arrays/err.D_DECL_ARRNULL.bad.d mode=0444
391 file path=opt/SUNWdtrt/tst/common/arrays/err.D_DECL_ARRSUB.bad.d mode=0444
392 file path=opt/SUNWdtrt/tst/common/arrays/err.D_DECL_PROTO_TYPE.badtuple.d \
393     mode=0444
394 file path=opt/SUNWdtrt/tst/common/arrays/err.D_IDENT_UNDEF.badureg.d mode=0444
395 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic1.d mode=0444
396 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic2.d mode=0444
397 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic3.d mode=0444
398 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic4.d mode=0444
399 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic5.d mode=0444
400 file path=opt/SUNWdtrt/tst/common/arrays/tst.basic6.d mode=0444
401 file path=opt/SUNWdtrt/tst/common/arrays/tst.uregsarray.d mode=0444
402 file path=opt/SUNWdtrt/tst/common/assocs/err.D_OP_INCOMPAT.dupgtype.d \
403     mode=0444
404 file path=opt/SUNWdtrt/tst/common/assocs/err.D_OP_INCOMPAT.dupttype.d \
405     mode=0444
406 file path=opt/SUNWdtrt/tst/common/assocs/err.D_OP_INCOMPAT.this.d mode=0444
407 file path=opt/SUNWdtrt/tst/common/assocs/err.D_PROTO_ARG.badsig.d mode=0444
408 file path=opt/SUNWdtrt/tst/common/assocs/err.D_PROTO_LEN.toofew.d mode=0444
409 file path=opt/SUNWdtrt/tst/common/assocs/err.D_PROTO_LEN.toomany.d mode=0444
410 file path=opt/SUNWdtrt/tst/common/assocs/err.D_SYNTAX.errassign.d mode=0444
411 file path=opt/SUNWdtrt/tst/common/assocs/err.tupoflow.d mode=0444
412 file path=opt/SUNWdtrt/tst/common/assocs/tst.cpyarray.d mode=0444
413 file path=opt/SUNWdtrt/tst/common/assocs/tst.diffprofile.d mode=0444
414 file path=opt/SUNWdtrt/tst/common/assocs/tst.initialize.d mode=0444
415 file path=opt/SUNWdtrt/tst/common/assocs/tst.invalidref.d mode=0444
416 file path=opt/SUNWdtrt/tst/common/assocs/tst.misc.d mode=0444
417 file path=opt/SUNWdtrt/tst/common/assocs/tst.orthogonality.d mode=0444
418 file path=opt/SUNWdtrt/tst/common/assocs/tst.this.d mode=0444
419 file path=opt/SUNWdtrt/tst/common/assocs/tst.valassign.d.out mode=0444
420 file path=opt/SUNWdtrt/tst/common/begin/err.D_PDESC_ZERO.begin.d mode=0444
421 file path=opt/SUNWdtrt/tst/common/begin/err.D_PDESC_ZERO.tick.d mode=0444
422 file path=opt/SUNWdtrt/tst/common/begin/tst.begin.d mode=0444
423 file path=opt/SUNWdtrt/tst/common/begin/tst.begin.d.out mode=0444
424 file path=opt/SUNWdtrt/tst/common/begin/tst.multibegin.d mode=0444
425 file path=opt/SUNWdtrt/tst/common/begin/tst.multibegin.d.out mode=0444
426 file \
427     path=opt/SUNWdtrt/tst/common/bitfields/err.D_ADDROF_BITFIELD.BitfieldAddress
428     mode=0444
429 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_DECL_BFCONST.NegBitField.d \
430     mode=0444
431 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_DECL_BFCONST.ZeroBitField.d \
432     mode=0444
433 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_DECL_BFSIZE.ExceedBaseType.d \
434     mode=0444
435 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_DECL_BFSIZE.GreaterThan64.d \
436     mode=0444
437 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_DECL_BFTYPE.badtype.d \
438     mode=0444
439 file path=opt/SUNWdtrt/tst/common/bitfields/err.D_OFFSETOF_BITFIELD.d \
440     mode=0444
441 file \
442     path=opt/SUNWdtrt/tst/common/bitfields/err.D_SIZEOF_BITFIELD.SizeofBitfield.
443     mode=0444
444 file path=opt/SUNWdtrt/tst/common/bitfields/tst.BitFieldPromotion.d mode=0444
445 file path=opt/SUNWdtrt/tst/common/bitfields/tst.SizeofBitField.d mode=0444
446 file path=opt/SUNWdtrt/tst/common/buffering/err.end.d mode=0444
447 file path=opt/SUNWdtrt/tst/common/buffering/err.resize1.d mode=0444
448 file path=opt/SUNWdtrt/tst/common/buffering/err.resize2.d mode=0444
449 file path=opt/SUNWdtrt/tst/common/buffering/err.resize3.d mode=0444
450 file path=opt/SUNWdtrt/tst/common/buffering/err.zerobuf.d mode=0444
451 file path=opt/SUNWdtrt/tst/common/buffering/tst.alignring.d mode=0444
452 file path=opt/SUNWdtrt/tst/common/buffering/tst.cputime.ksh mode=0444
453 file path=opt/SUNWdtrt/tst/common/buffering/tst.dynvarsize.d mode=0444
454 file path=opt/SUNWdtrt/tst/common/buffering/tst.fill1.d mode=0444
455 file path=opt/SUNWdtrt/tst/common/buffering/tst.fill1.d.out mode=0444
```



```
456 file path=opt/SUNWdtrt/tst/common/buffering/tst.resize1.d mode=0444
457 file path=opt/SUNWdtrt/tst/common/buffering/tst.resize2.d mode=0444
458 file path=opt/SUNWdtrt/tst/common/buffering/tst.resize3.d mode=0444
459 file path=opt/SUNWdtrt/tst/common/buffering/tst.ring1.d mode=0444
460 file path=opt/SUNWdtrt/tst/common/buffering/tst.ring2.d mode=0444
461 file path=opt/SUNWdtrt/tst/common/buffering/tst.ring2.d.out mode=0444
462 file path=opt/SUNWdtrt/tst/common/buffering/tst.ring3.d mode=0444
463 file path=opt/SUNWdtrt/tst/common/buffering/tst.ring3.d.out mode=0444
464 file path=opt/SUNWdtrt/tst/common/buffering/tst.smallring.d mode=0444
465 file path=opt/SUNWdtrt/tst/common/buffering/tst.switch1.d mode=0444
466 file path=opt/SUNWdtrt/tst/common/buffering/tst.switch1.d.out mode=0444
467 file path=opt/SUNWdtrt/tst/common/builtinvar/err.D_XLATE_NOCONV.cpuusage.d \
468     mode=0444
469 file path=opt/SUNWdtrt/tst/common/builtinvar/err.D_XLATE_NOCONV.nice.d \
470     mode=0444
471 file path=opt/SUNWdtrt/tst/common/builtinvar/err.D_XLATE_NOCONV.priority.d \
472     mode=0444
473 file path=opt/SUNWdtrt/tst/common/builtinvar/err.D_XLATE_NOCONV.prsize.d \
474     mode=0444
475 file path=opt/SUNWdtrt/tst/common/builtinvar/err.D_XLATE_NOCONV.rssize.d \
476     mode=0444
477 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.arg0.d mode=0444
478 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.arg0clause.d mode=0444
479 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.arg1.d mode=0444
480 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.arg1to8.d mode=0444
481 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.arg1to8clause.d mode=0444
482 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.caller.d mode=0444
483 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.caller1.d mode=0444
484 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.epid.d mode=0444
485 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.epid1.d mode=0444
486 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.errno.d mode=0444
487 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.errno1.d mode=0444
488 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.execname.d mode=0444
489 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.hpriority.d mode=0444
490 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.id.d mode=0444
491 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.id1.d mode=0444
492 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.ipl.d mode=0444
493 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.ipl1.d mode=0444
494 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.lwpsinfo.d mode=0444
495 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.lwpsinfo1.d mode=0444
496 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.pid.d mode=0444
497 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.pid1.d mode=0444
498 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.psinfo.d mode=0444
499 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.psinfo1.d mode=0444
500 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.tid.d mode=0444
501 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.tid1.d mode=0444
502 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.timestamp.d mode=0444
503 file path=opt/SUNWdtrt/tst/common/builtinvar/tst.vtimestamp.d mode=0444
504 file path=opt/SUNWdtrt/tst/common/cg/err.D_NOREG.noreg.d mode=0444
505 file path=opt/SUNWdtrt/tst/common/cg/err.baddif.d mode=0444
506 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.aggfun.d mode=0444
507 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.aggtup.d mode=0444
508 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.arrtup.d mode=0444
509 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.body.d mode=0444
510 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.both.d mode=0444
511 file path=opt/SUNWdtrt/tst/common/clauses/err.D_IDENT_UNDEF.pred.d mode=0444
512 file path=opt/SUNWdtrt/tst/common/clauses/tst.nopred.d mode=0444
513 file path=opt/SUNWdtrt/tst/common/clauses/tst.pred.d mode=0444
514 file path=opt/SUNWdtrt/tst/common/clauses/tst.predfirst.d mode=0444
515 file path=opt/SUNWdtrt/tst/common/clauses/tst.predlast.d mode=0444
516 file path=opt/SUNWdtrt/tst/common/cpc/err.D_PDESC_ZERO.lowfrequency.d \
517     mode=0444
518 file path=opt/SUNWdtrt/tst/common/cpc/err.D_PDESC_ZERO.malformedoverflow.d \
519     mode=0444
520 file path=opt/SUNWdtrt/tst/common/cpc/err.D_PDESC_ZERO.nonexistentevent.d \
521     mode=0444
```

```
file path=opt/SUNWdtrt/tst/common/cpc/err.cpcvscpustatpart1.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/err.cpcvscpustatpart2.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/err.cputrackfailtostart.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/err.cputracketerminates.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/err.toomanyenablings.d mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/tst.allcpus.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/tst.genericevent.d mode=0444
file path=opt/SUNWdtrt/tst/common/cpc/tst.platformevent.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_LOCASSC.NonLocalAssoc.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_LONGINT.LongStruct.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_PARMCLASS.BadStorageClass.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_PROTO_NAME.VoidName.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_PROTO_TYPE.Dyn.d mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_PROTO_VARARGS.VarLenArgs.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_PROTO_VOID.NonSoleVoid.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_SIGNINT.UnsignedStruct.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/err.D_DECL_VOIDATTR.ShortVoidDecl.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/decls/tst.arrays.d mode=0444
file path=opt/SUNWdtrt/tst/common/decls/tst.basics.d mode=0444
file path=opt/SUNWdtrt/tst/common/decls/tst.funcs.d mode=0444
file path=opt/SUNWdtrt/tst/common/decls/tst.pointers.d mode=0444
file path=opt/SUNWdtrt/tst/common/decls/tst.varargsfuncs.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_AGGREGATION.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_DBLERROR.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_DYNAMIC.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_PRINCIPAL.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_PRINCIPAL.end.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_SPEC.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_SPECUNAVAIL.d mode=0444
file path=opt/SUNWdtrt/tst/common/drops/drp.DTRACEDROP_STKSTROVERFLOW.d \
    mode=0444
file \
    path=opt/SUNWdtrt/tst/common/dtraceUtil/err.D_PDESC_ZERO.InvalidDescription1
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.AddSearchPath.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.BufsizeGiga.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.BufsizeKilo.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.BufsizeMega.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.BufsizeTera.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DataModel32.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DataModel64.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DefineNameWithCPP.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DefineNameWithCPP.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithFunction.d.ksh \
    mode=0444
file \
    path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithFunction.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithID.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithID.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithModule.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithModule.d.ksh.out \
```



```
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithName.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithName.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithProvider.d.ksh \
    mode=0444
file \
    path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithProvider.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.DestructWithoutW.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ELFGenerationOut.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ELFGenerationWithO.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ExitStatus1.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ExitStatus2.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ExtraneousProbeIds.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidFuncName1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidFuncName2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidId1.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidId2.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidId3.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidModule1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidModule2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidModule3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidModule4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidProbeIdentifier.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidProvider1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidProvider2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidProvider3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidProvider4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc5.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc6.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc7.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc8.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceFunc9.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID2.d.ksh \
```

