```
**********************************************************
    2660 Fri Jan 18 13:59:17 2013
new/usr/src/cmd/tail/extern.h
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
/*
 * Copyright (c) 1991, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */
/*
 * Copyright (c) 2013, Joyent, Inc. All Rights Reserved.
 */


#ifndef EXTERN_H
#define EXTERN_H

#include <sys/types.h>
#include <sys/stat.h>
#include <port.h>


#define WR(p, size) do { \
        if (write(STDOUT_FILENO, p, size) != (ssize_t)size) \
                oerr(); \
        } while (0)

#define TAILMAPLEN (4<<20)

struct mapinfo {
        off_t   mapoff;
        off_t   maxoff;
        size_t  maplen;
        char    *start;
        int     fd;
};

struct file_info {
        FILE *fp;
        char *file_name;
```



```
        struct stat st;
        file_obj_t fobj;
};

typedef struct file_info file_info_t;

enum STYLE { NOTSET = 0, FBYTES, FLINES, RBYTES, RLINES, REVERSE };

void follow(file_info_t *, enum STYLE, off_t);
void forward(FILE *, const char *, enum STYLE, off_t, struct stat *);
void reverse(FILE *, const char *, enum STYLE, off_t, struct stat *);

int bytes(FILE *, const char *, off_t);
int lines(FILE *, const char *, off_t);

void ierr(const char *);
void oerr(void);
int mapprint(struct mapinfo *, off_t, off_t);
int maparound(struct mapinfo *, off_t);

extern int Fflag, fflag, qflag, rflag, rval, no_files;

#endif  /* EXTERN_H */
```

```
**********************************************************
   10945 Fri Jan 18 13:59:17 2013
new/usr/src/cmd/tail/forward.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * Copyright (c) 1991, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * This code is derived from software contributed to Berkeley by
 * Edward Sze-Tyan Wang.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

/*
 * Copyright (c) 2013, Joyent, Inc. All rights reserved.
 * Solaris porting notes:  the original FreeBSD version made use of the
 *                         BSD kqueue event notification framework; this
 *                         was changed to use usleep()
 */

#include <sys/param.h>
#include <sys/mount.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/statfs.h>
#include <sys/statvfs.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/poll.h>
#include <port.h>
#include <err.h>
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <unistd.h>

#include "extern.h"
```



```
static void rlines(FILE *, const char *fn, off_t, struct stat *);
static int show(file_info_t *);
static void set_events(file_info_t *files);

/* defines for inner loop actions */
#define USE_SLEEP       0
#define USE_PORT        1
#define ADD_EVENTS      2

int port;
int action = USE_PORT;
int action = USE_SLEEP;

static const file_info_t *last;

/*
 * forward -- display the file, from an offset, forward.
 *
 * There are eight separate cases for this -- regular and non-regular
 * files, by bytes or lines and from the beginning or end of the file.
 *
 * FBYTES       byte offset from the beginning of the file
 *      REG     seek
 *      NOREG   read, counting bytes
 *
 * FLINES       line offset from the beginning of the file
 *      REG     read, counting lines
 *      NOREG   read, counting lines
 *
 * RBYTES       byte offset from the end of the file
 *      REG     seek
 *      NOREG   cyclically read characters into a wrap-around buffer
 *
 * RLINES
 *      REG     mmap the file and step back until reach the correct offset.
 *      NOREG   cyclically read lines into a wrap-around array of buffers
 */
void
forward(FILE *fp, const char *fn, enum STYLE style, off_t off, struct stat *sbp)
{
        int ch;

        switch (style) {
        case FBYTES:
                if (off == 0)
                        break;
                if (S_ISREG(sbp->st_mode)) {
                        if (sbp->st_size < off)
                                off = sbp->st_size;
                        if (fseeko(fp, off, SEEK_SET) == -1) {
                                ierr(fn);
                                return;
                        }
                } else while (off--)
                        if ((ch = getc(fp)) == EOF) {
                                if (ferror(fp)) {
                                        ierr(fn);
                                        return;
                                }
                                break;
                        }
                break;
        case FLINES:
                if (off == 0)
                        break;
```

```
123                 for (;;) {
124                         if ((ch = getc(fp)) == EOF) {
125                                 if (ferror(fp)) {
126                                         ierr(fn);
127                                         return;
128                                 }
129                                 break;
130                         }
131                         if (ch == '\n' && !--off)
132                                 break;
133                 }
134                 break;
135         case RBYTES:
136                 if (S_ISREG(sbp->st_mode)) {
137                         if (sbp->st_size >= off &&
138                             fseeko(fp, -off, SEEK_END) == -1) {
139                                 ierr(fn);
140                                 return;
141                         }
142                 } else if (off == 0) {
143                         while (getc(fp) != EOF)
144                                 ;
145                         if (ferror(fp)) {
146                                 ierr(fn);
147                                 return;
148                         }
149                 } else
150                         if (bytes(fp, fn, off))
151                                 return;
152                 break;
153         case RLINES:
154                 if (S_ISREG(sbp->st_mode))
155                         if (!off) {
156                                 if (fseeko(fp, (off_t)0, SEEK_END) == -1) {
157                                         ierr(fn);
158                                         return;
159                                 }
160                         } else
161                                 rlines(fp, fn, off, sbp);
162                 else if (off == 0) {
163                         while (getc(fp) != EOF)
164                                 ;
165                         if (ferror(fp)) {
166                                 ierr(fn);
167                                 return;
168                         }
169                 } else
170                         if (lines(fp, fn, off))
171                                 return;
172                 break;
173         default:
174                 break;
175         }

177         while ((ch = getc(fp)) != EOF)
178                 if (putchar(ch) == EOF)
179                         oerr();
180         if (ferror(fp)) {
181                 ierr(fn);
182                 return;
183         }
184         (void) fflush(stdout);
185 }
```

*unchanged_portion_omitted_*

```
264 static void
```



```
265 associate(file_info_t *file, boolean_t assoc)
266 {
267         char buf[64];

269         if (action != USE_PORT || file->fp == NULL)
270                 return;

272         if (!S_ISREG(file->st.st_mode)) {
273                 /*
274                  * For FIFOs, we use PORT_SOURCE_FD as our port event source.
275                  */
276                 if (assoc) {
277                         (void) port_associate(port, PORT_SOURCE_FD,
278                             fileno(file->fp), POLLIN, file);
279                 } else {
280                         (void) port_dissociate(port, PORT_SOURCE_FD,
281                             fileno(file->fp));
282                 }

284                 return;
285         }

287         bzero(&file->fobj, sizeof (file->fobj));

289         /*
290          * We pull a bit of a stunt here.  PORT_SOURCE_FILE only allows us to
291          * specify a file name -- not a file descriptor.  If we were to specify
292          * the name of the file to port_associate() and that file were moved
293          * aside, we would not be able to reassociate an event because we would
294          * not know a name that would resolve to the new file (indeed, there
295          * might not be such a name -- the file may have been unlinked).  But
296          * there _is_ a name that we know maps to the file and doesn't change:
297          * the name of the representation of the open file descriptor in /proc.
298          * We therefore associate with this name (and the underlying file),
299          * not the name of the file as specified at the command line.
300          */
301         (void) snprintf(buf,
302             sizeof (buf), "/proc/self/fd/%d", fileno(file->fp));

304         /*
305          * Note that portfs uses the address of the specified file_obj_t to
306          * tag an association; if one creates a different association with a
307          * (different) file_ob_t that happens to be at the same address,
308          * the first association will be implicitly removed.  To assure that
309          * each file has a disjoint file_obj_t, we allocate the memory for it
310          * in the file_info, not on the stack.
311          */
312         file->fobj.fo_name = buf;

314         if (assoc) {
315                 (void) port_associate(port, PORT_SOURCE_FILE,
316                     (uintptr_t)&file->fobj, FILE_MODIFIED | FILE_TRUNC, file);
317         } else {
318                 (void) port_dissociate(port, PORT_SOURCE_FILE,
319                     (uintptr_t)&file->fobj);
320         }
321 }

323 static void
324 set_events(file_info_t *files)
325 {
326         int i;
327         file_info_t *file;

329         for (i = 0, file = files; i < no_files; i++, file++) {
330                 if (! file->fp)
```

```
331                         continue;

333                 (void) fstat(fileno(file->fp), &file->st);

335                 associate(file, B_TRUE);
336         }
337 }

339 /*
340  * follow -- display the file, from an offset, forward.
341  *
342  */
343 void
344 follow(file_info_t *files, enum STYLE style, off_t off)
345 {
346         int active, ev_change, i, n = -1;
347         struct stat sb2;
348         file_info_t *file;
349         struct timespec ts;
350         port_event_t ev;

352         /* Position each of the files */

354         file = files;
355         active = 0;
356         n = 0;
357         for (i = 0; i < no_files; i++, file++) {
358                 if (file->fp) {
359                         active = 1;
360                         n++;
361                         if (no_files > 1 && !qflag)
362                                 (void) printf("\n==> %s <==\n",
363                                     file->file_name);
364                         forward(file->fp, file->file_name, style, off,
365                             &file->st);
366                         if (Fflag && fileno(file->fp) != STDIN_FILENO)
367                                 n++;
368                 }
369         }
370         if (!Fflag && !active)
371                 return;

373         last = --file;

375         if (action == USE_PORT &&
376             (stat("/proc/self/fd", &sb2) == -1 || !S_ISDIR(sb2.st_mode) ||
377             (port = port_create()) == -1))
378                 action = USE_SLEEP;

380         set_events(files);

382         for (;;) {
383                 ev_change = 0;
384                 if (Fflag) {
385                         for (i = 0, file = files; i < no_files; i++, file++) {
386                                 if (!file->fp) {
387                                         file->fp = fopen(file->file_name, "r");
388                                         if (file->fp != NULL &&
389                                             fstat(fileno(file->fp), &file->st)
390                                             == -1) {
391                                                 (void) fclose(file->fp);
392                                                 file->fp = NULL;
393                                         }
394                                         if (file->fp != NULL)
395                                                 ev_change++;
396                                         continue;
```



```
397                                 }
398                                 if (fileno(file->fp) == STDIN_FILENO)
399                                         continue;
400                                 if (stat(file->file_name, &sb2) == -1) {
401                                         if (errno != ENOENT)
402                                                 ierr(file->file_name);
403                                         (void) show(file);
404                                         (void) fclose(file->fp);
405                                         file->fp = NULL;
406                                         ev_change++;
407                                         continue;
408                                 }

410                                 if (sb2.st_ino != file->st.st_ino ||
411                                     sb2.st_dev != file->st.st_dev ||
412                                     sb2.st_nlink == 0) {
413                                         (void) show(file);
414                                         associate(file, B_FALSE);
415                                         file->fp = freopen(file->file_name, "r",
416                                             file->fp);
417                                         if (file->fp != NULL) {
346                                         if (file->fp != NULL)
418                                                 (void) memcpy(&file->st, &sb2,
419                                                     sizeof (struct stat));
420                                         } else if (errno != ENOENT)
349                                         else if (errno != ENOENT)
421                                                 ierr(file->file_name);
422                                         ev_change++;
423                                 }
424                         }
425                 }

427                 for (i = 0, file = files; i < no_files; i++, file++)
428                         if (file->fp && !show(file))
429                                 ev_change++;

431                 if (ev_change)
432                         set_events(files);

434                 switch (action) {
435                 case USE_PORT:
436                         ts.tv_sec = 1;
437                         ts.tv_nsec = 0;

439                         /*
440                          * In the -F case we set a timeout to ensure that
441                          * we re-stat the file at least once every second.
