```
**********************************************************
   36592 Wed Mar 13 17:08:29 2013
new/usr/src/cmd/stat/kstat/kstat.c
3623 kstat must accept partial stat specification
**********************************************************
/*
 * 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 (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2013 David Hoeppner. All rights reserved.
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * Display kernel statistics
 *
 * This is a reimplementation of the perl kstat command originally found
 * under usr/src/cmd/kstat/kstat.pl
 *
 * Incompatibilities:
 *      - perl regular expressions replaced with extended REs bracketed by '/'
 *      - options checking is stricter
 *
 * Flags added:
 *      -C      similar to the -p option but value is separated by a colon
 *      -h      display help
 *      -j      json format
 */

#include <assert.h>
#include <ctype.h>
#include <errno.h>
#include <kstat.h>
#include <langinfo.h>
#include <libgen.h>
#include <limits.h>
#include <locale.h>
#include <signal.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <time.h>
#include <unistd.h>
#include <sys/list.h>
#include <sys/time.h>
```

```
#include <sys/types.h>

#include "kstat.h"
#include "statcommon.h"

char    *cmdname = "kstat";     /* Name of this command */
int     caught_cont = 0;        /* Have caught a SIGCONT */

static uint_t   g_timestamp_fmt = NODATE;

/* Helper flag - header was printed already? */
static boolean_t g_headerflg;

/* Saved command line options */
static boolean_t g_cflg = B_FALSE;
static boolean_t g_jflg = B_FALSE;
static boolean_t g_lflg = B_FALSE;
static boolean_t g_pflg = B_FALSE;
static boolean_t g_qflg = B_FALSE;
static ks_pattern_t     g_ks_class = {"*", 0};

/* Return zero if a selector did match */
static int      g_matched = 1;

/* Sorted list of kstat instances */
static list_t   instances_list;
static list_t   selector_list;

int
main(int argc, char **argv)
{
        ks_selector_t   *nselector;
        ks_selector_t   *uselector;
        kstat_ctl_t     *kc;
        hrtime_t        start_n;
        hrtime_t        period_n;
        boolean_t       errflg = B_FALSE;
        boolean_t       nselflg = B_FALSE;
        boolean_t       uselflg = B_FALSE;
        char            *q;
        int             count = 1;
        int             infinite_cycles = 0;
        int             interval = 0;
        int             n = 0;
        int             c, m, tmp;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)               /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"          /* Use this only if it wasn't */
#endif
        (void) textdomain(TEXT_DOMAIN);

        /*
         * Create the selector list and a dummy default selector to match
         * everything. While we process the cmdline options we will add
         * selectors to this list.
         */
        list_create(&selector_list, sizeof (ks_selector_t),
            offsetof(ks_selector_t, ks_next));

        nselector = new_selector();

        /*
         * Parse named command line arguments.
         */
        while ((c = getopt(argc, argv, "h?CqjlpT:m:i:n:s:c:")) != EOF)
```

```
		switch (c) {
		case 'h':
		case '?':
			usage();
			exit(0);
			break;
		case 'C':
			g_pflg = g_cflg = B_TRUE;
			break;
		case 'q':
			g_qflg = B_TRUE;
			break;
		case 'j':
			g_jflg = B_TRUE;
			break;
		case 'l':
			g_pflg = g_lflg = B_TRUE;
			break;
		case 'p':
			g_pflg = B_TRUE;
			break;
		case 'T':
			switch (*optarg) {
			case 'd':
				g_timestamp_fmt = DDATE;
				break;
			case 'u':
				g_timestamp_fmt = UDATE;
				break;
			default:
				errflg = B_TRUE;
			}
			break;
		case 'm':
			nselflg = B_TRUE;
			nselector->ks_module.pstr =
			    (char *)ks_safe_strdup(optarg);
			break;
		case 'i':
			nselflg = B_TRUE;
			nselector->ks_instance.pstr =
			    (char *)ks_safe_strdup(optarg);
			break;
		case 'n':
			nselflg = B_TRUE;
			nselector->ks_name.pstr =
			    (char *)ks_safe_strdup(optarg);
			break;
		case 's':
			nselflg = B_TRUE;
			nselector->ks_statistic.pstr =
			    (char *)ks_safe_strdup(optarg);
			break;
		case 'c':
			g_ks_class.pstr =
			    (char *)ks_safe_strdup(optarg);
			break;
		default:
			errflg = B_TRUE;
			break;
		}

	if (g_qflg && (g_jflg || g_pflg)) {
		(void) fprintf(stderr, gettext(
		    "-q and -lpj are mutually exclusive\n"));
		errflg = B_TRUE;
```

```
	}

	if (errflg) {
		usage();
		exit(2);
	}

	argc -= optind;
	argv += optind;

	/*
	 * Consume the rest of the command line. Parsing the
	 * unnamed command line arguments.