```
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID5.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID6.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceID7.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule5.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule6.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule7.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceModule8.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName5.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName6.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName7.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName8.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceName9.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceProvider1.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceProvider2.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceProvider3.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceProvider4.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.InvalidTraceProvider5.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.MultipleInvalidProbeId.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.PreprocessorStatement.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.QuietMode.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.QuietMode.d.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.TestCompile.d.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.TestCompile.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.UnDefineNameWithCPP.d.ksh \
    mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroFunctionProbes.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroFunctionProbes.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroModuleProbes.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroModuleProbes.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroNameProbes.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroNameProbes.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroProbeIdentfier.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroProbesWithoutZ.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroProviderProbes.d.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/dtraceUtil/tst.ZeroProviderProbes.d.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/end/err.D_IDENT_UNDEF.timespent.d mode=0444
file path=opt/SUNWdtrt/tst/common/end/tst.end.d mode=0444
file path=opt/SUNWdtrt/tst/common/end/tst.endwithoutbegin.d mode=0444
file path=opt/SUNWdtrt/tst/common/end/tst.multibeginend.d mode=0444
file path=opt/SUNWdtrt/tst/common/end/tst.multiend.d mode=0444
file path=opt/SUNWdtrt/tst/common/enum/err.D_DECL_IDRED.EnumSameName.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/enum/err.D_UNKNOWN.RepeatIdentifiers.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/enum/tst.EnumEquality.d mode=0444
file path=opt/SUNWdtrt/tst/common/enum/tst.EnumSameValue.d mode=0444
file path=opt/SUNWdtrt/tst/common/enum/tst.EnumValAssign.d mode=0444
file path=opt/SUNWdtrt/tst/common/env/err.D_PRAGMA_OPTSET.setfromscript.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/env/err.D_PRAGMA_OPTSET.unsetfromscript.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.ld_nolazyload.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.ld_nolazyload.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.setenv1.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.setenv1.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.setenv2.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.setenv2.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.unsetenv1.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.unsetenv1.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.unsetenv2.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/env/tst.unsetenv2.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/error/tst.DTRACEFLT_BADADDR.d mode=0444
file path=opt/SUNWdtrt/tst/common/error/tst.DTRACEFLT_DIVZERO.d mode=0444
file path=opt/SUNWdtrt/tst/common/error/tst.DTRACEFLT_UNKNOWN.d mode=0444
file path=opt/SUNWdtrt/tst/common/error/tst.error.d mode=0444
file path=opt/SUNWdtrt/tst/common/error/tst.errorend.d mode=0444
file path=opt/SUNWdtrt/tst/common/exit/err.D_PROTO_LEN.noarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/exit/err.exitarg1.d mode=0444
file path=opt/SUNWdtrt/tst/common/exit/tst.basic1.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/err.D_PDESC_ZERO.notreturn.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.basic.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.functionentry.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.functionreturnvalue.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.ioctlargs.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.offset.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.offsetzero.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.return.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.return0.d mode=0444
file path=opt/SUNWdtrt/tst/common/fbtprovider/tst.tailcall.d mode=0444
```

```
file path=opt/SUNWdtrt/tst/common/funcs/err.D_FUNC_UNDEF.progenyofbad1.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_OP_VFPTR.badop.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.chillbadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.copyoutbadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.mobadarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.raisebadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.tolower.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_ARG.toupper.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.allocanoarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.badbreakpoint.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.chilltoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.chilltoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.copyoutstrbadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.copyoutstrtoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.copyouttoofew.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.copyouttoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.motoofew.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.motoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.mtabadarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.mtatoofew.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.mtatoomany.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.panicbadarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.progenyofbad2.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.stopbadarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.tolower.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.tolowertoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.toupper.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_PROTO_LEN.touppertoomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_STRINGOF_TYPE.badstringof.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.D_VAR_UNDEF.badvar.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badalloca.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badalloca2.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy1.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy2.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy3.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy4.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy5.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badbcopy6.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.badchill.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.chillbadarg.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.copyout.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.copyoutbadaddr.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.copyoutstrbadaddr.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.inet_ntoa6badaddr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.inet_ntoabadaddr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.inet_ntopbadaddr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/err.inet_ntopbadarg.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.badfreopen.ksh mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/funcs/tst.basename.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.basename.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.bcopy.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.chill.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.cleanpath.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.cleanpath.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.copyin.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.copyinto.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.ddi_pathname.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.default.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.freopen.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.ftruncate.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.ftruncate.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.hton.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.index.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.index.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntoa.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntoa.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntoa6.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntoa6.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntop.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.inet_ntop.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.lltostr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.lltostr.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.lltostrbase.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.lltostrbase.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.mutex_owned.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.mutex_owner.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.mutex_type_adaptive.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.progenyof.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.rand.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strchr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strchr.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strjoin.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strjoin.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strstr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strstr.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strtok.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strtok.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.strtok_null.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.substr.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.substr.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.substrminate.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.substrminate.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.system.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.system.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.tolower.d mode=0444
file path=opt/SUNWdtrt/tst/common/funcs/tst.toupper.d mode=0444
file path=opt/SUNWdtrt/tst/common/grammar/err.D_ADDROF_LVAL.d mode=0444
file path=opt/SUNWdtrt/tst/common/grammar/err.D_EMPTY.empty.d mode=0444
file path=opt/SUNWdtrt/tst/common/grammar/tst.clauses.d mode=0444
file path=opt/SUNWdtrt/tst/common/grammar/tst.stmts.d mode=0444
file path=opt/SUNWdtrt/tst/common/include/tst.includefirst.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_DECL_IDRED.redef1.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_DECL_IDRED.redef2.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_IDENT_UNDEF.recur.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_OP_INCOMPAT.baddef1.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_OP_INCOMPAT.baddef2.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/err.D_OP_INCOMPAT.badxlate.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineDataAssign.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineExpression.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineKinds.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineKinds.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineTypedef.d mode=0444
file path=opt/SUNWdtrt/tst/common/inline/tst.InlineWritableAssign.d mode=0444
```

```
file path=opt/SUNWdtrt/tst/common/io/tst.fds.d mode=0444
file path=opt/SUNWdtrt/tst/common/io/tst.fds.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/io/tst.fds.exe mode=0555
file path=opt/SUNWdtrt/tst/common/ip/get.ipv4remote.pl mode=0555
file path=opt/SUNWdtrt/tst/common/ip/get.ipv6remote.pl mode=0555
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localicmp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localicmp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localtcp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localtcp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localudp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4localudp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remoteicmp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remoteicmp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remotetcp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remotetcp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remoteudp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv4remoteudp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv6localicmp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv6localicmp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv6remoteicmp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.ipv6remoteicmp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.localtcpstate.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.localtcpstate.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.remotetcpstate.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/ip/tst.remotetcpstate.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/test.jar
file path=opt/SUNWdtrt/tst/common/java_api/tst.Abort.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Abort.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Bean.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Bean.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Close.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Close.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Drop.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Drop.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Enable.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.Enable.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.FunctionLookup.exe mode=0555
file path=opt/SUNWdtrt/tst/common/java_api/tst.FunctionLookup.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.FunctionLookup.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.GetAggregate.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.MaxConsumers.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.MaxConsumers.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.MultiAggPrinta.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.MultiAggPrinta.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.ProbeData.exe mode=0555
file path=opt/SUNWdtrt/tst/common/java_api/tst.ProbeData.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.ProbeData.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.ProbeDescription.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.ProbeDescription.ksh.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.StateMachine.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.StateMachine.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.StopLock.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.StopLock.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.printa.d mode=0444
file path=opt/SUNWdtrt/tst/common/java_api/tst.printa.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.general.d mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.general.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.strsize.d mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.strsize.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.usdt.d mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.usdt.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/json/tst.usdt.exe mode=0555
file path=opt/SUNWdtrt/tst/common/lexer/err.D_CHR_NL.char.d mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/lexer/err.D_CHR_NULL.char.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_INT_DIGIT.InvalidDigit.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_INT_OFLOW.BigInt.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_STR_NL.string.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.brace1.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.brace2.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.brack1.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.brack2.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.brack3.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.paren1.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.paren2.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/err.D_SYNTAX.paren3.d mode=0444
file path=opt/SUNWdtrt/tst/common/lexer/tst.D_MACRO_OFLOW.ParIntOvflow.d.ksh \
    mode=0444
file \
    path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTOREVEN.nodivide.d
    mode=0444
file \
    path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTOREVEN.notfactor.d
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTORMATCH.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTORNSTEPS.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTORSMALL.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTORTYPE.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_FACTORVAL.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_HIGHMATCH.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_HIGHTYPE.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_HIGHVAL.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_LOWMATCH.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_LOWTYPE.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_LOWVAL.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_MAGRANGE.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_MAGTOOBIG.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_NSTEPMATCH.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_NSTEPTYPE.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/err.D_LLQUANT_NSTEPVAL.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.bases.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.bases.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.basic.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.basic.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.negorder.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.negorder.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.negvalue.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.negvalue.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.normal.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.normal.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.range.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.range.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.steps.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.steps.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.trunc.d mode=0444
file path=opt/SUNWdtrt/tst/common/llquantize/tst.trunc.d.out mode=0444
```

```
1050 file path=opt/SUNWdtrt/tst/common/mdb/tst.dtracedcmd.ksh mode=0444
1051 file path=opt/SUNWdtrt/tst/common/mib/tst.icmp.ksh mode=0444
1052 file path=opt/SUNWdtrt/tst/common/mib/tst.tcp.ksh mode=0444
1053 file path=opt/SUNWdtrt/tst/common/mib/tst.udp.ksh mode=0444
1054 file path=opt/SUNWdtrt/tst/common/misc/err.D_PRAGMA_OPTSET.d mode=0444
1055 file path=opt/SUNWdtrt/tst/common/misc/tst.badopt.d mode=0444
1056 file path=opt/SUNWdtrt/tst/common/misc/tst.boolopt.d mode=0444
1057 file path=opt/SUNWdtrt/tst/common/misc/tst.boolopt.d.out mode=0444
1058 file path=opt/SUNWdtrt/tst/common/misc/tst.dynopt.d mode=0444
1059 file path=opt/SUNWdtrt/tst/common/misc/tst.dynopt.d.out mode=0444
1060 file path=opt/SUNWdtrt/tst/common/misc/tst.enablerace.ksh mode=0444
1061 file path=opt/SUNWdtrt/tst/common/misc/tst.haslam.d mode=0444
1062 file path=opt/SUNWdtrt/tst/common/misc/tst.include.ksh mode=0444
1063 file path=opt/SUNWdtrt/tst/common/misc/tst.macroglob.ksh mode=0444
1064 file path=opt/SUNWdtrt/tst/common/misc/tst.macroglob.ksh.out mode=0444
1065 file path=opt/SUNWdtrt/tst/common/misc/tst.roch.d mode=0444
1066 file path=opt/SUNWdtrt/tst/common/misc/tst.schrock.ksh mode=0444
1067 file path=opt/SUNWdtrt/tst/common/multiaggs/err.D_PRINTA_AGGKEY.d mode=0444
1068 file path=opt/SUNWdtrt/tst/common/multiaggs/err.D_PRINTA_AGGPROTO.d mode=0444
1069 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.many.d mode=0444
1070 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.many.d.out mode=0444
1071 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.same.d mode=0444
1072 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.same.d.out mode=0444
1073 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.sort.d mode=0444
1074 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.sort.d.out mode=0444
1075 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.sortpos.d mode=0444
1076 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.sortpos.d.out mode=0444
1077 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.tuplecompat.d mode=0444
1078 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.tuplecompat.d.out mode=0444
1079 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero.d mode=0444
1080 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero.d.out mode=0444
1081 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero2.d mode=0444
1082 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero2.d.out mode=0444
1083 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero3.d mode=0444
1084 file path=opt/SUNWdtrt/tst/common/multiaggs/tst.zero3.d.out mode=0444
1085 file path=opt/SUNWdtrt/tst/common/nfs/tst.call.d mode=0444
1086 file path=opt/SUNWdtrt/tst/common/nfs/tst.call.exe mode=0555
1087 file path=opt/SUNWdtrt/tst/common/nfs/tst.call3.d mode=0444
1088 file path=opt/SUNWdtrt/tst/common/nfs/tst.call3.exe mode=0555
1089 file path=opt/SUNWdtrt/tst/common/offsetof/err.D_OFFSETOF_BITFIELD.bitfield.d \
1090     mode=0444
1091 file path=opt/SUNWdtrt/tst/common/offsetof/err.D_OFFSETOF_TYPE.badtype.d \
1092     mode=0444
1093 file path=opt/SUNWdtrt/tst/common/offsetof/err.D_OFFSETOF_TYPE.notsou.d \
1094     mode=0444
1095 file path=opt/SUNWdtrt/tst/common/offsetof/err.D_UNKNOWN.OffsetofNULL.d \
1096     mode=0444
1097 file path=opt/SUNWdtrt/tst/common/offsetof/err.D_UNKNOWN.badmemb.d mode=0444
1098 file path=opt/SUNWdtrt/tst/common/offsetof/tst.OffsetofAlias.d mode=0444
1099 file path=opt/SUNWdtrt/tst/common/offsetof/tst.OffsetofArith.d mode=0444
1100 file path=opt/SUNWdtrt/tst/common/offsetof/tst.OffsetofUnion.d mode=0444
1101 file path=opt/SUNWdtrt/tst/common/offsetof/tst.struct.d mode=0444
1102 file path=opt/SUNWdtrt/tst/common/offsetof/tst.struct.d.out mode=0444
1103 file path=opt/SUNWdtrt/tst/common/offsetof/tst.union.d mode=0444
1104 file path=opt/SUNWdtrt/tst/common/offsetof/tst.union.d.out mode=0444
1105 file path=opt/SUNWdtrt/tst/common/operators/tst.ternary.d mode=0444
1106 file path=opt/SUNWdtrt/tst/common/operators/tst.ternary.d.out mode=0444
1107 file path=opt/SUNWdtrt/tst/common/pid/err.D_PDESC_ZERO.badlib.d mode=0444
1108 file path=opt/SUNWdtrt/tst/common/pid/err.D_PDESC_ZERO.badlib.exe mode=0555
1109 file path=opt/SUNWdtrt/tst/common/pid/err.D_PDESC_ZERO.badproc1.d mode=0444
1110 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_BADPID.badproc2.d mode=0444
1111 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_CREATEFAIL.many.d mode=0444
1112 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_CREATEFAIL.many.exe mode=0555
1113 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_FUNC.badfunc.d mode=0444
1114 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_FUNC.badfunc.exe mode=0555
1115 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_LIB.libdash.d mode=0444
```