442                          */
443                         n = port_get(port, &ev, Fflag ? &ts : NULL);

445                         if (n == 0) {
446                                 file = (file_info_t *)ev.portev_user;
447                                 associate(file, B_TRUE);

449                                 if (ev.portev_events & FILE_TRUNC)
450                                         (void) fseek(file->fp, 0, SEEK_SET);
451                         }

453                         break;

455                 case USE_SLEEP:
456                         (void) usleep(250000);
457                         break;
458                 }
459         }
460 }
```

```
**********************************************************
    7203 Fri Jan 18 13:59:17 2013
new/usr/src/cmd/tail/tests/tailtests.sh
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
#!/bin/bash
#
#
# 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 is of the CDDL is also available via the Internet
# at http://www.illumos.org/license/CDDL.
#

#
# Copyright 2010 Chris Love.  All rights reserved.
# Copyright (c) 2013, Joyent, Inc. All rights reserved.
#

checktest()
{
        local actual=$1
        local output=$2
        local test=$3

        if [[ "$actual" != "$output" ]]; then
                echo "$CMD: test $test: FAIL"
                echo -e "$CMD: test $test: expected output:\n$o"
                echo -e "$CMD: test $test: actual output:\n$a"
        else
                echo "$CMD: test $test: pass"
        fi
}

#
# Test cases for 'tail', some based on CoreUtils test cases (validated
# with legacy Solaris 'tail' and/or xpg4 'tail').  Note that this is designed
# to be able to run on BSD systems as well to check our behavior against
# theirs (some behavior that is known to be idiosyncratic to illumos is
# skipped on non-illumos systems).
# with legacy Solaris 'tail' and/or xpg4 'tail')
#
PROG=/usr/bin/tail
CMD=`basename $0`
DIR=""

while [[ $# -gt 0 ]]; do
        case $1 in
case $1 in
            -x)
                PROG=/usr/xpg4/bin/tail
                shift
                ;;
            -o)
                PROG=$2
                shift 2
                ;;
            -d)
                DIR=$2
                shift 2
                ;;
            *)
                echo "Usage: tailtests.sh" \
                    "[-x][-o <override tail executable>]" \
                    "[-d <override output directory>]"
    -?)
        echo "Usage: tailtests.sh [-x][-o <override tail executable>]"
                exit 1
                ;;
        esac
done
esac

#
# Shut bash up upon receiving a term so we can drop it on our children
# without disrupting the output.
#
trap "exit 0" TERM
echo "Using $PROG"

echo "$CMD: program is $PROG"

if [[ $DIR != "" ]]; then
        echo "$CMD: directory is $DIR"
fi

o=`echo -e "bcd"`
a=`echo -e "abcd" | $PROG +2c`
checktest "$a" "$o" 1
[[ "$a" != "$o" ]] && echo "Fail test 1 - $a"

o=`echo -e ""`
a=`echo "abcd" | $PROG +8c`
checktest "$a" "$o" 2
[[ "$a" != "$o" ]] && echo "Fail test 2 - $a"

o=`echo -e "abcd"`
a=`echo "abcd" | $PROG -9c`
checktest "$a" "$o" 3
[[ "$a" != "$o" ]] && echo "Fail test 3 - $a"

o=`echo -e "x"`
a=`echo -e "x" | $PROG -1l`
checktest "$a" "$o" 4
[[ "$a" != "x" ]] && echo "Fail test 4 - $a"

o=`echo -e "\n"`
a=`echo -e "x\ny\n" | $PROG -1l`
checktest "$a" "$o" 5
[[ "$a" != "$o" ]] && echo "Fail test 5 - $a"

o=`echo -e "y\n"`
a=`echo -e "x\ny\n" | $PROG -2l`
checktest "$a" "$o" 6
[[ "$a" != "$o" ]] && echo "Fail test 6 - $a"

o=`echo -e "y"`
a=`echo -e "x\ny" | $PROG -1l`
checktest "$a" "$o" 7
[[ "$a" != "$o" ]] && echo "Fail test 7 - $a"

o=`echo -e "x\ny\n"`
a=`echo -e "x\ny\n" | $PROG +1l`
checktest "$a" "$o" 8
[[ "$a" != "$o" ]] && echo "Fail test 8 - $a"

o=`echo -e "y\n"`
```

```
a=‘echo -e "x\ny\n" | $PROG +2l‘
checktest "$a" "$o" 9
[[ "$a" != "$o" ]] && echo "Fail test 9 - $a"

o=‘echo -e "x"‘
a=‘echo -e "x" | $PROG -1‘
checktest "$a" "$o" 10
[[ "$a" != "$o" ]] && echo "Fail test 10 - $a"

o=‘echo -e "\n"‘
a=‘echo -e "x\ny\n" | $PROG -1‘
checktest "$a" "$o" 11
[[ "$a" != "$o" ]] && echo "Fail test 11 - $a"

o=‘echo -e "y\n"‘
a=‘echo -e "x\ny\n" | $PROG -2‘
checktest "$a" "$o" 12
[[ "$a" != "$o" ]] && echo "Fail test 12 - $a"

o=‘echo -e "y"‘
a=‘echo -e "x\ny" | $PROG -1‘
checktest "$a" "$o" 13
[[ "$a" != "$o" ]] && echo "Fail test 13 - $a"

o=‘echo -e "x\ny\n"‘
a=‘echo -e "x\ny\n" | $PROG +1‘
checktest "$a" "$o" 14
[[ "$a" != "$o" ]] && echo "Fail test 14 - $a"

o=‘echo -e "y\n"‘
a=‘echo -e "x\ny\n" | $PROG +2‘
checktest "$a" "$o" 15
[[ "$a" != "$o" ]] && echo "Fail test 15 - $a"

# For compatibility with Legacy Solaris tail this should also work as ‘+c‘
o=‘echo -e "yyz"‘
a=‘echo -e "xyyyyyyyyyyz" | $PROG +10c‘
checktest "$a" "$o" 16
[[ "$a" != "$o" ]] && echo "Fail test 16 - $a"

o=‘echo -e "yyz"‘
a=‘echo -e "xyyyyyyyyyyz" | $PROG +c‘
[[ "$a" != "$o" ]] && echo "Fail test 16a - $a"


# For compatibility with Legacy Solaris tail this should also work as ‘+l‘
o=‘echo -e "y\ny\nz"‘
a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG +10l‘
checktest "$a" "$o" 17
[[ "$a" != "$o" ]] && echo "Fail test 17 - $a"

o=‘echo -e "y\ny\nz"‘
a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG +l‘
[[ "$a" != "$o" ]] && echo "Fail test 17a - $a"


# For compatibility with Legacy Solaris tail this should also work as ‘-l‘
o=‘echo -e "y\ny\ny\ny\ny\ny\ny\ny\ny\nz"‘
a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG -10l‘
checktest "$a" "$o" 18
[[ "$a" != "$o" ]] && echo "Fail test 18 - $a"

#
# For reasons that are presumably as accidental as they are ancient, legacy
# (and closed) Solaris tail(1) allows +c, +l and -l to be aliases for +10c,
# +10l and -10l, respectively.  If we are on SunOS, verify that this silly
```



```
# behavior is functional.
#
if [[ ‘uname -s‘ == "SunOS" ]]; then
        o=‘echo -e "yyz"‘
        a=‘echo -e "xyyyyyyyyyyz" | $PROG +c‘
        checktest "$a" "$o" 16a
o=‘echo -e "y\ny\ny\ny\ny\ny\ny\ny\nz"‘
a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG -l‘
[[ "$a" != "$o" ]] && echo "Fail test 18a - $a"

        o=‘echo -e "y\ny\nz"‘
        a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG +l‘
        checktest "$a" "$o" 17a

        o=‘echo -e "y\ny\ny\ny\ny\ny\ny\ny\ny\nz"‘
        a=‘echo -e "x\ny\ny\ny\ny\ny\ny\ny\ny\ny\ny\nz" | $PROG -l‘
        checktest "$a" "$o" 18a
fi

o=‘echo -e "c\nb\na"‘
a=‘echo -e "a\nb\nc" | $PROG -r‘
checktest "$a" "$o" 19
[[ "$a" != "$o" ]] && echo "Fail test 19 - $a"

#
# Now we want to do a series of follow tests.
#
if [[ $DIR == "" ]]; then
        tdir=$(mktemp -d -t tailtest.XXXXXXXX || exit 1)
else
        tdir=$(mktemp -d $DIR/tailtest.XXXXXXXX || exit 1)
fi

follow=$tdir/follow
moved=$tdir/follow.moved
out=$tdir/out
echo "Completed"

#
# First, verify that following works in its most basic sense.
#
echo -e "a\nb\nc" > $follow
$PROG -f $follow > $out 2> /dev/null &
child=$!
sleep 2
echo -e "d\ne\nf" >> $follow
sleep 1
kill $child
sleep 1
exit 0

o=‘echo -e "a\nb\nc\nd\ne\nf\n"‘
a=‘cat $out‘
checktest "$a" "$o" 20
rm $follow

#
# Now verify that following correctly follows the file being moved.
#
echo -e "a\nb\nc" > $follow
$PROG -f $follow > $out 2> /dev/null &
child=$!
sleep 2
mv $follow $moved

echo -e "d\ne\nf" >> $moved
```

```
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\n"`
a=`cat $out`
checktest "$a" "$o" 21
rm $moved

#
# And now truncation with the new offset being less than the old offset.
#
echo -e "a\nb\nc" > $follow
$PROG -f $follow > $out 2> /dev/null &
child=$!
sleep 2
echo -e "d\ne\nf" >> $follow
sleep 1
echo -e "g\nh\ni" > $follow
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\ng\nh\ni\n"`
a=`cat $out`
checktest "$a" "$o" 22
rm $follow

#
# And truncation with the new offset being greater than the old offset.
#
echo -e "a\nb\nc" > $follow
sleep 1
$PROG -f $follow > $out 2> /dev/null &
child=$!
sleep 2
echo -e "d\ne\nf" >> $follow
sleep 1
echo -e "g\nh\ni\nj\nk\nl\nm\no\np\nq" > $follow
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\ng\nh\ni\nj\nk\nl\nm\no\np\nq"`
a=`cat $out`
checktest "$a" "$o" 23
rm $follow

#
# Verify that we can follow the moved file and correctly see a truncation.
#
echo -e "a\nb\nc" > $follow
$PROG -f $follow > $out 2> /dev/null &
child=$!
sleep 2
mv $follow $moved

echo -e "d\ne\nf" >> $moved
sleep 1
echo -e "g\nh\ni\nj\nk\nl\nm\no\np\nq" > $moved
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\ng\nh\ni\nj\nk\nl\nm\no\np\nq"`
a=`cat $out`
```



```
checktest "$a" "$o" 24
rm $moved

#
# Verify that capital-F follow properly deals with truncation
#
echo -e "a\nb\nc" > $follow
$PROG -F $follow > $out 2> /dev/null &
child=$!
sleep 2
echo -e "d\ne\nf" >> $follow
sleep 1
echo -e "g\nh\ni" > $follow
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\ng\nh\ni\n"`
a=`cat $out`
checktest "$a" "$o" 25
rm $follow

#
# Verify that capital-F follow _won't_ follow the moved file and will
# correctly deal with recreation of the original file.
#
echo -e "a\nb\nc" > $follow
$PROG -F $follow > $out 2> /dev/null &
child=$!
sleep 2
mv $follow $moved

echo -e "x\ny\nz" >> $moved
echo -e "d\ne\nf" > $follow
sleep 1
kill $child
sleep 1

o=`echo -e "a\nb\nc\nd\ne\nf\n"`
a=`cat $out`
checktest "$a" "$o" 26
rm $moved

#
# Verify that following two files works.
#
echo -e "one" > $follow
echo -e "two" > $moved
$PROG -f $follow $moved > $out 2> /dev/null &
child=$!
sleep 2
echo -e "three" >> $follow
sleep 1
echo -e "four" >> $moved
sleep 1
echo -e "five" >> $follow
sleep 1
kill $child
sleep 1

# There is a bug where the content comes before the header lines,
# where rlines/mapprint happens before the header.  A pain to fix.
# In this test, just make sure we see both files change.
o="one

==> $follow <==
```

```
348 two

350 ==> $moved <==

352 ==> $follow <==
353 three

355 ==> $moved <==
356 four

358 ==> $follow <==
359 five"
360 a=‘cat $out‘
361 checktest "$a" "$o" 27
362 rm $follow $moved

364 echo "$CMD: completed"

366 exit $errs

138 # Template for additional test cases
139 #o=‘echo -e ""‘
140 #a=‘echo -e "" | $PROG ‘
141 #[[ "$a" != "$o" ]] && echo "Fail test  - $a"
```

```
***********************************************************
    8136 Fri Jan 18 13:59:17 2013
new/usr/src/man/man1/tail.1
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
***********************************************************
'\" te
.\"  Copyright 1989 AT&T  Copyright (c) 1992, X/Open Company Limited  All Rights
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.\" Copyright (c) 2013, Joyent, Inc. All Rights Reserved.
.TH TAIL 1 "Jan 18, 2013"
.TH TAIL 1 "Jul 13, 2005"
.SH NAME
tail \- deliver the last part of a file
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/tail\fR [\(+-s \fInumber\fR [lbcr]] [\fIfile\fR]
.fi

.LP
.nf
\fB/usr/bin/tail\fR [\fB-lbcr\fR] [\fIfile\fR]
.fi

.LP
.nf
\fB/usr/bin/tail\fR [\(+- \fInumber\fR [lbcfF]] [\fIfile\fR]
\fB/usr/bin/tail\fR [\(+- \fInumber\fR [lbcf]] [\fIfile\fR]
.fi

.LP
.nf
\fB/usr/bin/tail\fR [\fB-lbcfF\fR] [\fIfile\fR]
\fB/usr/bin/tail\fR [\fB-lbcf\fR] [\fIfile\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/tail\fR [\fB-f\fR | \fB-r\fR] [\fB-c\fR \fInumber\fR | \fB-n\fR
.fi

.LP
.nf
\fB/usr/xpg4/bin/tail\fR [\(+- \fInumber\fR [l | b | c] [f]] [\fIfile\fR]
.fi

.LP
.nf
\fB/usr/xpg4/bin/tail\fR [\(+- \fInumber\fR [l] [f | r]] [\fIfile\fR]
.fi

.SH DESCRIPTION
.sp
.LP
The \fBtail\fR utility copies the named file to the standard output beginning
at a designated place. If no file is named, the standard input is used.