	 */
	while (argc--) {
		errno = 0;
		tmp = strtoul(*argv, &q, 10);
		if (tmp == ULONG_MAX && errno == ERANGE) {
			if (n == 0) {
				(void) fprintf(stderr, gettext(
				    "Interval is too large\n"));
			} else if (n == 1) {
				(void) fprintf(stderr, gettext(
				    "Count is too large\n"));
			}
			usage();
			exit(2);
		}

		if (errno != 0 || *q != '\0') {
			m = 0;
			uselector = new_selector();
			while ((q = (char *)strsep(argv, ":")) != NULL) {
				m++;
				if (m > 4) {
					free(uselector);
					usage();
					exit(2);
				}

				if (*q != '\0') {
					switch (m) {
					case 1:
						uselector->ks_module.pstr =
						    (char *)ks_safe_strdup(q);
						break;
					case 2:
						uselector->ks_instance.pstr =
						    (char *)ks_safe_strdup(q);
						break;
					case 3:
						uselector->ks_name.pstr =
						    (char *)ks_safe_strdup(q);
						break;
					case 4:
						uselector->ks_statistic.pstr =
						    (char *)ks_safe_strdup(q);
						break;
					default:
						assert(B_FALSE);
					}
				}
			}

			if (m < 4) {
				free(uselector);
```

```
                                usage();
                                exit(2);
                        }

                        uselflg = B_TRUE;
                        list_insert_tail(&selector_list, uselector);
                } else {
                        if (tmp < 1) {
                                if (n == 0) {
                                        (void) fprintf(stderr, gettext(
                                            "Interval must be an "
                                            "integer >= 1"));
                                } else if (n == 1) {
                                        (void) fprintf(stderr, gettext(
                                            "Count must be an integer >= 1"));
                                }
                                usage();
                                exit(2);
                        } else {
                                if (n == 0) {
                                        interval = tmp;
                                        count = -1;
                                } else if (n == 1) {
                                        count = tmp;
                                } else {
                                        usage();
                                        exit(2);
                                }
                        }
                        n++;
                }
                argv++;
        }

        /*
         * Check if we founded a named selector on the cmdline.
         */
        if (uselflg) {
                if (nselflg) {
                        (void) fprintf(stderr, gettext(
                            "[module[:instance[:name[:statistic]]]] and "
                            "module:instance:name:statistic and "
                            "-m -i -n -s are mutually exclusive"));
                        usage();
                        exit(2);
                } else {
                        free(nselector);
                }
        } else {
                list_insert_tail(&selector_list, nselector);
        }

        assert(!list_is_empty(&selector_list));

        list_create(&instances_list, sizeof (ks_instance_t),
            offsetof(ks_instance_t, ks_next));

        while ((kc = kstat_open()) == NULL) {
                if (errno == EAGAIN) {
                        (void) poll(NULL, 0, 200);
                } else {
                        perror("kstat_open");
                        exit(3);
                }
        }
```

```
        if (count > 1) {
                if (signal(SIGCONT, cont_handler) == SIG_ERR) {
                        (void) fprintf(stderr, gettext(
                            "signal failed"));
                        exit(3);
                }
        }

        period_n = (hrtime_t)interval * NANOSEC;
        start_n = gethrtime();

        while (count == -1 || count-- > 0) {
                ks_instances_read(kc);
                ks_instances_print();

                if (interval && count) {
                        ks_sleep_until(&start_n, period_n, infinite_cycles,
                            &caught_cont);
                        (void) kstat_chain_update(kc);
                        (void) putchar('\n');
                }
        }

        (void) kstat_close(kc);

        return (g_matched);
}

/*
 * Print usage.