```
1116 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_LIB.libdash.exe mode=0555
1117 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.alldash.d mode=0444
1118 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.alldash.exe mode=0555
1119 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.badname.d mode=0444
1120 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.badname.exe mode=0555
1121 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.globdash.d mode=0444
1122 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_NAME.globdash.exe mode=0555
1123 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_OFF.toobig.d mode=0444
1124 file path=opt/SUNWdtrt/tst/common/pid/err.D_PROC_OFF.toobig.exe mode=0555
1125 file path=opt/SUNWdtrt/tst/common/pid/tst.addprobes.ksh mode=0444
1126 file path=opt/SUNWdtrt/tst/common/pid/tst.args1.d mode=0444
1127 file path=opt/SUNWdtrt/tst/common/pid/tst.args1.exe mode=0555
1128 file path=opt/SUNWdtrt/tst/common/pid/tst.coverage.d mode=0444
1129 file path=opt/SUNWdtrt/tst/common/pid/tst.coverage.exe mode=0555
1130 file path=opt/SUNWdtrt/tst/common/pid/tst.emptystack.d mode=0444
1131 file path=opt/SUNWdtrt/tst/common/pid/tst.emptystack.d.out mode=0444
1132 file path=opt/SUNWdtrt/tst/common/pid/tst.emptystack.exe mode=0555
1133 file path=opt/SUNWdtrt/tst/common/pid/tst.float.d mode=0444
1134 file path=opt/SUNWdtrt/tst/common/pid/tst.float.exe mode=0555
1135 file path=opt/SUNWdtrt/tst/common/pid/tst.fork.d mode=0444
1136 file path=opt/SUNWdtrt/tst/common/pid/tst.fork.exe mode=0555
1137 file path=opt/SUNWdtrt/tst/common/pid/tst.gcc.d mode=0444
1138 file path=opt/SUNWdtrt/tst/common/pid/tst.gcc.exe mode=0555
1139 file path=opt/SUNWdtrt/tst/common/pid/tst.killonerror.ksh mode=0444
1140 file path=opt/SUNWdtrt/tst/common/pid/tst.main.ksh mode=0444
1141 file path=opt/SUNWdtrt/tst/common/pid/tst.manypids.ksh mode=0444
1142 file path=opt/SUNWdtrt/tst/common/pid/tst.newprobes.ksh mode=0444
1143 file path=opt/SUNWdtrt/tst/common/pid/tst.newprobes.ksh.out mode=0444
1144 file path=opt/SUNWdtrt/tst/common/pid/tst.probemod.ksh mode=0444
1145 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex1.ksh mode=0444
1146 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex2.ksh mode=0444
1147 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex2.ksh.out mode=0444
1148 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex3.ksh mode=0444
1149 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex3.ksh.out mode=0444
1150 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex4.ksh mode=0444
1151 file path=opt/SUNWdtrt/tst/common/pid/tst.provregex4.ksh.out mode=0444
1152 file path=opt/SUNWdtrt/tst/common/pid/tst.ret1.d mode=0444
1153 file path=opt/SUNWdtrt/tst/common/pid/tst.ret1.exe mode=0555
1154 file path=opt/SUNWdtrt/tst/common/pid/tst.ret2.d mode=0444
1155 file path=opt/SUNWdtrt/tst/common/pid/tst.ret2.exe mode=0555
1156 file path=opt/SUNWdtrt/tst/common/pid/tst.utf8probefunc.ksh mode=0444
1157 file path=opt/SUNWdtrt/tst/common/pid/tst.utf8probefunc.ksh.out mode=0444
1158 file path=opt/SUNWdtrt/tst/common/pid/tst.utf8probemod.ksh mode=0444
1159 file path=opt/SUNWdtrt/tst/common/pid/tst.utf8probemod.ksh.out mode=0444
1160 file path=opt/SUNWdtrt/tst/common/pid/tst.vfork.d mode=0444
1161 file path=opt/SUNWdtrt/tst/common/pid/tst.vfork.exe mode=0555
1162 file path=opt/SUNWdtrt/tst/common/pid/tst.weak1.d mode=0444
1163 file path=opt/SUNWdtrt/tst/common/pid/tst.weak1.exe mode=0555
1164 file path=opt/SUNWdtrt/tst/common/pid/tst.weak2.d mode=0444
1165 file path=opt/SUNWdtrt/tst/common/pid/tst.weak2.exe mode=0555
1166 file path=opt/SUNWdtrt/tst/common/plockstat/tst.available.d mode=0444
1167 file path=opt/SUNWdtrt/tst/common/plockstat/tst.available.exe mode=0555
1168 file path=opt/SUNWdtrt/tst/common/plockstat/tst.libmap.d mode=0444
1169 file path=opt/SUNWdtrt/tst/common/plockstat/tst.libmap.exe mode=0555
1170 file path=opt/SUNWdtrt/tst/common/pointers/err.BadAlign.d mode=0444
1171 file path=opt/SUNWdtrt/tst/common/pointers/err.D_ADDROF_VAR.ArrayVar.d \
1172     mode=0444
1173 file path=opt/SUNWdtrt/tst/common/pointers/err.D_ADDROF_VAR.DynamicVar.d \
1174     mode=0444
1175 file path=opt/SUNWdtrt/tst/common/pointers/err.D_ADDROF_VAR.agg.d mode=0444
1176 file path=opt/SUNWdtrt/tst/common/pointers/err.D_DEREF_NONPTR.noptr.d \
1177     mode=0444
1178 file path=opt/SUNWdtrt/tst/common/pointers/err.D_DEREF_VOID.VoidPointerDeref.d \
1179     mode=0444
1180 file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_ARRFUN.ArrayAssignment.d \
1181     mode=0444
```

```
file \
    path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_INCOMPAT.VoidPointerArith.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_LVAL.AddressChange.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_PTR.NonPointerAccess.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_PTR.badpointer.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_SOU.BadPointerAccess.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.D_OP_SOU.badpointer.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.InvalidAddress1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.InvalidAddress2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.InvalidAddress3.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.InvalidAddress4.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/err.InvalidAddress5.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.ArrayPointer1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.ArrayPointer2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.ArrayPointer3.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.GlobalVar.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.IntegerArithmetic1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.PointerArithmetic1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.PointerArithmetic2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.PointerArithmetic3.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.PointerAssignment.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.ValidPointer1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.ValidPointer2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.VoidCast.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.assigncast1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.assigncast2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.basic1.d mode=0444
file path=opt/SUNWdtrt/tst/common/pointers/tst.basic2.d mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.D_PRAGERR.d mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.D_PRAGMA_DEPEND.main.d mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.D_PRAGMA_INVAL.d mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.D_PRAGMA_MALFORM.d mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.D_PRAGMA_UNUSED.UnusedPragma.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.circlibdep.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/err.invalidlibdep.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.libchain.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.libdep.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.libdepfullyconnected.ksh \
    mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.libdepsepdir.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.temporal.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.temporal2.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/pragma/tst.temporal3.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/err.D_PRED_SCALAR.NonScalarPred.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/err.D_SYNTAX.invalid.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/err.D_SYNTAX.operr.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/tst.argsnotcached.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/tst.basics.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/tst.basics.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/tst.complex.d mode=0444
file path=opt/SUNWdtrt/tst/common/predicates/tst.complex.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.D_IDENT_UNDEF.afterprobe.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.D_PRAGCTL_INVAL.tabdefine.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.D_SYNTAX.withoutpound.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.defincomp.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.ifdefelsenotendif.d \
    mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/preprocessor/err.ifdefincomp.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.ifdefnotendif.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.incompelse.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/err.mulelse.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifdef.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifdef.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifndef.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifndef.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifnotdef.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.ifnotdef.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicaland.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicaland.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicalandor.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicalandor.d.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicalor.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.logicalor.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.muland.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.muland.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.mulor.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.mulor.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.precondi.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.precondi.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.predicatedeclare.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexp.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexp.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpelse.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpelse.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpif.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpif.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpifelse.d mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.preexpifelse.d.out \
    mode=0444
file path=opt/SUNWdtrt/tst/common/preprocessor/tst.withinprobe.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/err.D_PRINT_AGG.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/err.D_PRINT_VOID.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/err.D_PROTO_LEN.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.array.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.array.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.bitfield.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.bitfield.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.dyn.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.enum.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.enum.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.primitive.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.primitive.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.struct.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.struct.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.xlate.d mode=0444
file path=opt/SUNWdtrt/tst/common/print/tst.xlate.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTA_AGGARG.badagg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTA_AGGARG.badfmt.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTA_AGGARG.badval.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTA_PROTO.bad.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTF_ARG_TYPE.jstack.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTF_ARG_TYPE.stack.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/err.D_PRINTF_ARG_TYPE.ustack.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.basics.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.basics.d.out mode=0444
```

```
file path=opt/SUNWdtrt/tst/common/printa/tst.def.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.def.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.dynwidth.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.dynwidth.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.fmt.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.fmt.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.largeusersym.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.many.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.manyval.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.manyval.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.stack.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.tuple.d mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.tuple.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.walltimestamp.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/printa/tst.walltimestamp.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_AGG_CONV.aggfmt.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_EXTRA.toomany.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_EXTRA.widths.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_FMT.badfmt.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_PROTO.novalue.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_TYPE.aggarg.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_ARG_TYPE.recursive.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_DYN_PROTO.noprec.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_DYN_PROTO.nowidth.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_DYN_TYPE.badprec.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PRINTF_DYN_TYPE.badwidth.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_PROTO_LEN.toofew.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_SYNTAX.badconv1.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_SYNTAX.badconv2.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/err.D_SYNTAX.badconv3.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.basics.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.basics.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.flags.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.flags.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.hello.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.hello.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.ints.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.ints.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.precs.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.precs.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.print-f.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.print-f.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printT.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printT.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printY.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printY.ksh.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printcont.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printcont.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printeE.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printeE.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printgG.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.printgG.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.rawfmt.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.rawfmt.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.signs.d mode=0444
```