.sp
.LP
Copying begins at a point in the file indicated by the \fB-c\fR\fInumber\fR,
```



```
\fB-n\fR\fInumber\fR, or \fB\(+-\fR\fInumber\fR options (if \fB+\fR\fInumber\fR
is specified, begins at distance number from the beginning; if
\fB-\fR\fInumber\fR is specified, from the end of the input; if \fInumber\fR is
\fINULL\fR, the value \fB10\fR is assumed). \fInumber\fR is counted in units of
lines or byte according to the \fB-c\fR \fB or \fR \fB-n\fR options, or lines,
blocks, or bytes, according to the appended option \fBl\fR, \fBb\fR, or
\fBc\fR. When no units are specified, counting is by lines.
.SH OPTIONS
.sp
.LP
The following options are supported for both \fB/usr/bin/tail\fR and
\fB/usr/xpg4/bin/tail\fR. The \fB-r\fR and \fB-f\fR options are mutually
exclusive. If both are specified on the command line, the \fB-f\fR option is
ignored.
.sp
.ne 2
.na
\fB\fB-b\fR \fR
.ad
.RS 7n
Units of blocks.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR \fR
.ad
.RS 7n
Units of bytes.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR \fR
.ad
.RS 7n
Follow. If the input-file is not a pipe, \fBtail\fR does not terminate after
the last line of the input-file has been copied, but enters an endless loop,
wherein it watches the file for modifications and attempts to read and copy
further records from the input-file. Thus it can be used to monitor the growth
of a file that is being written by some other process. If the watched file is
truncated \fBtail\fR will begin reading records from the start of the file.
Follow. If the input-file is not a pipe, the program does not terminate after
the line of the input-file has been copied, but enters an endless loop, wherein
it sleeps for a second and then attempts to read and copy further records from
the input-file. Thus it can be used to monitor the growth of a file that is
being written by some other process.
.RE

.sp
.ne 2
.na
\fB\fB-F\fR \fR
.ad
.RS 7n
Follow named file. Operates as with \fB-f\fR, except that if the file is moved
(e.g. if a watched log file is rotated) \fBtail\fR will close the original file
and begin reading records from the start of the file with the specified name
if and when that file is recreated.
.RE

.sp
.ne 2
.na
```

```
\fB\fB-l\fR \fR
.ad
.RS 7n
Units of lines.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR \fR
.ad
.RS 7n
Reverse. Copies lines from the specified starting point in the file in reverse
order. The default for \fBr\fR is to print the entire file in reverse order.
.RE

.SS "/usr/xpg4/bin/tail"
.sp
.LP
The following options are supported for \fB/usr/xpg4/bin/tail\fR only:
.sp
.ne 2
.na
\fB\fB-c\fR \fInumber\fR \fR
.ad
.RS 14n
The \fInumber\fR option-argument must be a decimal integer whose sign affects
the location in the file, measured in bytes, to begin the copying:
.sp
.ne 2
.na
\fB\fB+\fR \fR
.ad
.RS 9n
Copying starts relative to the beginning of the file.
.RE

.sp
.ne 2
.na
\fB\fB\(mi\fR \fR
.ad
.RS 9n
Copying starts relative to the end of the file.
.RE

.sp
.ne 2
.na
\fBnone\fR
.ad
.RS 9n
Copying starts relative to the end of the file.
.RE

The origin for counting is 1; that is, \fB\fR\fB-c\fR\fB+1\fR represents the
first byte of the file, \fB\fR\fB-c\fR\fB\(mi1\fR the last.
.RE

.sp
.ne 2
.na
\fB\fB-n\fR \fInumber\fR \fR
.ad
.RS 14n
Equivalent to \fB-c\fR\fInumber,\fR except the starting location in the file is
```



```
measured in lines instead of bytes. The origin for counting is \fB1\fR. That
is, \fB-n\fR\fB+1\fR represents the first line of the file, \fB-n\fR\fB\(mi1\fR
the last.
.RE

.SH OPERANDS
.sp
.LP
The following operand is supported:
.sp
.ne 2
.na
\fB\fIfile\fR \fR
.ad
.RS 9n
A path name of an input file. If no \fIfile\fR operands are specified, the
standard input is used.
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBtail\fR when
encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
.SH EXAMPLES
.LP
\fBExample 1 \fRUsing the tail Command
.sp
.LP
The following command prints the last ten lines of the file \fBfred\fR,
followed by any lines that are appended to \fBfred\fR between the time
\fBtail\fR is initiated and killed.

.sp
.in +2
.nf
example% \fBtail -f fred\fR
.fi
.in -2
.sp

.sp
.LP
The next command prints the last 15 bytes of the file \fBfred\fR, followed by
any lines that are appended to \fBfred\fR between the time \fBtail\fR is
initiated and killed:

.sp
.in +2
.nf
example% \fBtail -15cf fred\fR
.fi
.in -2
.sp

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBtail\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
```

```
.ne 2
.na
\fB\fB0\fR \fR
.ad
.RS 7n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fB>0\fR \fR
.ad
.RS 7n
An error occurred.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.SS "/usr/bin/tail"
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Enabled
.TE

.SS "/usr/xpg4/bin/tail"
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Enabled
_
Interface Stability     Standard
.TE

.SH SEE ALSO
.sp
.LP
\fBcat\fR(1), \fBhead\fR(1), \fBmore\fR(1), \fBpg\fR(1), \fBdd\fR(1M),
\fBattributes\fR(5), \fBenviron\fR(5), \fBlargefile\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
Piped tails relative to the end of the file are stored in a buffer, and thus
are limited in length. Various kinds of anomalous behavior can happen with
character special files.
```

```
**********************************************************
   13130 Fri Jan 18 13:59:18 2013
new/usr/src/man/man3c/port_associate.3c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
'\" te
.\" Copyright (c) 2007, Sun Microsystems, Inc. All Rights Reserved.
.\" Copyright (c) 2013, Joyent, Inc. All Rights Reserved.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH PORT_ASSOCIATE 3C "Jan 18, 2013"
.TH PORT_ASSOCIATE 3C "Nov 9, 2007"
.SH NAME
port_associate, port_dissociate \- associate or dissociate the object with the
port
.SH SYNOPSIS
.LP
.nf
#include <port.h>

\fBint\fR \fBport_associate\fR(\fBint\fR \fIport\fR, \fBint\fR \fIsource\fR, \fB
     \fBint\fR \fIevents\fR, \fBvoid *\fR\fIuser\fR);
.fi

.LP
.nf
\fBint\fR \fBport_dissociate\fR(\fBint\fR \fIport\fR, \fBint\fR \fIsource\fR, \f
.fi

.SH DESCRIPTION
.sp
.LP
The \fBport_associate()\fR function associates specific \fIevents\fR of a given
\fIobject\fR with a \fIport\fR.  Only objects associated with a particular port
are able to generate events that can be retrieved using \fBport_get\fR(3C) or
\fBport_getn\fR(3C). The delivery event has its \fBportev_user\fR member set to
the value specified in the \fIuser\fR parameter. If the specified object is
already associated with the specified port, the \fBport_associate()\fR function
serves to update the \fIevents\fR and \fIuser\fR arguments of the association.
The \fBport_dissociate()\fR function removes the association of an object with
a port.
.sp
.LP
The objects that can be associated with a port by way of the
\fBport_associate()\fR function are objects of type \fBPORT_SOURCE_FD\fR and
\fBPORT_SOURCE_FILE\fR. Objects of other types have type-specific association
mechanisms. A \fBport_notify_t\fR structure, defined in \fB<port.h>\fR, is used
to specify the event port and an application-defined cookie to associate with
these event sources. See \fBport_create\fR(3C) and \fBsignal.h\fR(3HEAD).
.sp
.LP
The \fBport_notify_t\fR structure contains the following members:
.sp
.in +2
.nf
int       portntfy_port;  /* bind request(s) to port */
void      *portntfy_user; /* user defined cookie */
.fi
.in -2

.sp
.LP
Objects of type \fBPORT_SOURCE_FD\fR are file descriptors. The event types for
\fBPORT_SOURCE_FD\fR objects are described in \fBpoll\fR(2). At most one event
```



```
notification will be generated per associated file descriptor.  For example, if
a file descriptor is associated with a port for the \fBPOLLRDNORM\fR event and
data is available on the file descriptor at the time the \fBport_associate()\fR
function is called, an event is immediately sent to the port. If data is not
yet available, one event is sent to the port when data first becomes available.
.sp
.LP
When an event for a \fBPORT_SOURCE_FD\fR object is retrieved, the object no
longer has an association with the port.  The event can be processed without
the possibility that another thread can retrieve a subsequent event for the
same object.  After processing of the file descriptor is completed, the
\fBport_associate()\fR function can be called to reassociate the object with
the port.
.sp
.LP
Objects of type \fBPORT_SOURCE_FILE\fR are pointer to the structure
\fBfile_obj\fR defined in \fB<sys/port.h>\fR. This event source provides event
notification when the specified file/directory is accessed, modified,
truncated or when its status changes. The path name of the file/directory to
be watched is passed in the \fBstruct file_obj\fR along with the \fBaccess\fR,
\fBmodification\fR, and \fBchange\fR time stamps acquired from a \fBstat\fR(2)
call. If the file name is a symbolic link, it is followed by default. The
\fBFILE_NOFOLLOW\fR needs to be passed in along with the specified events if
the symbolic link itself needs to be watched and \fBlstat()\fR needs to be
used to get the file status of the symbolic link file.
notification when the specified file/directory is accessed or modified or when
its status changes. The path name of the file/directory to be watched is passed
in the \fBstruct file_obj\fR along with the \fBaccess\fR, \fBmodification\fR,
and \fBchange\fR time stamps acquired from a \fBstat\fR(2) call. If the file
name is a symbolic links, it is followed by default. The \fBFILE_NOFOLLOW\fR
needs to be passed in along with the specified events if the symbolic link
itself needs to be watched and \fBlstat()\fR needs to be used to get the file
status of the symbolic link file.
.sp
.LP
The \fBstruct file_obj\fR contains the following elements:
.sp
.in +2
.nf
timestruc_t     fo_atime;  /* Access time from stat() */
timestruc_t     fo_atime;  /* Access time got from stat() */
timestruc_t     fo_mtime;  /* Modification time from stat() */
timestruc_t     fo_ctime;  /* Change time from stat() */
char            *fo_name;  /* Pointer to a null terminated path name */
.fi
.in -2

.sp
.LP
At the time the \fBport_associate()\fR function is called, the time stamps
passed in the structure \fBfile_obj\fR are compared with the file or
directory's current time stamps and, if there has been a change, an event is
immediately sent to the port. If not, an event will be sent when such a change
occurs.
.sp
.LP
The event types that can be specified at \fBport_associate()\fR time for
\fBPORT_SOURCE_FILE\fR are \fBFILE_ACCESS\fR, \fBFILE_MODIFIED\fR,
\fBFILE_ATTRIB\fR, and \fbFILE_TRUNC\fR. The first three of these correspond
to the three time stamps: an \fBfo_atime\fR change results in the
\fBFILE_ACCESS\fR event, an \fBfo_mtime\fR change results in the
\fBFILE_MODIFIED\fR event, and an \fBfo_ctime\fR change results in the
\fBFILE_ATTRIB\fR event. If the operation that induced the time stamp update
also truncated the file, \fBFILE_TRUNC\fR will be set in the resulting event.
\fBPORT_SOURCE_FILE\fR are \fBFILE_ACCESS\fR, \fBFILE_MODIFIED\fR, and
\fBFILE_ATTRIB\fR, corresponding to the three time stamps. An \fBfo_atime\fR
```

```
109 change results in the \fBFILE_ACCESS\fR event, an \fBfo_mtime\fR change results
110 in the \fBFILE_MODIFIED\fR event, and an \fBfo_time\fR change results in the
111 \fBFILE_ATTRIB\fR event.
115 .sp
116 .LP
117 The following exception events are delivered when they occur. These event types
118 cannot be filtered.