 */
static void
usage(void)
{
        (void) fprintf(stderr, gettext(
            "Usage:\n"
            "kstat [ -Cjlpq ] [ -T d|u ] [ -c class ]\n"
            "      [ -m module ] [ -i instance ] [ -n name ] [ -s statistic ]\n"
            "      [ interval [ count ] ]\n"
            "kstat [ -Cjlpq ] [ -T d|u ] [ -c class ]\n"
            "      [ module[:instance[:name[:statistic]]] ... ]\n"
            "      [ module:instance:name:statistic ... ]\n"
            "      [ interval [ count ] ]\n"));
}
______unchanged_portion_omitted_
```

```
**********************************************************
    9157 Wed Mar 13 17:08:30 2013
new/usr/src/man/man1m/kstat.1m
3623 kstat must accept partial stat specification
**********************************************************
'\" te
.\" Copyright (c) 2000, Sun Microsystems, Inc. All Rights Reserved
.\" The contents of this file are subject to the terms of the Common Development
.\"  See the License for the specific language governing permissions and limitat
.\" the fields enclosed by brackets "[]" replaced with your own identifying info
.TH KSTAT 1M "Jan 9, 2013"
.SH NAME
kstat \- display kernel statistics
.SH SYNOPSIS
.LP
.nf
\fBkstat\fR [\fB-Cjlpq\fR] [\fB-T\fR u | d ] [\fB-c\fR \fIclass\fR] [\fB-m\fR \f
     [\fB-i\fR \fIinstance\fR] [\fB-n\fR \fIname\fR] [\fB-s\fR \fIstatistic\fR]
     [interval [count]]
.fi

.LP
.nf
\fBkstat\fR [\fB-Cjlpq\fR] [\fB-T\fR u | d ] [\fB-c\fR \fIclass\fR]
     [\fImodule\fR[:\fIinstance\fR[:\fIname\fR[:\fIstatistic\fR]]]]...
     [\fImodule\fR:\fIinstance\fR:\fIname\fR:\fIstatistic\fR]...
     [interval [count]]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBkstat\fR utility examines the available kernel statistics, or kstats, on
the system and reports those statistics which match the criteria specified on
the command line. Each matching statistic is printed with its module, instance,
and name fields, as well as its actual value.
.sp
.LP
Kernel statistics may be published by various kernel subsystems, such as
drivers or loadable modules; each kstat has a module field that denotes its
publisher. Since each module might have countable entities (such as multiple
disks associated with the \fBsd\fR(7D) driver) for which it wishes to report
statistics, the kstat also has an instance field to index the statistics for
each entity; kstat instances are numbered starting from zero. Finally, the
kstat is given a name unique within its module.
.sp
.LP
Each kstat may be a special kstat type, an array of name-value pairs, or raw
data. In the name-value case, each reported value is given a label, which we
refer to as the statistic. Known raw and special kstats are given statistic
labels for each of their values by \fBkstat\fR; thus, all published values can
be referenced as \fImodule\fR:\fIinstance\fR:\fIname\fR:\fIstatistic\fR.
.sp
.LP
When invoked without any module operands or options, kstat will match all
defined statistics on the system. Example invocations are provided below. All
times are displayed as fractional seconds since system boot.