```
file path=opt/SUNWdtrt/tst/common/printf/tst.signs.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.str.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.str.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.sym.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.sym.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.uints.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.uints.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.widths.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.widths.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.widths1.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.wp.d mode=0444
file path=opt/SUNWdtrt/tst/common/printf/tst.wp.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.fds.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.func_access.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.getf.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.noprivdrop.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.noprivrestrict.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.op_access.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.procpriv.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.providers.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.tick.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/privs/tst.unpriv_funcs.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_PDESC_ZERO.probeqtn.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_PDESC_ZERO.probestar.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_PDESC_ZERO.tickstar.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.assign.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.declare.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.declarein.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.lbraces.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.probespec.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.rbraces.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/err.D_SYNTAX.recdec.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.basic1.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.check.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.declare.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.declareafter.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.emptyprobe.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.pragma.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.pragmaaftertab.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.pragmainside.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.pragmaoutside.d mode=0444
file path=opt/SUNWdtrt/tst/common/probes/tst.probestar.d mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.create.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.discard.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.exec.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.execfail.ENOENT.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.execfail.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.exitcore.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.exitexit.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.exitkilled.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.signal.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.sigwait.d mode=0444
file path=opt/SUNWdtrt/tst/common/proc/tst.sigwait.exe mode=0555
file path=opt/SUNWdtrt/tst/common/proc/tst.startexit.ksh mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/err.D_PDESC_ZERO.profile.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/err.D_PDESC_ZEROonens.d mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/err.D_PDESC_ZEROonensec.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/err.D_PDESC_ZEROoneus.d mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/err.D_PDESC_ZEROoneusec.d \
    mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/tst.argtest.d mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/tst.argtest.d.out mode=0444
file path=opt/SUNWdtrt/tst/common/profile-n/tst.basic.d mode=0444
```

```
1446 file path=opt/SUNWdtrt/tst/common/profile-n/tst.basic.d.out mode=0444
1447 file path=opt/SUNWdtrt/tst/common/profile-n/tst.func.ksh mode=0444
1448 file path=opt/SUNWdtrt/tst/common/profile-n/tst.mod.ksh mode=0444
1449 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilehz.d mode=0444
1450 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilehz.d.out mode=0444
1451 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilems.d mode=0444
1452 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilems.d.out mode=0444
1453 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilemsec.d mode=0444
1454 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilemsec.d.out mode=0444
1455 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilenhz.d mode=0444
1456 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilenhz.d.out mode=0444
1457 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilens.d mode=0444
1458 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilens.d.out mode=0444
1459 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilensec.d mode=0444
1460 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilensec.d.out mode=0444
1461 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profiles.d mode=0444
1462 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profiles.d.out mode=0444
1463 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilesec.d mode=0444
1464 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profilesec.d.out mode=0444
1465 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profileus.d mode=0444
1466 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profileus.d.out mode=0444
1467 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profileusec.d mode=0444
1468 file path=opt/SUNWdtrt/tst/common/profile-n/tst.profileusec.d.out mode=0444
1469 file path=opt/SUNWdtrt/tst/common/profile-n/tst.sym.ksh mode=0444
1470 file path=opt/SUNWdtrt/tst/common/profile-n/tst.ufunc.ksh mode=0444
1471 file path=opt/SUNWdtrt/tst/common/profile-n/tst.ufuncsort.exe mode=0555
1472 file path=opt/SUNWdtrt/tst/common/profile-n/tst.ufuncsort.ksh mode=0444
1473 file path=opt/SUNWdtrt/tst/common/profile-n/tst.ufuncsort.ksh.out mode=0444
1474 file path=opt/SUNWdtrt/tst/common/profile-n/tst.umod.ksh mode=0444
1475 file path=opt/SUNWdtrt/tst/common/profile-n/tst.usym.ksh mode=0444
1476 file path=opt/SUNWdtrt/tst/common/providers/err.D_PDESC_INVAL.wrongdec4.d \
1477     mode=0444
1478 file path=opt/SUNWdtrt/tst/common/providers/err.D_PDESC_ZERO.nonprofile.d \
1479     mode=0444
1480 file path=opt/SUNWdtrt/tst/common/providers/err.D_PDESC_ZERO.wrongdec1.d \
1481     mode=0444
1482 file path=opt/SUNWdtrt/tst/common/providers/err.D_PDESC_ZERO.wrongdec2.d \
1483     mode=0444
1484 file path=opt/SUNWdtrt/tst/common/providers/err.D_PDESC_ZERO.wrongdec3.d \
1485     mode=0444
1486 file path=opt/SUNWdtrt/tst/common/providers/tst.basics.d mode=0444
1487 file path=opt/SUNWdtrt/tst/common/providers/tst.basics.d.out mode=0444
1488 file path=opt/SUNWdtrt/tst/common/providers/tst.beginexit.d mode=0444
1489 file path=opt/SUNWdtrt/tst/common/providers/tst.beginprof.d mode=0444
1490 file path=opt/SUNWdtrt/tst/common/providers/tst.beginprof.d.out mode=0444
1491 file path=opt/SUNWdtrt/tst/common/providers/tst.probattrs.d mode=0444
1492 file path=opt/SUNWdtrt/tst/common/providers/tst.probattrs.d.out mode=0444
1493 file path=opt/SUNWdtrt/tst/common/providers/tst.probefunc.d mode=0444
1494 file path=opt/SUNWdtrt/tst/common/providers/tst.probefunc.d.out mode=0444
1495 file path=opt/SUNWdtrt/tst/common/providers/tst.probemod.d mode=0444
1496 file path=opt/SUNWdtrt/tst/common/providers/tst.probemod.d.out mode=0444
1497 file path=opt/SUNWdtrt/tst/common/providers/tst.probename.d mode=0444
1498 file path=opt/SUNWdtrt/tst/common/providers/tst.probename.d.out mode=0444
1499 file path=opt/SUNWdtrt/tst/common/providers/tst.probprov.d mode=0444
1500 file path=opt/SUNWdtrt/tst/common/providers/tst.probprov.d.out mode=0444
1501 file path=opt/SUNWdtrt/tst/common/providers/tst.profend.d mode=0444
1502 file path=opt/SUNWdtrt/tst/common/providers/tst.profend.d.out mode=0444
1503 file path=opt/SUNWdtrt/tst/common/providers/tst.profexit.d mode=0444
1504 file path=opt/SUNWdtrt/tst/common/providers/tst.profexit.d.out mode=0444
1505 file path=opt/SUNWdtrt/tst/common/providers/tst.trace.d mode=0444
1506 file path=opt/SUNWdtrt/tst/common/providers/tst.trace.d.out mode=0444
1507 file path=opt/SUNWdtrt/tst/common/providers/tst.twoprof.d mode=0444
1508 file path=opt/SUNWdtrt/tst/common/providers/tst.twoprof.d.out mode=0444
1509 file path=opt/SUNWdtrt/tst/common/raise/tst.raise1.d mode=0444
1510 file path=opt/SUNWdtrt/tst/common/raise/tst.raise1.exe mode=0555
1511 file path=opt/SUNWdtrt/tst/common/raise/tst.raise2.d mode=0444
```



```
1512 file path=opt/SUNWdtrt/tst/common/raise/tst.raise2.exe mode=0555
1513 file path=opt/SUNWdtrt/tst/common/raise/tst.raise3.d mode=0444
1514 file path=opt/SUNWdtrt/tst/common/raise/tst.raise3.exe mode=0555
1515 file path=opt/SUNWdtrt/tst/common/rates/tst.aggrate.d mode=0444
1516 file path=opt/SUNWdtrt/tst/common/rates/tst.aggrate.d.out mode=0444
1517 file path=opt/SUNWdtrt/tst/common/rates/tst.statusrate.d mode=0444
1518 file path=opt/SUNWdtrt/tst/common/rates/tst.switchrate.d mode=0444
1519 file path=opt/SUNWdtrt/tst/common/rates/tst.switchrate.d.out mode=0444
1520 file path=opt/SUNWdtrt/tst/common/safety/tst.basename.d mode=0444
1521 file path=opt/SUNWdtrt/tst/common/safety/tst.caller.d mode=0444
1522 file path=opt/SUNWdtrt/tst/common/safety/tst.cleanpath.d mode=0444
1523 file path=opt/SUNWdtrt/tst/common/safety/tst.copyin.d mode=0444
1524 file path=opt/SUNWdtrt/tst/common/safety/tst.copyin2.d mode=0444
1525 file path=opt/SUNWdtrt/tst/common/safety/tst.ddi_pathname.d mode=0444
1526 file path=opt/SUNWdtrt/tst/common/safety/tst.dirname.d mode=0444
1527 file path=opt/SUNWdtrt/tst/common/safety/tst.errno.d mode=0444
1528 file path=opt/SUNWdtrt/tst/common/safety/tst.execname.d mode=0444
1529 file path=opt/SUNWdtrt/tst/common/safety/tst.gid.d mode=0444
1530 file path=opt/SUNWdtrt/tst/common/safety/tst.hton.d mode=0444
1531 file path=opt/SUNWdtrt/tst/common/safety/tst.index.d mode=0444
1532 file path=opt/SUNWdtrt/tst/common/safety/tst.msgdsize.d mode=0444
1533 file path=opt/SUNWdtrt/tst/common/safety/tst.msgsize.d mode=0444
1534 file path=opt/SUNWdtrt/tst/common/safety/tst.null.d mode=0444
1535 file path=opt/SUNWdtrt/tst/common/safety/tst.pid.d mode=0444
1536 file path=opt/SUNWdtrt/tst/common/safety/tst.ppid.d mode=0444
1537 file path=opt/SUNWdtrt/tst/common/safety/tst.progenyof.d mode=0444
1538 file path=opt/SUNWdtrt/tst/common/safety/tst.random.d mode=0444
1539 file path=opt/SUNWdtrt/tst/common/safety/tst.rw.d mode=0444
1540 file path=opt/SUNWdtrt/tst/common/safety/tst.shortstr.d mode=0444
1541 file path=opt/SUNWdtrt/tst/common/safety/tst.stack.d mode=0444
1542 file path=opt/SUNWdtrt/tst/common/safety/tst.stackdepth.d mode=0444
1543 file path=opt/SUNWdtrt/tst/common/safety/tst.stddev.d mode=0444
1544 file path=opt/SUNWdtrt/tst/common/safety/tst.strchr.d mode=0444
1545 file path=opt/SUNWdtrt/tst/common/safety/tst.strjoin.d mode=0444
1546 file path=opt/SUNWdtrt/tst/common/safety/tst.strstr.d mode=0444
1547 file path=opt/SUNWdtrt/tst/common/safety/tst.strtok.d mode=0444
1548 file path=opt/SUNWdtrt/tst/common/safety/tst.substr.d mode=0444
1549 file path=opt/SUNWdtrt/tst/common/safety/tst.ucaller.d mode=0444
1550 file path=opt/SUNWdtrt/tst/common/safety/tst.uid.d mode=0444
1551 file path=opt/SUNWdtrt/tst/common/safety/tst.unalign.d mode=0444
1552 file path=opt/SUNWdtrt/tst/common/safety/tst.uregs.d mode=0444
1553 file path=opt/SUNWdtrt/tst/common/safety/tst.ustack.d mode=0444
1554 file path=opt/SUNWdtrt/tst/common/safety/tst.ustackdepth.d mode=0444
1555 file path=opt/SUNWdtrt/tst/common/safety/tst.vahole.d mode=0444
1556 file path=opt/SUNWdtrt/tst/common/safety/tst.violentdeath.ksh mode=0444
1557 file path=opt/SUNWdtrt/tst/common/safety/tst.zonename.d mode=0444
1558 file path=opt/SUNWdtrt/tst/common/scalars/err.D_ARR_LOCAL.thisarray.d \
1559     mode=0444
1560 file path=opt/SUNWdtrt/tst/common/scalars/err.D_DECL_CLASS.selfthis.d \
1561     mode=0444
1562 file path=opt/SUNWdtrt/tst/common/scalars/err.D_DECL_CLASS.thisself.d \
1563     mode=0444
1564 file path=opt/SUNWdtrt/tst/common/scalars/err.D_DECL_IDRED.errval.d mode=0444
1565 file path=opt/SUNWdtrt/tst/common/scalars/err.D_OP_INCOMPAT.dec.err.d \
1566     mode=0444
1567 file path=opt/SUNWdtrt/tst/common/scalars/err.D_OP_INCOMPAT.dupgtype.d \
1568     mode=0444
1569 file path=opt/SUNWdtrt/tst/common/scalars/err.D_OP_INCOMPAT.dupltype.d \
1570     mode=0444
1571 file path=opt/SUNWdtrt/tst/common/scalars/err.D_OP_INCOMPAT.dupttype.d \
1572     mode=0444
1573 file path=opt/SUNWdtrt/tst/common/scalars/err.D_SYNTAX.declare.d mode=0444
1574 file path=opt/SUNWdtrt/tst/common/scalars/tst.basicvar.d mode=0444
1575 file path=opt/SUNWdtrt/tst/common/scalars/tst.basicvar.d.out mode=0444
1576 file path=opt/SUNWdtrt/tst/common/scalars/tst.localvar.d mode=0444
1577 file path=opt/SUNWdtrt/tst/common/scalars/tst.misc.d mode=0444
```