119 .sp
120 .in +2
121 .nf
122 FILE_DELETE       /* Monitored file/directory was deleted */
123 FILE_RENAME_TO    /* Monitored file/directory was renamed */
124 FILE_RENAME_FROM  /* Monitored file/directory was renamed */
125 UNMOUNTED         /* Monitored file system got unmounted */
126 MOUNTEDOVER       /* Monitored file/directory was mounted over */
127 .fi
128 .in -2

130 .sp
131 .LP
132 At most one event notification will be generated per associated \fBfile_obj\fR.
133 When the event for the associated \fBfile_obj\fR is retrieved, the object is no
134 longer associated with the port. The event can be processed without the
135 possibility that another thread can retrieve a subsequent event for the same
136 object. The \fBport_associate()\fR can be called to reassociate the
137 \fBfile_obj\fR object with the port.
138 .sp
139 .LP
140 The association is also removed if the port gets closed or when
141 \fBport_dissociate()\fR is called.
142 .sp
143 .LP
144 The parent and child processes are allowed to retrieve events from file
145 descriptors shared after a call to \fBfork\fR(2). The process performing the
146 first association with a port (parent or child process) is designated as the
147 owner of the association. Only the owner of an association is allowed to
148 dissociate the file descriptor from a port. The association is removed if the
149 owner of the association closes the port .
150 .sp
151 .LP
152 On NFS file systems, events from only the client side (local)
153 access/modifications to files or directories will be delivered.
154 .SH RETURN VALUES
155 .sp
156 .LP
157 Upon succesful completion, 0 is returned. Otherwise, \(mi1 is returned and
158 \fBerrno\fR is set to indicate the error.
159 .SH ERRORS
160 .sp
161 .LP
162 The \fBport_associate()\fR and \fBport_dissociate()\fR functions will fail if:
163 .sp
164 .ne 2
165 .na
166 \fB\fBEBADF\fR\fR
167 .ad
168 .RS 10n
169 The \fIport\fR identifier is not valid.
170 .RE

172 .sp
173 .ne 2
174 .na
175 \fB\fBEBADFD\fR\fR
176 .ad
177 .RS 10n
```



```
178 The \fIsource\fR argument is of type \fBPORT_SOURCE_FD\fR and the object
179 argument is not a valid file descriptor.
180 .RE

182 .sp
183 .ne 2
184 .na
185 \fB\fBEINVAL\fR\fR
186 .ad
187 .RS 10n
188 The \fIsource\fR argument is not valid.
189 .RE

191 .sp
192 .LP
193 The \fBport_associate()\fR function will fail if:
194 .sp
195 .ne 2
196 .na
197 \fB\fBEACCES\fR\fR
198 .ad
199 .RS 11n
200 The source argument is \fBPORT_SOURCE_FILE\fR and, Search permission is denied
201 on a component of path prefix or the file exists and the permissions,
202 corresponding to the events argument, are denied.
203 .RE

205 .sp
206 .ne 2
207 .na
208 \fB\fBEAGAIN\fR\fR
209 .ad
210 .RS 11n
211 The maximum number of objects associated with the port was exceeded. The
212 maximum allowable number of events or association of objects per port is the
213 minimum value of the \fBprocess.max-port-events\fR resource control at the time
214 \fBport_create\fR(3C) was used to create the port. See \fBsetrctl\fR(2) and
215 \fBrctladm\fR(1M) for information on using resource controls.
216 .sp
217 The number of objects associated with a port is composed of all supported
218 resource types. Some of the source types do not explicitly use the
219 \fBport_associate()\fR function.
220 .RE

222 .sp
223 .ne 2
224 .na
225 \fB\fBENOENT\fR\fR
226 .ad
227 .RS 11n
228 The source argument is \fBPORT_SOURCE_FILE\fR and the file does not exist or
229 the path prefix does not exist or the path points to an empty string.
230 .RE

232 .sp
233 .ne 2
234 .na
235 \fB\fBENOMEM\fR\fR
236 .ad
237 .RS 11n
238 The physical memory limits of the system have been exceeded.
239 .RE

241 .sp
242 .ne 2
243 .na
```

```
 244 \fB\fBENOTSUP\fR\fR
 245 .ad
 246 .RS 11n
 247 The source argument is \fBPORT_SOURCE_FILE\fR and the file system on which the
 248 specified file resides, does not support watching for file events
 249 notifications.
 250 .RE

 252 .sp
 253 .LP
 254 The \fBport_dissociate()\fR function will fail if:
 255 .sp
 256 .ne 2
 257 .na
 258 \fB\fBEACCES\fR\fR
 259 .ad
 260 .RS 10n
 261 The process is not the owner of the association.
 262 .RE

 264 .sp
 265 .ne 2
 266 .na
 267 \fB\fBENOENT\fR\fR
 268 .ad
 269 .RS 10n
 270 The specified object is not associated with the port.
 271 .RE

 273 .SH EXAMPLES
 274 .LP
 275 \fBExample 1 \fRRetrieve data from a pipe file descriptor.
 276 .sp
 277 .LP
 278 The following example retrieves data from a pipe file descriptor.

 280 .sp
 281 .in +2
 282 .nf
 283 #include <port.h>

 285 int               port;
 286 int               fd;
 287 int               error;
 288 int               index;
 289 void              *mypointer;
 290 port_event_t      pev;
 291 struct timespec_t timeout;
 292 char              rbuf[STRSIZE];
 293 int               fds[MAXINDEX];

 295 /* create a port */
 296 port = port_create();

 298 for (index = 0; index < MAXINDEX; index++) {
 299     error = mkfifo(name[index], S_IRWXU | S_IRWXG | S_IRWXO);
 300     if (error)
 301             /* handle error code */
 302     fds[index] = open(name[index], O_RDWR);

 304     /* associate pipe file descriptor with the port */
 305     error = port_associate(port, PORT_SOURCE_FD, fds[index],
 306         POLLIN, mypointer);
 307 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
  171710 Fri Jan 18 13:59:18 2013
new/usr/src/uts/common/fs/nfs/nfs3_vnops.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * 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 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 *      Copyright (c) 1983,1984,1985,1986,1987,1988,1989 AT&T.
 *      All rights reserved.
 */

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

#include <sys/param.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <sys/cred.h>
#include <sys/time.h>
#include <sys/vnode.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/file.h>
#include <sys/filio.h>
#include <sys/uio.h>
#include <sys/buf.h>
#include <sys/mman.h>
#include <sys/pathname.h>
#include <sys/dirent.h>
#include <sys/debug.h>
#include <sys/vmsystm.h>
#include <sys/fcntl.h>
#include <sys/flock.h>
#include <sys/swap.h>
#include <sys/errno.h>
#include <sys/strsubr.h>
#include <sys/sysmacros.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/pathconf.h>
```



```
#include <sys/utsname.h>
#include <sys/dnlc.h>
#include <sys/acl.h>
#include <sys/systeminfo.h>
#include <sys/atomic.h>
#include <sys/policy.h>
#include <sys/sdt.h>
#include <sys/zone.h>

#include <rpc/types.h>
#include <rpc/auth.h>
#include <rpc/clnt.h>
#include <rpc/rpc_rdma.h>

#include <nfs/nfs.h>
#include <nfs/nfs_clnt.h>
#include <nfs/rnode.h>
#include <nfs/nfs_acl.h>
#include <nfs/lm.h>

#include <vm/hat.h>
#include <vm/as.h>
#include <vm/page.h>
#include <vm/pvn.h>
#include <vm/seg.h>
#include <vm/seg_map.h>
#include <vm/seg_kpm.h>
#include <vm/seg_vn.h>

#include <fs/fs_subr.h>

#include <sys/ddi.h>

static int      nfs3_rdwrlbn(vnode_t *, page_t *, u_offset_t, size_t, int,
                        cred_t *);
static int      nfs3write(vnode_t *, caddr_t, u_offset_t, int, cred_t *,
                        stable_how *);
static int      nfs3read(vnode_t *, caddr_t, offset_t, int, size_t *, cred_t *);
static int      nfs3setattr(vnode_t *, struct vattr *, int, cred_t *);
static int      nfs3_accessx(void *, int, cred_t *);
static int      nfs3lookup_dnlc(vnode_t *, char *, vnode_t **, cred_t *);
static int      nfs3lookup_otw(vnode_t *, char *, vnode_t **, cred_t *, int);
static int      nfs3create(vnode_t *, char *, struct vattr *, enum vcexcl,
                        int, vnode_t **, cred_t *, int);
static int      nfs3excl_create_settimes(vnode_t *, struct vattr *, cred_t *);
static int      nfs3mknod(vnode_t *, char *, struct vattr *, enum vcexcl,
                        int, vnode_t **, cred_t *);
static int      nfs3rename(vnode_t *, char *, vnode_t *, char *, cred_t *,
                        caller_context_t *);
static int      do_nfs3readdir(vnode_t *, rddir_cache *, cred_t *);
static void     nfs3readdir(vnode_t *, rddir_cache *, cred_t *);
static void     nfs3readdirplus(vnode_t *, rddir_cache *, cred_t *);
static int      nfs3_bio(struct buf *, stable_how *, cred_t *);
static int      nfs3_getapage(vnode_t *, u_offset_t, size_t, uint_t *,
                        page_t *[], size_t, struct seg *, caddr_t,
                        enum seg_rw, cred_t *);
static void     nfs3_readahead(vnode_t *, u_offset_t, caddr_t, struct seg *,
                        cred_t *);
static int      nfs3_sync_putapage(vnode_t *, page_t *, u_offset_t, size_t,
                        int, cred_t *);
static int      nfs3_sync_pageio(vnode_t *, page_t *, u_offset_t, size_t,
                        int, cred_t *);
static int      nfs3_commit(vnode_t *, offset3, count3, cred_t *);
static void     nfs3_set_mod(vnode_t *);
static void     nfs3_get_commit(vnode_t *);
static void     nfs3_get_commit_range(vnode_t *, u_offset_t, size_t);
```

```
127 static int      nfs3_putpage_commit(vnode_t *, offset_t, size_t, cred_t *);
128 static int      nfs3_commit_vp(vnode_t *, u_offset_t, size_t,  cred_t *);
129 static int      nfs3_sync_commit(vnode_t *, page_t *, offset3, count3,
130                         cred_t *);
131 static void     nfs3_async_commit(vnode_t *, page_t *, offset3, count3,
132                         cred_t *);
133 static void     nfs3_delmap_callback(struct as *, void *, uint_t);

135 /*
136  * Error flags used to pass information about certain special errors
137  * which need to be handled specially.
138  */
139 #define NFS_EOF                 -98
140 #define NFS_VERF_MISMATCH       -97

142 /* ALIGN64 aligns the given buffer and adjust buffer size to 64 bit */
143 #define ALIGN64(x, ptr, sz)                                             \
144         x = ((uintptr_t)(ptr)) & (sizeof (uint64_t) - 1);               \
145         if (x) {                                                        \
146                 x = sizeof (uint64_t) - (x);                            \
147                 sz -= (x);                                              \
148                 ptr += (x);                                             \
149         }

151 /*
152  * These are the vnode ops routines which implement the vnode interface to
153  * the networked file system.  These routines just take their parameters,
154  * make them look networkish by putting the right info into interface structs,
155  * and then calling the appropriate remote routine(s) to do the work.
156  *
157  * Note on directory name lookup cacheing:  If we detect a stale fhandle,
158  * we purge the directory cache relative to that vnode.  This way, the
159  * user won't get burned by the cache repeatedly.  See <nfs/rnode.h> for
160  * more details on rnode locking.