.SH OPTIONS
.sp
.LP
The tests specified by the following options are logically ANDed, and all
matching kstats will be selected. A regular expression containing shell
metacharacters must be protected from the shell by enclosing it with the
appropriate quotes.
.sp
.LP
```

```
The argument for the \fB-c\fR, \fB-i\fR, \fB-m\fR, \fB-n\fR, and \fB-s\fR
options may be specified as a shell glob pattern, or a regular expression
enclosed in '/' characters.
.sp
.ne 2
.na
\fB\fB-C\fR\fR
.ad
.RS 16n
Displays output in parseable format with a colon as separator.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fIclass\fR\fR
.ad
.RS 16n
Displays only kstats that match the specified class. \fIclass\fR is a
kernel-defined string which classifies the "type" of the kstat.
.RE

.sp
.ne 2
.na
\fB\fB-i\fR \fIinstance\fR\fR
.ad
.RS 16n
Displays only kstats that match the specified instance.
.RE

.sp
.ne 2
.na
\fB\fB-j\fR\fR
.ad
.RS 16n
Displays output in JSON format.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.RS 16n
Lists matching kstat names without displaying values.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR \fImodule\fR\fR
.ad
.RS 16n
Displays only kstats that match the specified module.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR \fIname\fR\fR
.ad
.RS 16n
Displays only kstats that match the specified name.
.RE
```

```
.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.RS 16n
Displays output in parseable format. All example output in this document is
given in this format. If this option is not specified, \fBkstat\fR produces
output in a human-readable, table format.
.RE

.sp
.ne 2
.na
\fB\fB-q\fR\fR
.ad
.RS 16n
Displays no output, but return appropriate exit status for matches against
given criteria.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR \fIstatistic\fR\fR
.ad
.RS 16n
Displays only kstats that match the specified statistic.
.RE

.sp
.ne 2
.na
\fB\fB-T\fR d | u\fR
.ad
.RS 16n
Displays a time stamp before each statistics block, either in \fBdate\fR(1)
format (\fBd\fR) or as an alphanumeric representation of the value returned by
\fBtime\fR(2) (\fBu\fR).
.RE

.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fImodule\fR:\fIinstance\fR:\fIname\fR:\fIstatistic\fR\fR
.ad
.sp .6
.RS 4n
Alternate method of specifying module, instance, name, and statistic as
described above. Each of the module, instance, name, or statistic specifiers
may be a shell glob pattern or a regular expression enclosed by '/'
characters. It is possible to use both specifier types within a single operand.
Leaving a specifier empty is equivalent to using the '*' glob pattern for that
specifier.
.RE

.sp
.ne 2
.na
\fB\fIinterval\fR\fR
.ad
```

```
.sp .6
.RS 4n
The number of seconds between reports.
.RE

.sp
.ne 2
.na
\fB\fIcount\fR\fR
.ad
.sp .6
.RS 4n
The number of reports to be printed.
.RE

.SH EXAMPLES
.sp
.LP
In the following examples, all the command lines in a block produce the same
output, as shown immediately below. The exact statistics and values will of
course vary from machine to machine.