```
1578 file path=opt/SUNWdtrt/tst/common/scalars/tst.self.d mode=0444
1579 file path=opt/SUNWdtrt/tst/common/scalars/tst.selfarray.d mode=0444
1580 file path=opt/SUNWdtrt/tst/common/scalars/tst.selfarray2.d mode=0444
1581 file path=opt/SUNWdtrt/tst/common/scalars/tst.selfthis.d mode=0444
1582 file path=opt/SUNWdtrt/tst/common/scalars/tst.this.d mode=0444
1583 file path=opt/SUNWdtrt/tst/common/scalars/tst.thisself.d mode=0444
1584 file path=opt/SUNWdtrt/tst/common/sched/tst.enqueue.d mode=0444
1585 file path=opt/SUNWdtrt/tst/common/sched/tst.oncpu.d mode=0444
1586 file path=opt/SUNWdtrt/tst/common/sched/tst.stackdepth.d mode=0444
1587 file path=opt/SUNWdtrt/tst/common/scripting/err.D_MACRO_UNDEF.invalidargs.d \
1588     mode=0444
1589 file path=opt/SUNWdtrt/tst/common/scripting/err.D_OP_LVAL.rdonly.d mode=0444
1590 file path=opt/SUNWdtrt/tst/common/scripting/err.D_OP_WRITE.usepidmacro.d \
1591     mode=0444
1592 file path=opt/SUNWdtrt/tst/common/scripting/err.D_SYNTAX.concat.d mode=0444
1593 file path=opt/SUNWdtrt/tst/common/scripting/err.D_SYNTAX.desc.d mode=0444
1594 file path=opt/SUNWdtrt/tst/common/scripting/err.D_SYNTAX.inval.d mode=0444
1595 file path=opt/SUNWdtrt/tst/common/scripting/err.D_SYNTAX.pid.d mode=0444
1596 file path=opt/SUNWdtrt/tst/common/scripting/tst.D_MACRO_UNUSED.overflow.ksh \
1597     mode=0444
1598 file path=opt/SUNWdtrt/tst/common/scripting/tst.arg0.d mode=0444
1599 file path=opt/SUNWdtrt/tst/common/scripting/tst.arguments.ksh mode=0444
1600 file path=opt/SUNWdtrt/tst/common/scripting/tst.assign.d mode=0444
1601 file path=opt/SUNWdtrt/tst/common/scripting/tst.basic.d mode=0444
1602 file path=opt/SUNWdtrt/tst/common/scripting/tst.egid.d mode=0444
1603 file path=opt/SUNWdtrt/tst/common/scripting/tst.egid.ksh mode=0444
1604 file path=opt/SUNWdtrt/tst/common/scripting/tst.euid.d mode=0444
1605 file path=opt/SUNWdtrt/tst/common/scripting/tst.euid.ksh mode=0444
1606 file path=opt/SUNWdtrt/tst/common/scripting/tst.gid.d mode=0444
1607 file path=opt/SUNWdtrt/tst/common/scripting/tst.gid.ksh mode=0444
1608 file path=opt/SUNWdtrt/tst/common/scripting/tst.pgid.d mode=0444
1609 file path=opt/SUNWdtrt/tst/common/scripting/tst.pid.d mode=0444
1610 file path=opt/SUNWdtrt/tst/common/scripting/tst.ppid.d mode=0444
1611 file path=opt/SUNWdtrt/tst/common/scripting/tst.ppid.ksh mode=0444
1612 file path=opt/SUNWdtrt/tst/common/scripting/tst.projid.d mode=0444
1613 file path=opt/SUNWdtrt/tst/common/scripting/tst.projid.ksh mode=0444
1614 file path=opt/SUNWdtrt/tst/common/scripting/tst.quite.d mode=0444
1615 file path=opt/SUNWdtrt/tst/common/scripting/tst.sid.d mode=0444
1616 file path=opt/SUNWdtrt/tst/common/scripting/tst.sid.ksh mode=0444
1617 file path=opt/SUNWdtrt/tst/common/scripting/tst.stringmacro.ksh mode=0444
1618 file path=opt/SUNWdtrt/tst/common/scripting/tst.taskid.d mode=0444
1619 file path=opt/SUNWdtrt/tst/common/scripting/tst.taskid.ksh mode=0444
1620 file path=opt/SUNWdtrt/tst/common/scripting/tst.trace.d mode=0444
1621 file path=opt/SUNWdtrt/tst/common/scripting/tst.uid.d mode=0444
1622 file path=opt/SUNWdtrt/tst/common/scripting/tst.uid.ksh mode=0444
1623 file path=opt/SUNWdtrt/tst/common/sdt/tst.sdtargs.d mode=0444
1624 file path=opt/SUNWdtrt/tst/common/sdt/tst.sdtargs.exe mode=0555
1625 file path=opt/SUNWdtrt/tst/common/sizeof/err.D_IDENT_BADREF.SizeofAssoc.d \
1626     mode=0444
1627 file path=opt/SUNWdtrt/tst/common/sizeof/err.D_IDENT_UNDEF.UnknownSymbol.d \
1628     mode=0444
1629 file path=opt/SUNWdtrt/tst/common/sizeof/err.D_SIZEOF_TYPE.badstruct.d \
1630     mode=0444
1631 file path=opt/SUNWdtrt/tst/common/sizeof/err.D_SIZEOF_TYPE.d mode=0444
1632 file path=opt/SUNWdtrt/tst/common/sizeof/err.D_SYNTAX.SizeofBadType.d \
1633     mode=0444
1634 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofArray.d mode=0444
1635 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofDataTypes.d mode=0444
1636 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofExpression.d mode=0444
1637 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofNULL.d mode=0444
1638 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofStrConst.d mode=0444
1639 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofStrConst.d.out mode=0444
1640 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofString1.d mode=0444
1641 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofString1.d.out mode=0444
1642 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofString2.d mode=0444
1643 file path=opt/SUNWdtrt/tst/common/sizeof/tst.SizeofString2.d.out mode=0444
```



```
1644 file path=opt/SUNWdtrt/tst/common/speculation/err.BufSizeVariations1.d \
1645     mode=0444
1646 file path=opt/SUNWdtrt/tst/common/speculation/err.BufSizeVariations2.d \
1647     mode=0444
1648 file \
1649     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithBreakPo
1650     mode=0444
1651 file \
1652     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithChill.d
1653     mode=0444
1654 file \
1655     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithCopyOut
1656     mode=0444
1657 file \
1658     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithCopyOut
1659     mode=0444
1660 file \
1661     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithPanic.d
1662     mode=0444
1663 file \
1664     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithRaise.d
1665     mode=0444
1666 file \
1667     path=opt/SUNWdtrt/tst/common/speculation/err.D_ACT_SPEC.SpeculateWithStop.d
1668     mode=0444
1669 file path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_COMM.AggAftCommit.d \
1670     mode=0444
1671 file \
1672     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithAvg.d \
1673     mode=0444
1674 file \
1675     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithCount.d
1676     mode=0444
1677 file \
1678     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithLquant.
1679     mode=0444
1680 file \
1681     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithMax.d \
1682     mode=0444
1683 file \
1684     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithMin.d \
1685     mode=0444
1686 file \
1687     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithQuant.d
1688     mode=0444
1689 file \
1690     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithStddev.
1691     mode=0444
1692 file \
1693     path=opt/SUNWdtrt/tst/common/speculation/err.D_AGG_SPEC.SpeculateWithSum.d \
1694     mode=0444
1695 file \
1696     path=opt/SUNWdtrt/tst/common/speculation/err.D_COMM_COMM.CommitAftCommit.d \
1697     mode=0444
1698 file path=opt/SUNWdtrt/tst/common/speculation/err.D_COMM_COMM.DisjointCommit.d \
1699     mode=0444
1700 file \
1701     path=opt/SUNWdtrt/tst/common/speculation/err.D_COMM_DREC.CommitAftDataRec.d
1702     mode=0444
1703 file \
1704     path=opt/SUNWdtrt/tst/common/speculation/err.D_DREC_COMM.DataRecAftCommit.d
1705     mode=0444
1706 file \
1707     path=opt/SUNWdtrt/tst/common/speculation/err.D_DREC_COMM.ExitAfterCommit.d \
1708     mode=0444
1709 file path=opt/SUNWdtrt/tst/common/speculation/err.D_EXIT_SPEC.ExitAftSpec.d \
```

```
1710     mode=0444
1711 file path=opt/SUNWdtrt/tst/common/speculation/err.D_PRAGMA_MALFORM.NspecExpr.d \
1712     mode=0444
1713 file \
1714     path=opt/SUNWdtrt/tst/common/speculation/err.D_PRAGMA_OPTSET.HugeNspecValue.
1715     mode=0444
1716 file \
1717     path=opt/SUNWdtrt/tst/common/speculation/err.D_PRAGMA_OPTSET.InvalidSpecSize
1718     mode=0444
1719 file \
1720     path=opt/SUNWdtrt/tst/common/speculation/err.D_PRAGMA_OPTSET.NegSpecSize.d \
1721     mode=0444
1722 file path=opt/SUNWdtrt/tst/common/speculation/err.D_PROTO_LEN.SpecNoId.d \
1723     mode=0444
1724 file path=opt/SUNWdtrt/tst/common/speculation/err.D_SPEC_COMM.SpecAftCommit.d \
1725     mode=0444
1726 file path=opt/SUNWdtrt/tst/common/speculation/err.D_SPEC_DREC.SpecAftDataRec.d \
1727     mode=0444
1728 file path=opt/SUNWdtrt/tst/common/speculation/err.D_SPEC_SPEC.SpecAftSpec.d \
1729     mode=0444
1730 file path=opt/SUNWdtrt/tst/common/speculation/err.NegativeBufSize.d mode=0444
1731 file path=opt/SUNWdtrt/tst/common/speculation/err.NegativeNspec.d mode=0444
1732 file path=opt/SUNWdtrt/tst/common/speculation/err.NegativeSpecSize.d mode=0444
1733 file path=opt/SUNWdtrt/tst/common/speculation/err.SpecSizeVariations1.d \
1734     mode=0444
1735 file path=opt/SUNWdtrt/tst/common/speculation/err.SpecSizeVariations2.d \
1736     mode=0444
1737 file path=opt/SUNWdtrt/tst/common/speculation/tst.CommitAfterDiscard.d \
1738     mode=0444
1739 file path=opt/SUNWdtrt/tst/common/speculation/tst.CommitWithZero.d mode=0444
1740 file path=opt/SUNWdtrt/tst/common/speculation/tst.DataRecAftDiscard.d \
1741     mode=0444
1742 file path=opt/SUNWdtrt/tst/common/speculation/tst.DiscardAftCommit.d mode=0444
1743 file path=opt/SUNWdtrt/tst/common/speculation/tst.DiscardAftDataRec.d \
1744     mode=0444
1745 file path=opt/SUNWdtrt/tst/common/speculation/tst.DiscardAftDiscard.d \
1746     mode=0444
1747 file path=opt/SUNWdtrt/tst/common/speculation/tst.DiscardWithZero.d mode=0444
1748 file path=opt/SUNWdtrt/tst/common/speculation/tst.ExitAftDiscard.d mode=0444
1749 file path=opt/SUNWdtrt/tst/common/speculation/tst.NoSpecBuffer.d mode=0444
1750 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpecSizeVariations1.d \
1751     mode=0444
1752 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpecSizeVariations2.d \
1753     mode=0444
1754 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpecSizeVariations3.d \
1755     mode=0444
1756 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpeculateWithRandom.d \
1757     mode=0444
1758 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpeculationCommit.d \
1759     mode=0444
1760 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpeculationDiscard.d \
1761     mode=0444
1762 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpeculationID.d mode=0444
1763 file path=opt/SUNWdtrt/tst/common/speculation/tst.SpeculationWithZero.d \
1764     mode=0444
1765 file path=opt/SUNWdtrt/tst/common/speculation/tst.TwoSpecBuffers.d mode=0444
1766 file path=opt/SUNWdtrt/tst/common/speculation/tst.negcommit.d mode=0444
1767 file path=opt/SUNWdtrt/tst/common/speculation/tst.negspec.d mode=0444
1768 file path=opt/SUNWdtrt/tst/common/speculation/tst.zerosize.d mode=0444
1769 file path=opt/SUNWdtrt/tst/common/stability/err.D_ATTR_MIN.MinAttributes.d \
1770     mode=0444
1771 file path=opt/SUNWdtrt/tst/common/stack/err.D_STACK_PROTO.bad.d mode=0444
1772 file path=opt/SUNWdtrt/tst/common/stack/err.D_STACK_SIZE.d mode=0444
1773 file path=opt/SUNWdtrt/tst/common/stack/err.D_USTACK_FRAMES.bad.d mode=0444
1774 file path=opt/SUNWdtrt/tst/common/stack/err.D_USTACK_PROTO.bad.d mode=0444
1775 file path=opt/SUNWdtrt/tst/common/stack/err.D_USTACK_STRSIZE.bad.d mode=0444
```