161  */

163 static int      nfs3_open(vnode_t **, int, cred_t *, caller_context_t *);
164 static int      nfs3_close(vnode_t *, int, int, offset_t, cred_t *,
165                         caller_context_t *);
166 static int      nfs3_read(vnode_t *, struct uio *, int, cred_t *,
167                         caller_context_t *);
168 static int      nfs3_write(vnode_t *, struct uio *, int, cred_t *,
169                         caller_context_t *);
170 static int      nfs3_ioctl(vnode_t *, int, intptr_t, int, cred_t *, int *,
171                         caller_context_t *);
172 static int      nfs3_getattr(vnode_t *, struct vattr *, int, cred_t *,
173                         caller_context_t *);
174 static int      nfs3_setattr(vnode_t *, struct vattr *, int, cred_t *,
175                         caller_context_t *);
176 static int      nfs3_access(vnode_t *, int, int, cred_t *, caller_context_t *);
177 static int      nfs3_readlink(vnode_t *, struct uio *, cred_t *,
178                         caller_context_t *);
179 static int      nfs3_fsync(vnode_t *, int, cred_t *, caller_context_t *);
180 static void     nfs3_inactive(vnode_t *, cred_t *, caller_context_t *);
181 static int      nfs3_lookup(vnode_t *, char *, vnode_t **,
182                         struct pathname *, int, vnode_t *, cred_t *,
183                         caller_context_t *, int *, pathname_t *);
184 static int      nfs3_create(vnode_t *, char *, struct vattr *, enum vcexcl,
185                         int, vnode_t **, cred_t *, int, caller_context_t *,
186                         vsecattr_t *);
187 static int      nfs3_remove(vnode_t *, char *, cred_t *, caller_context_t *,
188                         int);
189 static int      nfs3_link(vnode_t *, vnode_t *, char *, cred_t *,
190                         caller_context_t *, int);
191 static int      nfs3_rename(vnode_t *, char *, vnode_t *, char *, cred_t *,
192                         caller_context_t *, int);
```



```
193 static int      nfs3_mkdir(vnode_t *, char *, struct vattr *, vnode_t **,
194                         cred_t *, caller_context_t *, int, vsecattr_t *);
195 static int      nfs3_rmdir(vnode_t *, char *, vnode_t *, cred_t *,
196                         caller_context_t *, int);
197 static int      nfs3_symlink(vnode_t *, char *, struct vattr *, char *,
198                         cred_t *, caller_context_t *, int);
199 static int      nfs3_readdir(vnode_t *, struct uio *, cred_t *, int *,
200                         caller_context_t *, int);
201 static int      nfs3_fid(vnode_t *, fid_t *, caller_context_t *);
202 static int      nfs3_rwlock(vnode_t *, int, caller_context_t *);
203 static void     nfs3_rwunlock(vnode_t *, int, caller_context_t *);
204 static int      nfs3_seek(vnode_t *, offset_t, offset_t *, caller_context_t *);
205 static int      nfs3_getpage(vnode_t *, offset_t, size_t, uint_t *,
206                         page_t *[], size_t, struct seg *, caddr_t,
207                         enum seg_rw, cred_t *, caller_context_t *);
208 static int      nfs3_putpage(vnode_t *, offset_t, size_t, int, cred_t *,
209                         caller_context_t *);
210 static int      nfs3_map(vnode_t *, offset_t, struct as *, caddr_t *, size_t,
211                         uchar_t, uchar_t, uint_t, cred_t *, caller_context_t *);
212 static int      nfs3_addmap(vnode_t *, offset_t, struct as *, caddr_t, size_t,
213                         uchar_t, uchar_t, uint_t, cred_t *, caller_context_t *);
214 static int      nfs3_frlock(vnode_t *, int, struct flock64 *, int, offset_t,
215                         struct flk_callback *, cred_t *, caller_context_t *);
216 static int      nfs3_space(vnode_t *, int, struct flock64 *, int, offset_t,
217                         cred_t *, caller_context_t *);
218 static int      nfs3_realvp(vnode_t *, vnode_t **, caller_context_t *);
219 static int      nfs3_delmap(vnode_t *, offset_t, struct as *, caddr_t, size_t,
220                         uint_t, uint_t, uint_t, cred_t *, caller_context_t *);
221 static int      nfs3_pathconf(vnode_t *, int, ulong_t *, cred_t *,
222                         caller_context_t *);
223 static int      nfs3_pageio(vnode_t *, page_t *, u_offset_t, size_t, int,
224                         cred_t *, caller_context_t *);
225 static void     nfs3_dispose(vnode_t *, page_t *, int, int, cred_t *,
226                         caller_context_t *);
227 static int      nfs3_setsecattr(vnode_t *, vsecattr_t *, int, cred_t *,
228                         caller_context_t *);
229 static int      nfs3_getsecattr(vnode_t *, vsecattr_t *, int, cred_t *,
230                         caller_context_t *);
231 static int      nfs3_shrlock(vnode_t *, int, struct shrlock *, int, cred_t *,
232                         caller_context_t *);

234 struct vnodeops *nfs3_vnodeops;

236 const fs_operation_def_t nfs3_vnodeops_template[] = {
237         VOPNAME_OPEN,           { .vop_open = nfs3_open },
238         VOPNAME_CLOSE,          { .vop_close = nfs3_close },
239         VOPNAME_READ,           { .vop_read = nfs3_read },
240         VOPNAME_WRITE,          { .vop_write = nfs3_write },
241         VOPNAME_IOCTL,          { .vop_ioctl = nfs3_ioctl },
242         VOPNAME_GETATTR,        { .vop_getattr = nfs3_getattr },
243         VOPNAME_SETATTR,        { .vop_setattr = nfs3_setattr },
244         VOPNAME_ACCESS,         { .vop_access = nfs3_access },
245         VOPNAME_LOOKUP,         { .vop_lookup = nfs3_lookup },
246         VOPNAME_CREATE,         { .vop_create = nfs3_create },
247         VOPNAME_REMOVE,         { .vop_remove = nfs3_remove },
248         VOPNAME_LINK,           { .vop_link = nfs3_link },
249         VOPNAME_RENAME,         { .vop_rename = nfs3_rename },
250         VOPNAME_MKDIR,          { .vop_mkdir = nfs3_mkdir },
251         VOPNAME_RMDIR,          { .vop_rmdir = nfs3_rmdir },
252         VOPNAME_READDIR,        { .vop_readdir = nfs3_readdir },
253         VOPNAME_SYMLINK,        { .vop_symlink = nfs3_symlink },
254         VOPNAME_READLINK,       { .vop_readlink = nfs3_readlink },
255         VOPNAME_FSYNC,          { .vop_fsync = nfs3_fsync },
256         VOPNAME_INACTIVE,       { .vop_inactive = nfs3_inactive },
257         VOPNAME_FID,            { .vop_fid = nfs3_fid },
258         VOPNAME_RWLOCK,         { .vop_rwlock = nfs3_rwlock },
```

```
 259         VOPNAME_RWUNLOCK,       { .vop_rwunlock = nfs3_rwunlock },
 260         VOPNAME_SEEK,           { .vop_seek = nfs3_seek },
 261         VOPNAME_FRLOCK,         { .vop_frlock = nfs3_frlock },
 262         VOPNAME_SPACE,          { .vop_space = nfs3_space },
 263         VOPNAME_REALVP,         { .vop_realvp = nfs3_realvp },
 264         VOPNAME_GETPAGE,        { .vop_getpage = nfs3_getpage },
 265         VOPNAME_PUTPAGE,        { .vop_putpage = nfs3_putpage },
 266         VOPNAME_MAP,            { .vop_map = nfs3_map },
 267         VOPNAME_ADDMAP,         { .vop_addmap = nfs3_addmap },
 268         VOPNAME_DELMAP,         { .vop_delmap = nfs3_delmap },
 269         /* no separate nfs3_dump */
 270         VOPNAME_DUMP,           { .vop_dump = nfs_dump },
 271         VOPNAME_PATHCONF,       { .vop_pathconf = nfs3_pathconf },
 272         VOPNAME_PAGEIO,         { .vop_pageio = nfs3_pageio },
 273         VOPNAME_DISPOSE,        { .vop_dispose = nfs3_dispose },
 274         VOPNAME_SETSECATTR,     { .vop_setsecattr = nfs3_setsecattr },
 275         VOPNAME_GETSECATTR,     { .vop_getsecattr = nfs3_getsecattr },
 276         VOPNAME_SHRLOCK,        { .vop_shrlock = nfs3_shrlock },
 277         VOPNAME_VNEVENT,        { .vop_vnevent = fs_vnevent_support },
 278         NULL,                   NULL
 279 };
_____unchanged_portion_omitted_

2212 #ifdef DEBUG
2213 static int nfs3_create_misses = 0;
2214 #endif

2216 /* ARGSUSED */
2217 static int
2218 nfs3_create(vnode_t *dvp, char *nm, struct vattr *va, enum vcexcl exclusive,
2219         int mode, vnode_t **vpp, cred_t *cr, int lfaware, caller_context_t *ct,
2220         vsecattr_t *vsecp)
2221 {
2222         int error;
2223         vnode_t *vp;
2224         rnode_t *rp;
2225         struct vattr vattr;
2226         rnode_t *drp;
2227         vnode_t *tempvp;

2229         drp = VTOR(dvp);
2230         if (nfs_zone() != VTOMI(dvp)->mi_zone)
2231                 return (EPERM);
2232         if (nfs_rw_enter_sig(&drp->r_rwlock, RW_WRITER, INTR(dvp)))
2233                 return (EINTR);

2235 top:
2236         /*
2237          * We make a copy of the attributes because the caller does not
2238          * expect us to change what va points to.
2239          */
2240         vattr = *va;

2242         /*
2243          * If the pathname is "", just use dvp.  Don't need
2244          * to send it over the wire, look it up in the dnlc,
2245          * or perform any access checks.
2246          */
2247         if (*nm == '\0') {
2248                 error = 0;
2249                 VN_HOLD(dvp);
2250                 vp = dvp;
2251         /*
2252          * If the pathname is ".", just use dvp.  Don't need
2253          * to send it over the wire or look it up in the dnlc,
2254          * just need to check access.
```



```
2255          */
2256         } else if (strcmp(nm, ".") == 0) {
2257                 error = nfs3_access(dvp, VEXEC, 0, cr, ct);
2258                 if (error) {
2259                         nfs_rw_exit(&drp->r_rwlock);
2260                         return (error);
2261                 }
2262                 VN_HOLD(dvp);
2263                 vp = dvp;
2264         /*
2265          * We need to go over the wire, just to be sure whether the
2266          * file exists or not.  Using the DNLC can be dangerous in
2267          * this case when making a decision regarding existence.
2268          */
2269         } else {
2270                 error = nfs3lookup_otw(dvp, nm, &vp, cr, 0);
2271         }
2272         if (!error) {
2273                 if (exclusive == EXCL)
2274                         error = EEXIST;
2275                 else if (vp->v_type == VDIR && (mode & VWRITE))
2276                         error = EISDIR;
2277                 else {
2278                         /*
2279                          * If vnode is a device, create special vnode.
2280                          */
2281                         if (IS_DEVVP(vp)) {
2282                                 tempvp = vp;
2283                                 vp = specvp(vp, vp->v_rdev, vp->v_type, cr);
2284                                 VN_RELE(tempvp);
2285                         }
2286                         if (!(error = VOP_ACCESS(vp, mode, 0, cr, ct))) {
2287                                 if ((vattr.va_mask & AT_SIZE) &&
2288                                     vp->v_type == VREG) {
2289                                         rp = VTOR(vp);
2290                                         /*
2291                                          * Check here for large file handled
2292                                          * by LF-unaware process (as
2293                                          * ufs_create() does)
2294                                          */
2295                                         if (!(lfaware & FOFFMAX)) {
2296                                                 mutex_enter(&rp->r_statelock);
2297                                                 if (rp->r_size > MAXOFF32_T)
2298                                                         error = EOVERFLOW;
2299                                                 mutex_exit(&rp->r_statelock);
2300                                         }
2301                                         if (!error) {
2302                                                 vattr.va_mask = AT_SIZE;
2303                                                 error = nfs3setattr(vp,
2304                                                     &vattr, 0, cr);
2306                                                 /*
2307                                                  * Existing file was truncated;
2308                                                  * emit a create event.
2309                                                  */
2310                                                 vnevent_create(vp, ct);
2311                                         }
2312                                 }
2313                         }
2314                 }
2315                 nfs_rw_exit(&drp->r_rwlock);
2316                 if (error) {
2317                         VN_RELE(vp);
2318                 } else {
2309                         /*
2310                          * existing file got truncated, notify.
```

```
2311                         */
2312                        vnevent_create(vp, ct);
2319                        *vpp = vp;
2320                }

2322                return (error);
2323        }

2325        dnlc_remove(dvp, nm);

2327        /*
2328         * Decide what the group-id of the created file should be.
2329         * Set it in attribute list as advisory...
2330         */
2331        error = setdirgid(dvp, &vattr.va_gid, cr);
2332        if (error) {
2333                nfs_rw_exit(&drp->r_rwlock);
2334                return (error);
2335        }
2336        vattr.va_mask |= AT_GID;

2338        ASSERT(vattr.va_mask & AT_TYPE);
2339        if (vattr.va_type == VREG) {
2340                ASSERT(vattr.va_mask & AT_MODE);
2341                if (MANDMODE(vattr.va_mode)) {
2342                        nfs_rw_exit(&drp->r_rwlock);
2343                        return (EACCES);
2344                }
2345                error = nfs3create(dvp, nm, &vattr, exclusive, mode, vpp, cr,
2346                    lfaware);
2347                /*
2348                 * If this is not an exclusive create, then the CREATE
2349                 * request will be made with the GUARDED mode set.  This
2350                 * means that the server will return EEXIST if the file
2351                 * exists.  The file could exist because of a retransmitted
2352                 * request.  In this case, we recover by starting over and
2353                 * checking to see whether the file exists.  This second
2354                 * time through it should and a CREATE request will not be
2355                 * sent.