.LP
\fBExample 1 \fRUsing the \fBkstat\fR Command
.sp
.in +2
.nf
example$ \fBkstat -p -m unix -i 0 -n system_misc -s 'avenrun*'\fR
example$ \fBkstat -p -s 'avenrun*'\fR
example$ \fBkstat -p 'unix:0:system_misc:avenrun*'\fR
example$ \fBkstat -p ':::avenrun*'\fR
example$ \fBkstat -p ':::/^avenrun_[0-9]+min$/'\fR

unix:0:system_misc:avenrun_15min        3
unix:0:system_misc:avenrun_1min 4
unix:0:system_misc:avenrun_5min 2
.fi
.in -2
.sp

.LP
\fBExample 2 \fRUsing the \fBkstat\fR Command
.sp
.in +2
.nf
example$ \fBkstat -p -m cpu_stat -s 'intr*'\fR
example$ \fBkstat -p cpu_stat:::/^intr/\fR

cpu_stat:0:cpu_stat0:intr       29682330
cpu_stat:0:cpu_stat0:intrblk    87
cpu_stat:0:cpu_stat0:intrthread 15054222
cpu_stat:1:cpu_stat1:intr       426073
cpu_stat:1:cpu_stat1:intrblk    51
cpu_stat:1:cpu_stat1:intrthread 289668
cpu_stat:2:cpu_stat2:intr       134160
cpu_stat:2:cpu_stat2:intrblk    0
cpu_stat:2:cpu_stat2:intrthread 131
cpu_stat:3:cpu_stat3:intr       196566
cpu_stat:3:cpu_stat3:intrblk    30
cpu_stat:3:cpu_stat3:intrthread 59626
.fi
.in -2
.sp

.LP
\fBExample 3 \fRUsing the \fBkstat\fR Command
.sp
```

```
.in +2
.nf
example$ \fBkstat -p :::state ’:::avenrun*’\fR
example$ \fBkstat -p :::state :::/^avenrun/\fR

cpu_info:0:cpu_info0:state      on-line
cpu_info:1:cpu_info1:state      on-line
cpu_info:2:cpu_info2:state      on-line
cpu_info:3:cpu_info3:state      on-line
unix:0:system_misc:avenrun_15min        4
unix:0:system_misc:avenrun_1min 10
unix:0:system_misc:avenrun_5min 3
.fi
.in -2
.sp

.LP
\fBExample 4 \fRUsing the \fBkstat\fR Command
.sp
.in +2
.nf
example$ \fBkstat -p ’unix:0:system_misc:avenrun*’ 1 3\fR
unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21
.fi
.in -2
.sp

.LP
\fBExample 5 \fRUsing the \fBkstat\fR Command
.sp
.in +2
.nf
example$ \fBkstat -p -T d ’unix:0:system_misc:avenrun*’ 5 2\fR
Thu Jul 22 19:39:50 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
unix:0:system_misc:avenrun_5min 11

Thu Jul 22 19:39:55 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
unix:0:system_misc:avenrun_5min 11
.fi
.in -2
.sp

.LP
\fBExample 6 \fRUsing the \fBkstat\fR Command
.sp
.in +2
.nf
example$ \fBkstat -p -T u ’unix:0:system_misc:avenrun*’\fR
932668656
unix:0:system_misc:avenrun_15min        14
unix:0:system_misc:avenrun_1min 5
unix:0:system_misc:avenrun_5min 18
```

```
.fi
.in -2
.sp

.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 5n
One or more statistics were matched.
.RE

.sp
.ne 2
.na
\fB\fB1\fR\fR
.ad
.RS 5n
No statistics were matched.
.RE

.sp
.ne 2
.na
\fB\fB2\fR\fR
.ad
.RS 5n
Invalid command line options were specified.
.RE

.sp
.ne 2
.na
\fB\fB3\fR\fR
.ad
.RS 5n
A fatal error occurred.
.RE

.SH FILES
.sp
.ne 2
.na
\fB\fB/dev/kstat\fR\fR
.ad
.RS 14n
kernel statistics driver
.RE

.SH SEE ALSO
.sp
.LP
\fBdate\fR(1), \fBsh\fR(1), \fBtime\fR(2), \fBgmatch\fR(3GEN),
\fBkstat\fR(3KSTAT), \fBattributes\fR(5), \fBregex\fR(5), \fBkstat\fR(7D),
\fBsd\fR(7D), \fBkstat\fR(9S)
.SH NOTES
.sp
.LP
If the pattern argument contains glob or RE metacharacters which are also
shell metacharacters, it will be necessary to enclose the pattern with
appropriate shell quotes.
```