```
1776 file path=opt/SUNWdtrt/tst/common/stack/tst.default.d mode=0444
1777 file path=opt/SUNWdtrt/tst/common/stackdepth/tst.default.d mode=0444
1778 file path=opt/SUNWdtrt/tst/common/stop/tst.stop1.d mode=0444
1779 file path=opt/SUNWdtrt/tst/common/stop/tst.stop1.exe mode=0555
1780 file path=opt/SUNWdtrt/tst/common/stop/tst.stop2.d mode=0444
1781 file path=opt/SUNWdtrt/tst/common/stop/tst.stop2.exe mode=0555
1782 file path=opt/SUNWdtrt/tst/common/strlen/tst.strlen1.d mode=0444
1783 file path=opt/SUNWdtrt/tst/common/strtoll/err.BaseTooLarge.d mode=0444
1784 file path=opt/SUNWdtrt/tst/common/strtoll/err.BaseTooSmall.d mode=0444
1785 file path=opt/SUNWdtrt/tst/common/strtoll/tst.strtoll.d mode=0444
1786 file path=opt/SUNWdtrt/tst/common/strtoll/tst.strtoll.d.out mode=0444
1787 file path=opt/SUNWdtrt/tst/common/struct/err.D_ADDROF_VAR.StructPointer.d \
1788     mode=0444
1789 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_COMBO.StructWithoutColon.d \
1790     mode=0444
1791 file \
1792     path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_COMBO.StructWithoutColon1.d \
1793     mode=0444
1794 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_INCOMPLETE.circular.d \
1795     mode=0444
1796 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_INCOMPLETE.order.d \
1797     mode=0444
1798 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_INCOMPLETE.order2.d \
1799     mode=0444
1800 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_INCOMPLETE.recursive.d \
1801     mode=0444
1802 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_INCOMPLETE.simple.d \
1803     mode=0444
1804 file path=opt/SUNWdtrt/tst/common/struct/err.D_DECL_VOIDOBJ.baddec.d mode=0444
1805 file path=opt/SUNWdtrt/tst/common/struct/err.D_PROTO_ARG.DupStructAssoc.d \
1806     mode=0444
1807 file path=opt/SUNWdtrt/tst/common/struct/tst.StructAssoc.d mode=0444
1808 file path=opt/SUNWdtrt/tst/common/struct/tst.StructDataTypes.d mode=0444
1809 file path=opt/SUNWdtrt/tst/common/struct/tst.StructInside.d mode=0444
1810 file path=opt/SUNWdtrt/tst/common/struct/tst.clauselocal.d mode=0444
1811 file path=opt/SUNWdtrt/tst/common/struct/tst.clauselocal.d.out mode=0444
1812 file path=opt/SUNWdtrt/tst/common/syscall/tst.args.d mode=0444
1813 file path=opt/SUNWdtrt/tst/common/syscall/tst.args.exe mode=0555
1814 file path=opt/SUNWdtrt/tst/common/syscall/tst.openret.ksh mode=0444
1815 file path=opt/SUNWdtrt/tst/common/sysevent/tst.post.d mode=0444
1816 file path=opt/SUNWdtrt/tst/common/sysevent/tst.post.exe mode=0555
1817 file path=opt/SUNWdtrt/tst/common/sysevent/tst.post_chan.d mode=0444
1818 file path=opt/SUNWdtrt/tst/common/sysevent/tst.post_chan.exe mode=0555
1819 file path=opt/SUNWdtrt/tst/common/tick-n/err.D_PDESC_ZERO.tick.d mode=0444
1820 file path=opt/SUNWdtrt/tst/common/tick-n/err.D_PDESC_ZEROonens.d mode=0444
1821 file path=opt/SUNWdtrt/tst/common/tick-n/err.D_PDESC_ZEROonensec.d mode=0444
1822 file path=opt/SUNWdtrt/tst/common/tick-n/err.D_PDESC_ZEROoneus.d mode=0444
1823 file path=opt/SUNWdtrt/tst/common/tick-n/err.D_PDESC_ZEROoneusec.d mode=0444
1824 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickarg0.d mode=0444
1825 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickms.d mode=0444
1826 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickms.d.out mode=0444
1827 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickmsec.d mode=0444
1828 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickmsec.d.out mode=0444
1829 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickns.d mode=0444
1830 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickns.d.out mode=0444
1831 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticknsec.d mode=0444
1832 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticknsec.d.out mode=0444
1833 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticks.d mode=0444
1834 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticks.d.out mode=0444
1835 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticksec.d mode=0444
1836 file path=opt/SUNWdtrt/tst/common/tick-n/tst.ticksec.d.out mode=0444
1837 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickus.d mode=0444
1838 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickus.d.out mode=0444
1839 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickusec.d mode=0444
1840 file path=opt/SUNWdtrt/tst/common/tick-n/tst.tickusec.d.out mode=0444
1841 file path=opt/SUNWdtrt/tst/common/trace/err.D_PROTO_LEN.bad.d mode=0444
```

```
1842 file path=opt/SUNWdtrt/tst/common/trace/err.D_TRACE_AGG.bad.d mode=0444
1843 file path=opt/SUNWdtrt/tst/common/trace/err.D_TRACE_VOID.bad.d mode=0444
1844 file path=opt/SUNWdtrt/tst/common/trace/tst.dyn.d mode=0444
1845 file path=opt/SUNWdtrt/tst/common/trace/tst.misc.d mode=0444
1846 file path=opt/SUNWdtrt/tst/common/trace/tst.qstring.d mode=0444
1847 file path=opt/SUNWdtrt/tst/common/trace/tst.qstring.d.out mode=0444
1848 file path=opt/SUNWdtrt/tst/common/trace/tst.string.d mode=0444
1849 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_PROTO_ARG.badsize.d mode=0444
1850 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_PROTO_LEN.toofew.d mode=0444
1851 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_TRACEMEM_ADDR.badaddr.d \
1852     mode=0444
1853 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_TRACEMEM_ARGS.d mode=0444
1854 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_TRACEMEM_DYNSIZE.d mode=0444
1855 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_TRACEMEM_SIZE.negsize.d \
1856     mode=0444
1857 file path=opt/SUNWdtrt/tst/common/tracemem/err.D_TRACEMEM_SIZE.zerosize.d \
1858     mode=0444
1859 file path=opt/SUNWdtrt/tst/common/tracemem/tst.dynsize.d mode=0444
1860 file path=opt/SUNWdtrt/tst/common/tracemem/tst.dynsize.d.out mode=0444
1861 file path=opt/SUNWdtrt/tst/common/tracemem/tst.rootvp.d mode=0444
1862 file path=opt/SUNWdtrt/tst/common/tracemem/tst.smallsize.d mode=0444
1863 file path=opt/SUNWdtrt/tst/common/tracemem/tst.smallsize.d.out mode=0444
1864 file \
1865     path=opt/SUNWdtrt/tst/common/translators/err.D_DECL_TYPERED.BadTransDecl.d \
1866     mode=0444
1867 file \
1868     path=opt/SUNWdtrt/tst/common/translators/err.D_OP_INCOMPLETE.NonExistentInpu
1869     mode=0444
1870 file path=opt/SUNWdtrt/tst/common/translators/err.D_SYNTAX.BadTransDecl1.d \
1871     mode=0444
1872 file path=opt/SUNWdtrt/tst/common/translators/err.D_SYNTAX.BadTransDecl3.d \
1873     mode=0444
1874 file path=opt/SUNWdtrt/tst/common/translators/err.D_SYNTAX.BadTransDecl4.d \
1875     mode=0444
1876 file \
1877     path=opt/SUNWdtrt/tst/common/translators/err.D_TYPE_MEMBER.NonExistentInput2
1878     mode=0444
1879 file \
1880     path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_INCOMPAT.BadInputType1.
1881     mode=0444
1882 file \
1883     path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_MEMB.NonExistentOutput2
1884     mode=0444
1885 file path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_NONE.BadTransDecl6.d \
1886     mode=0444
1887 file \
1888     path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_REDECL.RepeatTransDecl.
1889     mode=0444
1890 file path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_SOU.BadTransDecl8.d \
1891     mode=0444
1892 file path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_SOU.BadTransInt.d \
1893     mode=0444
1894 file \
1895     path=opt/SUNWdtrt/tst/common/translators/err.D_XLATE_SOU.NonExistentOutput1.
1896     mode=0444
1897 file path=opt/SUNWdtrt/tst/common/translators/tst.CircularTransDecl.d \
1898     mode=0444
1899 file path=opt/SUNWdtrt/tst/common/translators/tst.EmptyTransDecl.d mode=0444
1900 file path=opt/SUNWdtrt/tst/common/translators/tst.ForwardTag.d mode=0444
1901 file path=opt/SUNWdtrt/tst/common/translators/tst.InputAliasTrans.d mode=0444
1902 file path=opt/SUNWdtrt/tst/common/translators/tst.InputIntTrans.d mode=0444
1903 file path=opt/SUNWdtrt/tst/common/translators/tst.OutputAliasTrans.d mode=0444
1904 file path=opt/SUNWdtrt/tst/common/translators/tst.PartialDereferencing.d \
1905     mode=0444
1906 file path=opt/SUNWdtrt/tst/common/translators/tst.PartialOutputTransDefn.d \
1907     mode=0444
```