2356                 *
2357                 * This handles the problem of a dangling CREATE request
2358                 * which contains attributes which indicate that the file
2359                 * should be truncated.  This retransmitted request could
2360                 * possibly truncate valid data in the file if not caught
2361                 * by the duplicate request mechanism on the server or if
2362                 * not caught by other means.  The scenario is:
2363                 *
2364                 * Client transmits CREATE request with size = 0
2365                 * Client times out, retransmits request.
2366                 * Response to the first request arrives from the server
2367                 *  and the client proceeds on.
2368                 * Client writes data to the file.
2369                 * The server now processes retransmitted CREATE request
2370                 *  and truncates file.
2371                 *
2372                 * The use of the GUARDED CREATE request prevents this from
2373                 * happening because the retransmitted CREATE would fail
2374                 * with EEXIST and would not truncate the file.
2375                 */
2376                if (error == EEXIST && exclusive == NONEXCL) {
2377 #ifdef DEBUG
2378                        nfs3_create_misses++;
2379 #endif
2380                        goto top;
2381                }
2382                nfs_rw_exit(&drp->r_rwlock);
```



```
2383                return (error);
2384        }
2385        error = nfs3mknod(dvp, nm, &vattr, exclusive, mode, vpp, cr);
2386        nfs_rw_exit(&drp->r_rwlock);
2387        return (error);
2388 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
  429877 Fri Jan 18 13:59:18 2013
new/usr/src/uts/common/fs/nfs/nfs4_vnops.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * 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 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2012 Nexenta Systems, Inc. All rights reserved.
 */

/*
 *      Copyright 1983,1984,1985,1986,1987,1988,1989 AT&T.
 *      All Rights Reserved
 */

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

#include <sys/param.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <sys/cred.h>
#include <sys/time.h>
#include <sys/vnode.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/file.h>
#include <sys/filio.h>
#include <sys/uio.h>
#include <sys/buf.h>
#include <sys/mman.h>
#include <sys/pathname.h>
#include <sys/dirent.h>
#include <sys/debug.h>
#include <sys/vmsystm.h>
#include <sys/fcntl.h>
#include <sys/flock.h>
#include <sys/swap.h>
#include <sys/errno.h>
#include <sys/strsubr.h>
#include <sys/sysmacros.h>
```



```
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/pathconf.h>
#include <sys/utsname.h>
#include <sys/dnlc.h>
#include <sys/acl.h>
#include <sys/systeminfo.h>
#include <sys/policy.h>
#include <sys/sdt.h>
#include <sys/list.h>
#include <sys/stat.h>
#include <sys/zone.h>

#include <rpc/types.h>
#include <rpc/auth.h>
#include <rpc/clnt.h>

#include <nfs/nfs.h>
#include <nfs/nfs_clnt.h>
#include <nfs/nfs_acl.h>
#include <nfs/lm.h>
#include <nfs/nfs4.h>
#include <nfs/nfs4_kprot.h>
#include <nfs/rnode4.h>
#include <nfs/nfs4_clnt.h>

#include <vm/hat.h>
#include <vm/as.h>
#include <vm/page.h>
#include <vm/pvn.h>
#include <vm/seg.h>
#include <vm/seg_map.h>
#include <vm/seg_kpm.h>
#include <vm/seg_vn.h>

#include <fs/fs_subr.h>

#include <sys/ddi.h>
#include <sys/int_fmtio.h>
#include <sys/fs/autofs.h>

typedef struct {
        nfs4_ga_res_t   *di_garp;
        cred_t          *di_cred;
        hrtime_t        di_time_call;
} dirattr_info_t;
```

_____***unchanged_portion_omitted_***

```
/* ARGSUSED */
static int
nfs4_create(vnode_t *dvp, char *nm, struct vattr *va, enum vcexcl exclusive,
        int mode, vnode_t **vpp, cred_t *cr, int flags, caller_context_t *ct,
        vsecattr_t *vsecp)
{
        int error;
        vnode_t *vp = NULL;
        rnode4_t *rp;
        struct vattr vattr;
        rnode4_t *drp;
        vnode_t *tempvp;
        enum createmode4 createmode;
        bool_t must_trunc = FALSE;
        int     truncating = 0;

        if (nfs_zone() != VTOMI4(dvp)->mi_zone)
                return (EPERM);
```

```
6521         if (exclusive == EXCL && (dvp->v_flag & V_XATTRDIR)) {
6522                 return (EINVAL);
6523         }

6525         /* . and .. have special meaning in the protocol, reject them. */

6527         if (nm[0] == '.' && (nm[1] == '\0' || (nm[1] == '.' && nm[2] == '\0')))
6528                 return (EISDIR);

6530         drp = VTOR4(dvp);

6532         if (nfs_rw_enter_sig(&drp->r_rwlock, RW_WRITER, INTR4(dvp)))
6533                 return (EINTR);

6535 top:
6536         /*
6537          * We make a copy of the attributes because the caller does not
6538          * expect us to change what va points to.
6539          */
6540         vattr = *va;

6542         /*
6543          * If the pathname is "", then dvp is the root vnode of
6544          * a remote file mounted over a local directory.
6545          * All that needs to be done is access
6546          * checking and truncation.  Note that we avoid doing
6547          * open w/ create because the parent directory might
6548          * be in pseudo-fs and the open would fail.
6549          */
6550         if (*nm == '\0') {
6551                 error = 0;
6552                 VN_HOLD(dvp);
6553                 vp = dvp;
6554                 must_trunc = TRUE;
6555         } else {
6556                 /*
6557                  * We need to go over the wire, just to be sure whether the
6558                  * file exists or not.  Using the DNLC can be dangerous in
6559                  * this case when making a decision regarding existence.
6560                  */
6561                 error = nfs4lookup(dvp, nm, &vp, cr, 1);
6562         }

6564         if (exclusive)
6565                 createmode = EXCLUSIVE4;
6566         else
6567                 createmode = GUARDED4;

6569         /*
6570          * error would be set if the file does not exist on the
6571          * server, so lets go create it.
6572          */
6573         if (error) {
6574                 goto create_otw;
6575         }

6577         /*
6578          * File does exist on the server
6579          */
6580         if (exclusive == EXCL)
6581                 error = EEXIST;
6582         else if (vp->v_type == VDIR && (mode & VWRITE))
6583                 error = EISDIR;
6584         else {
6585                 /*
6586                  * If vnode is a device, create special vnode.
```



```
6587                  */
6588                 if (ISVDEV(vp->v_type)) {
6589                         tempvp = vp;
6590                         vp = specvp(vp, vp->v_rdev, vp->v_type, cr);
6591                         VN_RELE(tempvp);
6592                 }
6593                 if (!(error = VOP_ACCESS(vp, mode, 0, cr, ct))) {
6594                         if ((vattr.va_mask & AT_SIZE) &&
6595                             vp->v_type == VREG) {
6596                                 rp = VTOR4(vp);
6597                                 /*
6598                                  * Check here for large file handled
6599                                  * by LF-unaware process (as
6600                                  * ufs_create() does)
6601                                  */
6602                                 if (!(flags & FOFFMAX)) {
6603                                         mutex_enter(&rp->r_statelock);
6604                                         if (rp->r_size > MAXOFF32_T)
6605                                                 error = EOVERFLOW;
6606                                         mutex_exit(&rp->r_statelock);
6607                                 }

6609                                 /* if error is set then we need to return */
6610                                 if (error) {
6611                                         nfs_rw_exit(&drp->r_rwlock);
6612                                         VN_RELE(vp);
6613                                         return (error);
6614                                 }

6616                                 if (must_trunc) {
6617                                         vattr.va_mask = AT_SIZE;
6618                                         error = nfs4setattr(vp, &vattr, 0, cr,
6619                                             NULL);
6620                                 } else {
6621                                 /*
6622                                  * we know we have a regular file that already
6623                                  * exists and we may end up truncating the file
6624                                  * as a result of the open_otw, so flush out
6625                                  * any dirty pages for this file first.
6626                                  */
6627                                         if (nfs4_has_pages(vp) &&
6628                                             ((rp->r_flags & R4DIRTY) ||
6629                                             rp->r_count > 0 ||
6630                                             rp->r_mapcnt > 0)) {
6631                                                 error = nfs4_putpage(vp,
6632                                                     (offset_t)0, 0, 0, cr, ct);
6633                                                 if (error && (error == ENOSPC ||
6634                                                     error == EDQUOT)) {
6635                                                         mutex_enter(
6636                                                             &rp->r_statelock);
6637                                                         if (!rp->r_error)
6638                                                                 rp->r_error =
6639                                                                     error;
6640                                                         mutex_exit(
6641                                                             &rp->r_statelock);
6642                                                 }
6643                                         }
6644                                         vattr.va_mask = (AT_SIZE |
6645                                             AT_TYPE | AT_MODE);
6646                                         vattr.va_type = VREG;
6647                                         createmode = UNCHECKED4;
6648                                         truncating = 1;
6649                                         goto create_otw;
6650                                 }
6651                         }
6652                 }
```

```
6653         }
6654         nfs_rw_exit(&drp->r_rwlock);
6655         if (error) {
6656                 VN_RELE(vp);
6657         } else {
6658                 vnode_t *tvp;
6659                 rnode4_t *trp;
6656                 /*
6657                  * existing file got truncated, notify.
6658                  */
6660                 tvp = vp;
6661                 if (vp->v_type == VREG) {
6662                         trp = VTOR4(vp);
6663                         if (IS_SHADOW(vp, trp))
6664                                 tvp = RTOV4(trp);
6665                 }

6667                 if (must_trunc) {
6668                         /*
6669                          * existing file got truncated, notify.
6670                          */
6671                         vnevent_create(tvp, ct);
6672                 }

6674                 *vpp = vp;
6675         }
6676         return (error);

6678 create_otw:
6679         dnlc_remove(dvp, nm);

6681         ASSERT(vattr.va_mask & AT_TYPE);

6683         /*
6684          * If not a regular file let nfs4mknod() handle it.
6685          */
6686         if (vattr.va_type != VREG) {
6687                 error = nfs4mknod(dvp, nm, &vattr, exclusive, mode, vpp, cr);
6688                 nfs_rw_exit(&drp->r_rwlock);
6689                 return (error);
6690         }

6692         /*
6693          * It _is_ a regular file.
6694          */
6695         ASSERT(vattr.va_mask & AT_MODE);
6696         if (MANDMODE(vattr.va_mode)) {
6697                 nfs_rw_exit(&drp->r_rwlock);
6698                 return (EACCES);
6699         }

6701         /*
6702          * If this happens to be a mknod of a regular file, then flags will
6703          * have neither FREAD or FWRITE.  However, we must set at least one
6704          * for the call to nfs4open_otw.  If it's open(O_CREAT) driving
6705          * nfs4_create, then either FREAD, FWRITE, or FRDWR has already been
6706          * set (based on openmode specified by app).
6707          */
6708         if ((flags & (FREAD|FWRITE)) == 0)
6709                 flags |= (FREAD|FWRITE);

6711         error = nfs4open_otw(dvp, nm, &vattr, vpp, cr, 1, flags, createmode, 0);

6713         if (vp != NULL) {
6714                 /* if create was successful, throw away the file's pages */
6715                 if (!error && (vattr.va_mask & AT_SIZE))
```



```
6716                         nfs4_invalidate_pages(vp, (vattr.va_size & PAGEMASK),
6717                             cr);
6718                 /* release the lookup hold */
6719                 VN_RELE(vp);
6720                 vp = NULL;
6721         }

6723         /*
6724          * validate that we opened a regular file. This handles a misbehaving
6725          * server that returns an incorrect FH.
6726          */
6727         if ((error == 0) && *vpp && (*vpp)->v_type != VREG) {
6728                 error = EISDIR;
6729                 VN_RELE(*vpp);
6730         }

6732         /*
6733          * If this is not an exclusive create, then the CREATE
6734          * request will be made with the GUARDED mode set.  This
6735          * means that the server will return EEXIST if the file
6736          * exists.  The file could exist because of a retransmitted
6737          * request.  In this case, we recover by starting over and
6738          * checking to see whether the file exists.  This second
6739          * time through it should and a CREATE request will not be
6740          * sent.
6741          *
6742          * This handles the problem of a dangling CREATE request
6743          * which contains attributes which indicate that the file
6744          * should be truncated.  This retransmitted request could
6745          * possibly truncate valid data in the file if not caught
6746          * by the duplicate request mechanism on the server or if
6747          * not caught by other means.  The scenario is:
6748          *
6749          * Client transmits CREATE request with size = 0
6750          * Client times out, retransmits request.
6751          * Response to the first request arrives from the server
6752          *  and the client proceeds on.
6753          * Client writes data to the file.
6754          * The server now processes retransmitted CREATE request
6755          *  and truncates file.
6756          *
6757          * The use of the GUARDED CREATE request prevents this from
6758          * happening because the retransmitted CREATE would fail
6759          * with EEXIST and would not truncate the file.
6760          */
6761         if (error == EEXIST && exclusive == NONEXCL) {
6762 #ifdef DEBUG
6763                 nfs4_create_misses++;
6764 #endif
6765                 goto top;
6766         }
6767         nfs_rw_exit(&drp->r_rwlock);
6768         if (truncating && !error && *vpp) {
6769                 vnode_t *tvp;
6770                 rnode4_t *trp;
6771                 /*
6772                  * existing file got truncated, notify.
6773                  */
6774                 tvp = *vpp;
6775                 trp = VTOR4(tvp);
6776                 if (IS_SHADOW(tvp, trp))
6777                         tvp = RTOV4(trp);
6778                 vnevent_create(tvp, ct);
6779         }
6780         return (error);
6781 }
```

unchanged_portion_omitted_

```
**********************************************************
  130804 Fri Jan 18 13:59:20 2013
new/usr/src/uts/common/fs/nfs/nfs_vnops.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
/*
 * 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) 1990, 2010, Oracle and/or its affiliates. All rights reserved.
 *
 *      Copyright (c) 1983,1984,1985,1986,1987,1988,1989 AT&T.
 *      All rights reserved.
 */

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

#include <sys/param.h>
#include <sys/types.h>
#include <sys/systm.h>
#include <sys/cred.h>
#include <sys/time.h>
#include <sys/vnode.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/file.h>
#include <sys/filio.h>
#include <sys/uio.h>
#include <sys/buf.h>
#include <sys/mman.h>
#include <sys/pathname.h>
#include <sys/dirent.h>
#include <sys/debug.h>
#include <sys/vmsystm.h>
#include <sys/fcntl.h>
#include <sys/flock.h>
#include <sys/swap.h>
#include <sys/errno.h>
#include <sys/strsubr.h>
#include <sys/sysmacros.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/pathconf.h>
#include <sys/utsname.h>
#include <sys/dnlc.h>
#include <sys/acl.h>
```



```
#include <sys/atomic.h>
#include <sys/policy.h>
#include <sys/sdt.h>

#include <rpc/types.h>
#include <rpc/auth.h>
#include <rpc/clnt.h>

#include <nfs/nfs.h>
#include <nfs/nfs_clnt.h>
#include <nfs/rnode.h>
#include <nfs/nfs_acl.h>
#include <nfs/lm.h>

#include <vm/hat.h>
#include <vm/as.h>
#include <vm/page.h>
#include <vm/pvn.h>
#include <vm/seg.h>
#include <vm/seg_map.h>
#include <vm/seg_kpm.h>
#include <vm/seg_vn.h>

#include <fs/fs_subr.h>

#include <sys/ddi.h>

static int      nfs_rdwrlbn(vnode_t *, page_t *, u_offset_t, size_t, int,
                        cred_t *);
static int      nfswrite(vnode_t *, caddr_t, uint_t, int, cred_t *);
static int      nfsread(vnode_t *, caddr_t, uint_t, int, size_t *, cred_t *);
static int      nfssetattr(vnode_t *, struct vattr *, int, cred_t *);
static int      nfslookup_dnlc(vnode_t *, char *, vnode_t **, cred_t *);
static int      nfslookup_otw(vnode_t *, char *, vnode_t **, cred_t *, int);
static int      nfsrename(vnode_t *, char *, vnode_t *, char *, cred_t *,
                        caller_context_t *);
static int      nfsreaddir(vnode_t *, rddir_cache *, cred_t *);
static int      nfs_bio(struct buf *, cred_t *);
static int      nfs_getapage(vnode_t *, u_offset_t, size_t, uint_t *,
                        page_t *[], size_t, struct seg *, caddr_t,
                        enum seg_rw, cred_t *);
static void     nfs_readahead(vnode_t *, u_offset_t, caddr_t, struct seg *,
                        cred_t *);
static int      nfs_sync_putapage(vnode_t *, page_t *, u_offset_t, size_t,
                        int, cred_t *);
static int      nfs_sync_pageio(vnode_t *, page_t *, u_offset_t, size_t,
                        int, cred_t *);
static void     nfs_delmap_callback(struct as *, void *, uint_t);

/*
 * Error flags used to pass information about certain special errors
 * which need to be handled specially.
 */
#define NFS_EOF                 -98

/*
 * These are the vnode ops routines which implement the vnode interface to
 * the networked file system.  These routines just take their parameters,
 * make them look networkish by putting the right info into interface structs,
 * and then calling the appropriate remote routine(s) to do the work.
 *
 * Note on directory name lookup cacheing:  If we detect a stale fhandle,
 * we purge the directory cache relative to that vnode.  This way, the
 * user won't get burned by the cache repeatedly.  See <nfs/rnode.h> for
 * more details on rnode locking.
 */
```

```
128 static int      nfs_open(vnode_t **, int, cred_t *, caller_context_t *);
129 static int      nfs_close(vnode_t *, int, int, offset_t, cred_t *,
130                         caller_context_t *);
131 static int      nfs_read(vnode_t *, struct uio *, int, cred_t *,
132                         caller_context_t *);
133 static int      nfs_write(vnode_t *, struct uio *, int, cred_t *,
134                         caller_context_t *);
135 static int      nfs_ioctl(vnode_t *, int, intptr_t, int, cred_t *, int *,
136                         caller_context_t *);
137 static int      nfs_getattr(vnode_t *, struct vattr *, int, cred_t *,
138                         caller_context_t *);
139 static int      nfs_setattr(vnode_t *, struct vattr *, int, cred_t *,
140                         caller_context_t *);
141 static int      nfs_access(vnode_t *, int, int, cred_t *, caller_context_t *);
142 static int      nfs_accessx(void *, int, cred_t *);
143 static int      nfs_readlink(vnode_t *, struct uio *, cred_t *,
144                         caller_context_t *);
145 static int      nfs_fsync(vnode_t *, int, cred_t *, caller_context_t *);
146 static void     nfs_inactive(vnode_t *, cred_t *, caller_context_t *);
147 static int      nfs_lookup(vnode_t *, char *, vnode_t **, struct pathname *,
148                         int, vnode_t *, cred_t *, caller_context_t *,
149                         int *, pathname_t *);
150 static int      nfs_create(vnode_t *, char *, struct vattr *, enum vcexcl,
151                         int, vnode_t **, cred_t *, int, caller_context_t *,
152                         vsecattr_t *);
153 static int      nfs_remove(vnode_t *, char *, cred_t *, caller_context_t *,
154                         int);
155 static int      nfs_link(vnode_t *, vnode_t *, char *, cred_t *,
156                         caller_context_t *, int);
157 static int      nfs_rename(vnode_t *, char *, vnode_t *, char *, cred_t *,
158                         caller_context_t *, int);
159 static int      nfs_mkdir(vnode_t *, char *, struct vattr *, vnode_t **,
160                         cred_t *, caller_context_t *, int, vsecattr_t *);
161 static int      nfs_rmdir(vnode_t *, char *, vnode_t *, cred_t *,
162                         caller_context_t *, int);
163 static int      nfs_symlink(vnode_t *, char *, struct vattr *, char *,
164                         cred_t *, caller_context_t *, int);
165 static int      nfs_readdir(vnode_t *, struct uio *, cred_t *, int *,
166                         caller_context_t *, int);
167 static int      nfs_fid(vnode_t *, fid_t *, caller_context_t *);
168 static int      nfs_rwlock(vnode_t *, int, caller_context_t *);
169 static void     nfs_rwunlock(vnode_t *, int, caller_context_t *);
170 static int      nfs_seek(vnode_t *, offset_t, offset_t *, caller_context_t *);
171 static int      nfs_getpage(vnode_t *, offset_t, size_t, uint_t *,
172                         page_t *[], size_t, struct seg *, caddr_t,
173                         enum seg_rw, cred_t *, caller_context_t *);
174 static int      nfs_putpage(vnode_t *, offset_t, size_t, int, cred_t *,
175                         caller_context_t *);
176 static int      nfs_map(vnode_t *, offset_t, struct as *, caddr_t *, size_t,
177                         uchar_t, uchar_t, uint_t, cred_t *, caller_context_t *);
178 static int      nfs_addmap(vnode_t *, offset_t, struct as *, caddr_t, size_t,
179                         uchar_t, uchar_t, uint_t, cred_t *, caller_context_t *);
180 static int      nfs_frlock(vnode_t *, int, struct flock64 *, int, offset_t,
181                         struct flk_callback *, cred_t *, caller_context_t *);
182 static int      nfs_space(vnode_t *, int, struct flock64 *, int, offset_t,
183                         cred_t *, caller_context_t *);
184 static int      nfs_realvp(vnode_t *, vnode_t **, caller_context_t *);
185 static int      nfs_delmap(vnode_t *, offset_t, struct as *, caddr_t, size_t,
186                         uint_t, uint_t, uint_t, cred_t *, caller_context_t *);
187 static int      nfs_pathconf(vnode_t *, int, ulong_t *, cred_t *,
188                         caller_context_t *);
189 static int      nfs_pageio(vnode_t *, page_t *, u_offset_t, size_t, int,
190                         cred_t *, caller_context_t *);
191 static int      nfs_setsecattr(vnode_t *, vsecattr_t *, int, cred_t *,
192                         caller_context_t *);
```



```
193 static int      nfs_getsecattr(vnode_t *, vsecattr_t *, int, cred_t *,
194                         caller_context_t *);
195 static int      nfs_shrlock(vnode_t *, int, struct shrlock *, int, cred_t *,
196                         caller_context_t *);

198 struct vnodeops *nfs_vnodeops;

200 const fs_operation_def_t nfs_vnodeops_template[] = {
201         VOPNAME_OPEN,           { .vop_open = nfs_open },
202         VOPNAME_CLOSE,          { .vop_close = nfs_close },
203         VOPNAME_READ,           { .vop_read = nfs_read },
204         VOPNAME_WRITE,          { .vop_write = nfs_write },
205         VOPNAME_IOCTL,          { .vop_ioctl = nfs_ioctl },
206         VOPNAME_GETATTR,        { .vop_getattr = nfs_getattr },
207         VOPNAME_SETATTR,        { .vop_setattr = nfs_setattr },
208         VOPNAME_ACCESS,         { .vop_access = nfs_access },
209         VOPNAME_LOOKUP,         { .vop_lookup = nfs_lookup },
210         VOPNAME_CREATE,         { .vop_create = nfs_create },
211         VOPNAME_REMOVE,         { .vop_remove = nfs_remove },
212         VOPNAME_LINK,           { .vop_link = nfs_link },
213         VOPNAME_RENAME,         { .vop_rename = nfs_rename },
214         VOPNAME_MKDIR,          { .vop_mkdir = nfs_mkdir },
215         VOPNAME_RMDIR,          { .vop_rmdir = nfs_rmdir },
216         VOPNAME_READDIR,        { .vop_readdir = nfs_readdir },
217         VOPNAME_SYMLINK,        { .vop_symlink = nfs_symlink },
218         VOPNAME_READLINK,       { .vop_readlink = nfs_readlink },
219         VOPNAME_FSYNC,          { .vop_fsync = nfs_fsync },
220         VOPNAME_INACTIVE,       { .vop_inactive = nfs_inactive },
221         VOPNAME_FID,            { .vop_fid = nfs_fid },
222         VOPNAME_RWLOCK,         { .vop_rwlock = nfs_rwlock },
223         VOPNAME_RWUNLOCK,       { .vop_rwunlock = nfs_rwunlock },
224         VOPNAME_SEEK,           { .vop_seek = nfs_seek },
225         VOPNAME_FRLOCK,         { .vop_frlock = nfs_frlock },
226         VOPNAME_SPACE,          { .vop_space = nfs_space },
227         VOPNAME_REALVP,         { .vop_realvp = nfs_realvp },
228         VOPNAME_GETPAGE,        { .vop_getpage = nfs_getpage },
229         VOPNAME_PUTPAGE,        { .vop_putpage = nfs_putpage },
230         VOPNAME_MAP,            { .vop_map = nfs_map },
231         VOPNAME_ADDMAP,         { .vop_addmap = nfs_addmap },
232         VOPNAME_DELMAP,         { .vop_delmap = nfs_delmap },
233         VOPNAME_DUMP,           { .vop_dump = nfs_dump },
234         VOPNAME_PATHCONF,       { .vop_pathconf = nfs_pathconf },
235         VOPNAME_PAGEIO,         { .vop_pageio = nfs_pageio },
236         VOPNAME_SETSECATTR,     { .vop_setsecattr = nfs_setsecattr },
237         VOPNAME_GETSECATTR,     { .vop_getsecattr = nfs_getsecattr },
238         VOPNAME_SHRLOCK,        { .vop_shrlock = nfs_shrlock },
239         VOPNAME_VNEVENT,        { .vop_vnevent = fs_vnevent_support },
240         NULL,                   NULL
241 };
_____unchanged_portion_omitted_

1958 /* ARGSUSED */
1959 static int
1960 nfs_create(vnode_t *dvp, char *nm, struct vattr *va, enum vcexcl exclusive,
1961         int mode, vnode_t **vpp, cred_t *cr, int lfaware, caller_context_t *ct,
1962         vsecattr_t *vsecp)
1963 {
1964         int error;
1965         struct nfscreatargs args;
1966         struct nfsdiropres dr;
1967         int douprintf;
1968         vnode_t *vp;
1969         rnode_t *rp;
1970         struct vattr vattr;
1971         rnode_t *drp;
1972         vnode_t *tempvp;
```

```
1973         hrtime_t t;

1975         drp = VTOR(dvp);

1977         if (nfs_zone() != VTOMI(dvp)->mi_zone)
1978                 return (EPERM);
1979         if (nfs_rw_enter_sig(&drp->r_rwlock, RW_WRITER, INTR(dvp)))
1980                 return (EINTR);

1982         /*
1983          * We make a copy of the attributes because the caller does not
1984          * expect us to change what va points to.