```
1908 file path=opt/SUNWdtrt/tst/common/translators/tst.ProcModelTrans.d mode=0444
1909 file path=opt/SUNWdtrt/tst/common/translators/tst.RepeatDeclaration.d \
1910     mode=0444
1911 file path=opt/SUNWdtrt/tst/common/translators/tst.SimultaneousTranslators.d \
1912     mode=0444
1913 file path=opt/SUNWdtrt/tst/common/translators/tst.StructureAssignment.d \
1914     mode=0444
1915 file path=opt/SUNWdtrt/tst/common/translators/tst.TestTransStability1.ksh \
1916     mode=0444
1917 file path=opt/SUNWdtrt/tst/common/translators/tst.TestTransStability1.ksh.out \
1918     mode=0444
1919 file path=opt/SUNWdtrt/tst/common/translators/tst.TestTransStability2.ksh \
1920     mode=0444
1921 file path=opt/SUNWdtrt/tst/common/translators/tst.TestTransStability2.ksh.out \
1922     mode=0444
1923 file path=opt/SUNWdtrt/tst/common/translators/tst.TransNonPointer.d mode=0444
1924 file path=opt/SUNWdtrt/tst/common/translators/tst.TransOutputPointer.d \
1925     mode=0444
1926 file path=opt/SUNWdtrt/tst/common/translators/tst.TransPointer.d mode=0444
1927 file path=opt/SUNWdtrt/tst/common/translators/tst.TranslateSelf.d mode=0444
1928 file path=opt/SUNWdtrt/tst/common/translators/tst.UnionInputTrans.d mode=0444
1929 file path=opt/SUNWdtrt/tst/common/translators/tst.UnionOutputTrans.d mode=0444
1930 file path=opt/SUNWdtrt/tst/common/typedef/err.D_DECL_IDRED.DupTypeDef.d \
1931     mode=0444
1932 file path=opt/SUNWdtrt/tst/common/typedef/err.D_SYNTAX.BadExistingTypedef.d \
1933     mode=0444
1934 file path=opt/SUNWdtrt/tst/common/typedef/err.D_SYNTAX.TypedefInClause.d \
1935     mode=0444
1936 file path=opt/SUNWdtrt/tst/common/typedef/tst.ChainTypedef.d mode=0444
1937 file path=opt/SUNWdtrt/tst/common/typedef/tst.TypedefDataAssign.d mode=0444
1938 file path=opt/SUNWdtrt/tst/common/types/err.D_CAST_INVAL.badcast.d mode=0444
1939 file path=opt/SUNWdtrt/tst/common/types/err.D_CG_DYN.ResultDynType.d mode=0444
1940 file path=opt/SUNWdtrt/tst/common/types/err.D_CHR_OFLOW.charconst.d mode=0444
1941 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_BADCLASS.bad.d mode=0444
1942 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_CHARATTR.badtype3.d \
1943     mode=0444
1944 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_COMBO.badtype4.d mode=0444
1945 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_COMBO.badtype5.d mode=0444
1946 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_ENCONST.badeval.d mode=0444
1947 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_ENOFLOW.enoflow.d mode=0444
1948 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_ENOFLOW.enuflow.d mode=0444
1949 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_SCOPE.scopeop.d mode=0444
1950 file path=opt/SUNWdtrt/tst/common/types/err.D_DECL_USELESS.baddec.d mode=0444
1951 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_ACT.badcond.d mode=0444
1952 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_ARITH.badoperand.d mode=0444
1953 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_INCOMPAT.badassign.d \
1954     mode=0444
1955 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_INT.badbitop.d mode=0444
1956 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_INT.badshift.d mode=0444
1957 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_SCALAR.badcond.d mode=0444
1958 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_SCALAR.badincop.d mode=0444
1959 file path=opt/SUNWdtrt/tst/common/types/err.D_OP_SCALAR.badlogop.d mode=0444
1960 file path=opt/SUNWdtrt/tst/common/types/err.D_PROTO_LEN.badcond1.d mode=0444
1961 file path=opt/SUNWdtrt/tst/common/types/err.D_SYNTAX.badenum.d mode=0444
1962 file path=opt/SUNWdtrt/tst/common/types/err.D_SYNTAX.badid.d mode=0444
1963 file path=opt/SUNWdtrt/tst/common/types/err.D_SYNTAX.badstruct.d mode=0444
1964 file path=opt/SUNWdtrt/tst/common/types/err.D_UNKNOWN.badtype1.d mode=0444
1965 file path=opt/SUNWdtrt/tst/common/types/err.D_UNKNOWN.badtype2.d mode=0444
1966 file path=opt/SUNWdtrt/tst/common/types/err.D_UNKNOWN.dupenum.d mode=0444
1967 file path=opt/SUNWdtrt/tst/common/types/err.D_UNKNOWN.dupstruct.d mode=0444
1968 file path=opt/SUNWdtrt/tst/common/types/err.D_XLATE_REDECL.ResultDynType.d \
1969     mode=0444
1970 file path=opt/SUNWdtrt/tst/common/types/tst.assignops.d mode=0444
1971 file path=opt/SUNWdtrt/tst/common/types/tst.badshiftops.d mode=0444
1972 file path=opt/SUNWdtrt/tst/common/types/tst.basics.d mode=0444
1973 file path=opt/SUNWdtrt/tst/common/types/tst.basics.d.out mode=0444
```

```
1974 file path=opt/SUNWdtrt/tst/common/types/tst.bitops.d mode=0444
1975 file path=opt/SUNWdtrt/tst/common/types/tst.charconstants.d mode=0444
1976 file path=opt/SUNWdtrt/tst/common/types/tst.complex.d mode=0444
1977 file path=opt/SUNWdtrt/tst/common/types/tst.condexpr.d mode=0444
1978 file path=opt/SUNWdtrt/tst/common/types/tst.const.d mode=0444
1979 file path=opt/SUNWdtrt/tst/common/types/tst.constants.d mode=0444
1980 file path=opt/SUNWdtrt/tst/common/types/tst.conv.d mode=0444
1981 file path=opt/SUNWdtrt/tst/common/types/tst.enum.d mode=0444
1982 file path=opt/SUNWdtrt/tst/common/types/tst.intincop.d mode=0444
1983 file path=opt/SUNWdtrt/tst/common/types/tst.intops.d mode=0444
1984 file path=opt/SUNWdtrt/tst/common/types/tst.inttypes.d mode=0444
1985 file path=opt/SUNWdtrt/tst/common/types/tst.ptrincop.d mode=0444
1986 file path=opt/SUNWdtrt/tst/common/types/tst.ptrops.d mode=0444
1987 file path=opt/SUNWdtrt/tst/common/types/tst.relenum.d mode=0444
1988 file path=opt/SUNWdtrt/tst/common/types/tst.relstring.d mode=0444
1989 file path=opt/SUNWdtrt/tst/common/types/tst.shiftops.d mode=0444
1990 file path=opt/SUNWdtrt/tst/common/types/tst.stringconstants.d mode=0444
1991 file path=opt/SUNWdtrt/tst/common/types/tst.struct.d mode=0444
1992 file path=opt/SUNWdtrt/tst/common/types/tst.typedef.d mode=0444
1993 file path=opt/SUNWdtrt/tst/common/types/tst.unaryop.d mode=0444
1994 file path=opt/SUNWdtrt/tst/common/uctf/err.invalidpid.d mode=0444
1995 file path=opt/SUNWdtrt/tst/common/uctf/err.invalidpid2.d mode=0444
1996 file path=opt/SUNWdtrt/tst/common/uctf/err.invalidpid3.d mode=0444
1997 file path=opt/SUNWdtrt/tst/common/uctf/err.invalidtype.ksh mode=0444
1998 file path=opt/SUNWdtrt/tst/common/uctf/err.invalidtype2.ksh mode=0444
1999 file path=opt/SUNWdtrt/tst/common/uctf/err.user64mode.ksh mode=0444
2000 file path=opt/SUNWdtrt/tst/common/uctf/tst.aouttype.exe mode=0555
2001 file path=opt/SUNWdtrt/tst/common/uctf/tst.aouttype.ksh mode=0444
2002 file path=opt/SUNWdtrt/tst/common/uctf/tst.chasestrings.exe mode=0555
2003 file path=opt/SUNWdtrt/tst/common/uctf/tst.chasestrings.ksh mode=0444
2004 file path=opt/SUNWdtrt/tst/common/uctf/tst.chasestrings.ksh.out mode=0444
2005 file path=opt/SUNWdtrt/tst/common/uctf/tst.libtype.exe mode=0555
2006 file path=opt/SUNWdtrt/tst/common/uctf/tst.libtype.ksh mode=0444
2007 file path=opt/SUNWdtrt/tst/common/uctf/tst.linkmap.ksh mode=0444
2008 file path=opt/SUNWdtrt/tst/common/uctf/tst.pidprint.ksh mode=0444
2009 file path=opt/SUNWdtrt/tst/common/uctf/tst.pidprinttarg.ksh mode=0444
2010 file path=opt/SUNWdtrt/tst/common/uctf/tst.printtype.exe mode=0555
2011 file path=opt/SUNWdtrt/tst/common/uctf/tst.printtype.ksh mode=0444
2012 file path=opt/SUNWdtrt/tst/common/uctf/tst.printtype.ksh.out mode=0444
2013 file path=opt/SUNWdtrt/tst/common/uctf/tst.printtypetarg.ksh mode=0444
2014 file path=opt/SUNWdtrt/tst/common/uctf/tst.userlandkey.ksh mode=0444
2015 file path=opt/SUNWdtrt/tst/common/uctf/tst.userlandkey.ksh.out mode=0444
2016 file path=opt/SUNWdtrt/tst/common/uctf/tst.userstrings.ksh mode=0444
2017 file path=opt/SUNWdtrt/tst/common/uctf/tst.userstrings.ksh.out mode=0444
2018 file path=opt/SUNWdtrt/tst/common/union/err.D_ADDROF_VAR.UnionPointer.d \
2019     mode=0444
2020 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_COMBO.UnionWithoutColon.d \
2021     mode=0444
2022 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_COMBO.UnionWithoutColon1.d \
2023     mode=0444
2024 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_INCOMPLETE.circular.d \
2025     mode=0444
2026 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_INCOMPLETE.order.d \
2027     mode=0444
2028 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_INCOMPLETE.recursive.d \
2029     mode=0444
2030 file path=opt/SUNWdtrt/tst/common/union/err.D_DECL_INCOMPLETE.simple.d \
2031     mode=0444
2032 file path=opt/SUNWdtrt/tst/common/union/err.D_PROTO_ARG.DupUnionAssoc.d \
2033     mode=0444
2034 file path=opt/SUNWdtrt/tst/common/union/tst.UnionAssoc.d mode=0444
2035 file path=opt/SUNWdtrt/tst/common/union/tst.UnionDataTypes.d mode=0444
2036 file path=opt/SUNWdtrt/tst/common/union/tst.UnionInside.d mode=0444
2037 file path=opt/SUNWdtrt/tst/common/usdt/tst.andpid.ksh mode=0444
2038 file path=opt/SUNWdtrt/tst/common/usdt/tst.argmap.d mode=0444
2039 file path=opt/SUNWdtrt/tst/common/usdt/tst.argmap.exe mode=0555
```



```
2040 file path=opt/SUNWdtrt/tst/common/usdt/tst.args.d mode=0444
2041 file path=opt/SUNWdtrt/tst/common/usdt/tst.args.exe mode=0555
2042 file path=opt/SUNWdtrt/tst/common/usdt/tst.badguess.ksh mode=0444
2043 file path=opt/SUNWdtrt/tst/common/usdt/tst.corruptenv.ksh mode=0444
2044 file path=opt/SUNWdtrt/tst/common/usdt/tst.dlclose1.ksh mode=0444
2045 file path=opt/SUNWdtrt/tst/common/usdt/tst.dlclose1.ksh.out mode=0444
2046 file path=opt/SUNWdtrt/tst/common/usdt/tst.dlclose2.ksh mode=0444
2047 file path=opt/SUNWdtrt/tst/common/usdt/tst.dlclose2.ksh.out mode=0444
2048 file path=opt/SUNWdtrt/tst/common/usdt/tst.dlclose3.ksh mode=0444
2049 file path=opt/SUNWdtrt/tst/common/usdt/tst.eliminate.ksh mode=0444
2050 file path=opt/SUNWdtrt/tst/common/usdt/tst.enabled.ksh mode=0444
2051 file path=opt/SUNWdtrt/tst/common/usdt/tst.enabled.ksh.out mode=0444
2052 file path=opt/SUNWdtrt/tst/common/usdt/tst.enabled2.ksh mode=0444
2053 file path=opt/SUNWdtrt/tst/common/usdt/tst.enabled2.ksh.out mode=0444
2054 file path=opt/SUNWdtrt/tst/common/usdt/tst.entryreturn.ksh mode=0444
2055 file path=opt/SUNWdtrt/tst/common/usdt/tst.entryreturn.ksh.out mode=0444
2056 file path=opt/SUNWdtrt/tst/common/usdt/tst.fork.ksh mode=0444
2057 file path=opt/SUNWdtrt/tst/common/usdt/tst.fork.ksh.out mode=0444
2058 file path=opt/SUNWdtrt/tst/common/usdt/tst.forker.exe mode=0555
2059 file path=opt/SUNWdtrt/tst/common/usdt/tst.forker.ksh mode=0444
2060 file path=opt/SUNWdtrt/tst/common/usdt/tst.guess32.ksh mode=0444
2061 file path=opt/SUNWdtrt/tst/common/usdt/tst.guess64.ksh mode=0444
2062 file path=opt/SUNWdtrt/tst/common/usdt/tst.header.ksh mode=0444
2063 file path=opt/SUNWdtrt/tst/common/usdt/tst.include.ksh mode=0444
2064 file path=opt/SUNWdtrt/tst/common/usdt/tst.lazyprobe.exe mode=0555
2065 file path=opt/SUNWdtrt/tst/common/usdt/tst.lazyprobe1.ksh mode=0444
2066 file path=opt/SUNWdtrt/tst/common/usdt/tst.lazyprobe2.ksh mode=0444
2067 file path=opt/SUNWdtrt/tst/common/usdt/tst.linkpriv.ksh mode=0444
2068 file path=opt/SUNWdtrt/tst/common/usdt/tst.linkunpriv.ksh mode=0444
2069 file path=opt/SUNWdtrt/tst/common/usdt/tst.multiple.ksh mode=0444
2070 file path=opt/SUNWdtrt/tst/common/usdt/tst.multiple.ksh.out mode=0444
2071 file path=opt/SUNWdtrt/tst/common/usdt/tst.multiprov.ksh mode=0444
2072 file path=opt/SUNWdtrt/tst/common/usdt/tst.multiprov.ksh.out mode=0444
2073 file path=opt/SUNWdtrt/tst/common/usdt/tst.nodtrace.ksh mode=0444
2074 file path=opt/SUNWdtrt/tst/common/usdt/tst.noprobes.ksh mode=0444
2075 file path=opt/SUNWdtrt/tst/common/usdt/tst.noreap.ksh mode=0444
2076 file path=opt/SUNWdtrt/tst/common/usdt/tst.noreapring.ksh mode=0444
2077 file path=opt/SUNWdtrt/tst/common/usdt/tst.onlyenabled.ksh mode=0444
2078 file path=opt/SUNWdtrt/tst/common/usdt/tst.reap.ksh mode=0444
2079 file path=opt/SUNWdtrt/tst/common/usdt/tst.reeval.ksh mode=0444
2080 file path=opt/SUNWdtrt/tst/common/usdt/tst.static.ksh mode=0444
2081 file path=opt/SUNWdtrt/tst/common/usdt/tst.static.ksh.out mode=0444
2082 file path=opt/SUNWdtrt/tst/common/usdt/tst.static2.ksh mode=0444
2083 file path=opt/SUNWdtrt/tst/common/usdt/tst.static2.ksh.out mode=0444
2084 file path=opt/SUNWdtrt/tst/common/usdt/tst.user.ksh mode=0444
2085 file path=opt/SUNWdtrt/tst/common/usdt/tst.user.ksh.out mode=0444
2086 file path=opt/SUNWdtrt/tst/common/ustack/tst.bigstack.d mode=0444
2087 file path=opt/SUNWdtrt/tst/common/ustack/tst.bigstack.exe mode=0555
2088 file path=opt/SUNWdtrt/tst/common/ustack/tst.depth.ksh mode=0444
2089 file path=opt/SUNWdtrt/tst/common/ustack/tst.spin.exe mode=0555
2090 file path=opt/SUNWdtrt/tst/common/ustack/tst.spin.ksh mode=0444
2091 file path=opt/SUNWdtrt/tst/common/vars/tst.gid.d mode=0444
2092 file path=opt/SUNWdtrt/tst/common/vars/tst.nullassign.d mode=0444
2093 file path=opt/SUNWdtrt/tst/common/vars/tst.ppid.d mode=0444
2094 file path=opt/SUNWdtrt/tst/common/vars/tst.ucaller.ksh mode=0444
2095 file path=opt/SUNWdtrt/tst/common/vars/tst.ucaller.ksh.out mode=0444
2096 file path=opt/SUNWdtrt/tst/common/vars/tst.uid.d mode=0444
2097 file path=opt/SUNWdtrt/tst/common/vars/tst.walltimestamp.d mode=0444
2098 file path=opt/SUNWdtrt/tst/common/version/tst.1.0.d mode=0444
2099 $(i386_ONLY)file path=opt/SUNWdtrt/tst/i86xpv/xdt/tst.basic.ksh mode=0444
2100 $(i386_ONLY)file path=opt/SUNWdtrt/tst/i86xpv/xdt/tst.hvmenable.ksh mode=0444
2101 $(i386_ONLY)file path=opt/SUNWdtrt/tst/i86xpv/xdt/tst.memenable.ksh mode=0444
2102 $(i386_ONLY)file path=opt/SUNWdtrt/tst/i86xpv/xdt/tst.schedargs.ksh mode=0444
2103 $(i386_ONLY)file path=opt/SUNWdtrt/tst/i86xpv/xdt/tst.schedenable.ksh \
2104     mode=0444
2105 legacy pkg=SUNWdtrt category=internal \
```

```
2106     desc="DTrace Test Suite Internal Distribution" \
2107     hotline="Contact the DTrace discussion forum" name="DTrace Test Suite"
2108 license cr_Sun license=cr_Sun
2109 license lic_CDDL license=lic_CDDL
2110 depend fmri=runtime/java type=require
2111 depend fmri=runtime/java/runtime64 type=require
```

```
**********************************************************
  102301 Tue Jan 14 20:13:07 2014
new/usr/src/uts/common/sys/dtrace.h
4471 DTrace count() with histogram
4472 DTrace full width distribution histograms
4473 DTrace frequency trails
**********************************************************
______unchanged_portion_omitted_

 977 #define DTRACE_SIZEOF_EPROBEDESC(desc)                          \
 978         (sizeof (dtrace_eprobedesc_t) + ((desc)->dtepd_nrecs ?  \
 979         (((desc)->dtepd_nrecs - 1) * sizeof (dtrace_recdesc_t)) : 0))

 981 #define DTRACE_SIZEOF_AGGDESC(desc)                             \
 982         (sizeof (dtrace_aggdesc_t) + ((desc)->dtagd_nrecs ?     \
 983         (((desc)->dtagd_nrecs - 1) * sizeof (dtrace_recdesc_t)) : 0))

 985 /*
 986  * DTrace Option Interface
 987  *
 988  * Run-time DTrace options are set and retrieved via DOF_SECT_OPTDESC sections
 989  * in a DOF image.  The dof_optdesc structure contains an option identifier and
 990  * an option value.  The valid option identifiers are found below; the mapping
 991  * between option identifiers and option identifying strings is maintained at
 992  * user-level.  Note that the value of DTRACEOPT_UNSET is such that all of the
 993  * following are potentially valid option values:  all positive integers, zero
 994  * and negative one.  Some options (notably "bufpolicy" and "bufresize") take
 995  * predefined tokens as their values; these are defined with
 996  * DTRACEOPT_{option}_{token}.
 997  */
 998 #define DTRACEOPT_BUFSIZE       0       /* buffer size */
 999 #define DTRACEOPT_BUFPOLICY     1       /* buffer policy */
1000 #define DTRACEOPT_DYNVARSIZE    2       /* dynamic variable size */
1001 #define DTRACEOPT_AGGSIZE       3       /* aggregation size */
1002 #define DTRACEOPT_SPECSIZE      4       /* speculation size */
1003 #define DTRACEOPT_NSPEC         5       /* number of speculations */
1004 #define DTRACEOPT_STRSIZE       6       /* string size */
1005 #define DTRACEOPT_CLEANRATE     7       /* dynvar cleaning rate */
1006 #define DTRACEOPT_CPU           8       /* CPU to trace */
1007 #define DTRACEOPT_BUFRESIZE     9       /* buffer resizing policy */
1008 #define DTRACEOPT_GRABANON      10      /* grab anonymous state, if any */
1009 #define DTRACEOPT_FLOWINDENT    11      /* indent function entry/return */
1010 #define DTRACEOPT_QUIET         12      /* only output explicitly traced data */
1011 #define DTRACEOPT_STACKFRAMES   13      /* number of stack frames */
1012 #define DTRACEOPT_USTACKFRAMES  14      /* number of user stack frames */
1013 #define DTRACEOPT_AGGRATE       15      /* aggregation snapshot rate */
1014 #define DTRACEOPT_SWITCHRATE    16      /* buffer switching rate */
1015 #define DTRACEOPT_STATUSRATE    17      /* status rate */
1016 #define DTRACEOPT_DESTRUCTIVE   18      /* destructive actions allowed */
1017 #define DTRACEOPT_STACKINDENT   19      /* output indent for stack traces */
1018 #define DTRACEOPT_RAWBYTES      20      /* always print bytes in raw form */
1019 #define DTRACEOPT_JSTACKFRAMES  21      /* number of jstack() frames */
1020 #define DTRACEOPT_JSTACKSTRSIZE 22      /* size of jstack() string table */
1021 #define DTRACEOPT_AGGSORTKEY    23      /* sort aggregations by key */
1022 #define DTRACEOPT_AGGSORTREV    24      /* reverse-sort aggregations */
1023 #define DTRACEOPT_AGGSORTPOS    25      /* agg. position to sort on */
1024 #define DTRACEOPT_AGGSORTKEYPOS 26      /* agg. key position to sort on */
1025 #define DTRACEOPT_TEMPORAL      27      /* temporally ordered output */
1026 #define DTRACEOPT_AGGHIST       28      /* histogram aggregation output */
1027 #define DTRACEOPT_AGGPACK       29      /* packed aggregation output */
1028 #define DTRACEOPT_AGGZOOM       30      /* zoomed aggregation scaling */
1029 #define DTRACEOPT_ZONE          31      /* zone in which to enable probes */
1030 #define DTRACEOPT_MAX           32      /* number of options */
1026 #define DTRACEOPT_MAX           28      /* number of options */

1032 #define DTRACEOPT_UNSET         (dtrace_optval_t)-2     /* unset option */
```



```
1034 #define DTRACEOPT_BUFPOLICY_RING        0       /* ring buffer */
1035 #define DTRACEOPT_BUFPOLICY_FILL        1       /* fill buffer, then stop */
1036 #define DTRACEOPT_BUFPOLICY_SWITCH      2       /* switch buffers */

1038 #define DTRACEOPT_BUFRESIZE_AUTO        0       /* automatic resizing */
1039 #define DTRACEOPT_BUFRESIZE_MANUAL      1       /* manual resizing */

1041 /*
1042  * DTrace Buffer Interface
1043  *
1044  * In order to get a snapshot of the principal or aggregation buffer,
1045  * user-level passes a buffer description to the kernel with the dtrace_bufdesc
1046  * structure.  This describes which CPU user-level is interested in, and
1047  * where user-level wishes the kernel to snapshot the buffer to (the
1048  * dtbd_data field).  The kernel uses the same structure to pass back some
1049  * information regarding the buffer:  the size of data actually copied out, the
1050  * number of drops, the number of errors, the offset of the oldest record,
1051  * and the time of the snapshot.
1052  *
1053  * If the buffer policy is a "switch" policy, taking a snapshot of the
1054  * principal buffer has the additional effect of switching the active and
1055  * inactive buffers.  Taking a snapshot of the aggregation buffer _always_ has
1056  * the additional effect of switching the active and inactive buffers.
1057  */
1058 typedef struct dtrace_bufdesc {
1059         uint64_t dtbd_size;                     /* size of buffer */
1060         uint32_t dtbd_cpu;                      /* CPU or DTRACE_CPUALL */
1061         uint32_t dtbd_errors;                   /* number of errors */
1062         uint64_t dtbd_drops;                    /* number of drops */
1063         DTRACE_PTR(char, dtbd_data);            /* data */
1064         uint64_t dtbd_oldest;                   /* offset of oldest record */
1065         uint64_t dtbd_timestamp;                /* hrtime of snapshot */
1066 } dtrace_bufdesc_t;
______unchanged_portion_omitted_
```