1985          */
1986         vattr = *va;

1988         /*
1989          * If the pathname is "", just use dvp.  Don't need
1990          * to send it over the wire, look it up in the dnlc,
1991          * or perform any access checks.
1992          */
1993         if (*nm == '\0') {
1994                 error = 0;
1995                 VN_HOLD(dvp);
1996                 vp = dvp;
1997         /*
1998          * If the pathname is ".", just use dvp.  Don't need
1999          * to send it over the wire or look it up in the dnlc,
2000          * just need to check access.
2001          */
2002         } else if (strcmp(nm, ".") == 0) {
2003                 error = nfs_access(dvp, VEXEC, 0, cr, ct);
2004                 if (error) {
2005                         nfs_rw_exit(&drp->r_rwlock);
2006                         return (error);
2007                 }
2008                 VN_HOLD(dvp);
2009                 vp = dvp;
2010         /*
2011          * We need to go over the wire, just to be sure whether the
2012          * file exists or not.  Using the DNLC can be dangerous in
2013          * this case when making a decision regarding existence.
2014          */
2015         } else {
2016                 error = nfslookup_otw(dvp, nm, &vp, cr, 0);
2017         }
2018         if (!error) {
2019                 if (exclusive == EXCL)
2020                         error = EEXIST;
2021                 else if (vp->v_type == VDIR && (mode & VWRITE))
2022                         error = EISDIR;
2023                 else {
2024                         /*
2025                          * If vnode is a device, create special vnode.
2026                          */
2027                         if (IS_DEVVP(vp)) {
2028                                 tempvp = vp;
2029                                 vp = specvp(vp, vp->v_rdev, vp->v_type, cr);
2030                                 VN_RELE(tempvp);
2031                         }
2032                         if (!(error = VOP_ACCESS(vp, mode, 0, cr, ct))) {
2033                                 if ((vattr.va_mask & AT_SIZE) &&
2034                                     vp->v_type == VREG) {
2035                                         vattr.va_mask = AT_SIZE;
2036                                         error = nfssetattr(vp, &vattr, 0, cr);

2038                                         if (!error) {
```



```
2039                                                 /*
2040                                                  * Existing file was truncated;
2041                                                  * emit a create event.
2042                                                  */
2043                                                 vnevent_create(vp, ct);
2044                                         }
2045                                 }
2046                         }
2047                 }
2048                 nfs_rw_exit(&drp->r_rwlock);
2049                 if (error) {
2050                         VN_RELE(vp);
2051                 } else {
2040                         /*
2041                          * existing file got truncated, notify.
2042                          */
2043                         vnevent_create(vp, ct);
2052                         *vpp = vp;
2053                 }
2054                 return (error);
2055         }

2057         ASSERT(vattr.va_mask & AT_TYPE);
2058         if (vattr.va_type == VREG) {
2059                 ASSERT(vattr.va_mask & AT_MODE);
2060                 if (MANDMODE(vattr.va_mode)) {
2061                         nfs_rw_exit(&drp->r_rwlock);
2062                         return (EACCES);
2063                 }
2064         }

2066         dnlc_remove(dvp, nm);

2068         setdiropargs(&args.ca_da, nm, dvp);

2070         /*
2071          * Decide what the group-id of the created file should be.
2072          * Set it in attribute list as advisory...then do a setattr
2073          * if the server didn't get it right the first time.
2074          */
2075         error = setdirgid(dvp, &vattr.va_gid, cr);
2076         if (error) {
2077                 nfs_rw_exit(&drp->r_rwlock);
2078                 return (error);
2079         }
2080         vattr.va_mask |= AT_GID;

2082         /*
2083          * This is a completely gross hack to make mknod
2084          * work over the wire until we can wack the protocol
2085          */
2086 #define IFCHR           0020000         /* character special */
2087 #define IFBLK           0060000         /* block special */
2088 #define IFSOCK          0140000         /* socket */

2090         /*
2091          * dev_t is uint_t in 5.x and short in 4.x. Both 4.x
2092          * supports 8 bit majors. 5.x supports 14 bit majors. 5.x supports 18
2093          * bits in the minor number where 4.x supports 8 bits.  If the 5.x
2094          * minor/major numbers <= 8 bits long, compress the device
2095          * number before sending it. Otherwise, the 4.x server will not
2096          * create the device with the correct device number and nothing can be
2097          * done about this.
2098          */
2099         if (vattr.va_type == VCHR || vattr.va_type == VBLK) {
2100                 dev_t d = vattr.va_rdev;
```

```
2101                dev32_t dev32;

2103                if (vattr.va_type == VCHR)
2104                        vattr.va_mode |= IFCHR;
2105                else
2106                        vattr.va_mode |= IFBLK;

2108                (void) cmpldev(&dev32, d);
2109                if (dev32 & ~((SO4_MAXMAJ << L_BITSMINOR32) | SO4_MAXMIN))
2110                        vattr.va_size = (u_offset_t)dev32;
2111                else
2112                        vattr.va_size = (u_offset_t)nfsv2_cmpdev(d);

2114                vattr.va_mask |= AT_MODE|AT_SIZE;
2115        } else if (vattr.va_type == VFIFO) {
2116                vattr.va_mode |= IFCHR;         /* xtra kludge for namedpipe */
2117                vattr.va_size = (u_offset_t)NFS_FIFO_DEV;       /* blech */
2118                vattr.va_mask |= AT_MODE|AT_SIZE;
2119        } else if (vattr.va_type == VSOCK) {
2120                vattr.va_mode |= IFSOCK;
2121                /*
2122                 * To avoid triggering bugs in the servers set AT_SIZE
2123                 * (all other RFS_CREATE calls set this).
2124                 */
2125                vattr.va_size = 0;
2126                vattr.va_mask |= AT_MODE|AT_SIZE;
2127        }

2129        args.ca_sa = &args.ca_sa_buf;
2130        error = vattr_to_sattr(&vattr, args.ca_sa);
2131        if (error) {
2132                /* req time field(s) overflow - return immediately */
2133                nfs_rw_exit(&drp->r_rwlock);
2134                return (error);
2135        }

2137        douprintf = 1;

2139        t = gethrtime();

2141        error = rfs2call(VTOMI(dvp), RFS_CREATE,
2142            xdr_creatargs, (caddr_t)&args,
2143            xdr_diropres, (caddr_t)&dr, cr,
2144            &douprintf, &dr.dr_status, 0, NULL);

2146        PURGE_ATTRCACHE(dvp);   /* mod time changed */

2148        if (!error) {
2149                error = geterrno(dr.dr_status);
2150                if (!error) {
2151                        if (HAVE_RDDIR_CACHE(drp))
2152                                nfs_purge_rddir_cache(dvp);
2153                        vp = makenfsnode(&dr.dr_fhandle, &dr.dr_attr,
2154                            dvp->v_vfsp, t, cr, NULL, NULL);
2155                        /*
2156                         * If NFS_ACL is supported on the server, then the
2157                         * attributes returned by server may have minimal
2158                         * permissions sometimes denying access to users having
2159                         * proper access.  To get the proper attributes, mark
2160                         * the attributes as expired so that they will be
2161                         * regotten via the NFS_ACL GETATTR2 procedure.
2162                         */
2163                        if (VTOMI(vp)->mi_flags & MI_ACL) {
2164                                PURGE_ATTRCACHE(vp);
2165                        }
2166                        dnlc_update(dvp, nm, vp);
```



```
2167                        rp = VTOR(vp);
2168                        if (vattr.va_size == 0) {
2169                                mutex_enter(&rp->r_statelock);
2170                                rp->r_size = 0;
2171                                mutex_exit(&rp->r_statelock);
2172                                if (vn_has_cached_data(vp)) {
2173                                        ASSERT(vp->v_type != VCHR);
2174                                        nfs_invalidate_pages(vp,
2175                                            (u_offset_t)0, cr);
2176                                }
2177                        }

2179                        /*
2180                         * Make sure the gid was set correctly.
2181                         * If not, try to set it (but don't lose
2182                         * any sleep over it).
2183                         */
2184                        if (vattr.va_gid != rp->r_attr.va_gid) {
2185                                vattr.va_mask = AT_GID;
2186                                (void) nfssetattr(vp, &vattr, 0, cr);
2187                        }

2189                        /*
2190                         * If vnode is a device create special vnode
2191                         */
2192                        if (IS_DEVVP(vp)) {
2193                                *vpp = specvp(vp, vp->v_rdev, vp->v_type, cr);
2194                                VN_RELE(vp);
2195                        } else
2196                                *vpp = vp;
2197                } else {
2198                        PURGE_STALE_FH(error, dvp, cr);
2199                }
2200        }

2202        nfs_rw_exit(&drp->r_rwlock);

2204        return (error);
2205 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   63316 Fri Jan 18 13:59:20 2013
new/usr/src/uts/common/fs/portfs/port_fop.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
/*
 * 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 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

/*
 * File Events Notification
 * ------------------------
 *
 * The File Events Notification facility provides file and directory change
 * notification. It is implemented as an event source(PORT_SOURCE_FILE)
 * under the Event Ports framework. Therefore the API is an extension to
 * the Event Ports API.
 *
 * It uses the FEM (File Events Monitoring) framework to intercept
 * operations on the files & directories and generate appropriate events.
 *
 * It provides event notification in accordance with what an application
 * can find out by stat'ing the file and comparing time stamps. The various
 * system calls that update the file's access, modification, and change
 * time stamps are documented in the man page section 2.
 *
 * It is non intrusive. That is, having an active file event watch on a file
 * or directory will not prevent it from being removed or renamed or block an
 * unmount operation of the file system where the watched file or directory
 * resides.
 *
 *
 * Interface:
 * ----------
 *
 *   The object for this event source is of type 'struct file_obj *'
 *
 *   The file that needs to be monitored is specified in 'fo_name'.
 *   The time stamps collected by a stat(2) call are passed in fo_atime,
 *   fo_mtime, fo_ctime. At the time a file events watch is registered, the
```



```
 *   time stamps passed in are compared with the current time stamps of the
 *   file. If it has changed, relevant events are sent immediately. If the time
 *   stamps are all '0', they will not be compared.
 *
 *
 * The events are delivered to an event port. A port is created using
 * port_create().
 *
 * To register a file events watch on a file or directory.
 *
 *   port_associate(int port, PORT_SOURCE_FILE, (uintptr_t)&fobj, events, user)
 *
 *   'user' is the user pointer to be returned with the event.
 *
 * To de-register a file events watch,
 *
 *   port_dissociate(int port, PORT_SOURCE_FILE, (uintptr_t)&fobj)
 *
 * The events are collected using the port_get()/port_getn() interface. The
 * event source will be PORT_SOURCE_FILE.
 *
 * After an event is delivered, the file events watch gets de-activated. To
 * receive the next event, the process will have to re-register the watch and
 * activate it by calling port_associate() again. This behavior is intentional
 * and supports proper multi threaded programming when using file events
 * notification API.
 *
 *
 * Implementation overview:
 * ------------------------
 *
 * Each file events watch is represented by 'portfop_t' in the kernel. A
 * cache(in portfop_cache_t) of these portfop_t's are maintained per event
 * port by this source. The object here is the pointer to the file_obj
 * structure. The portfop_t's are hashed in using the object pointer. Therefore
 * it is possible to have multiple file events watches on a file by the same
 * process by using different object structure(file_obj_t) and hence can
 * receive multiple event notification for a file. These watches can be for
 * different event types.
 *
 * The cached entries of these file objects are retained, even after delivering
 * an event, marking them inactive for performance reasons. The assumption
 * is that the process would come back and re-register the file to receive
 * further events. When there are more then 'port_fop_maxpfps' watches per file
 * it will attempt to free the oldest inactive watches.
 *
 * In case the event that is being delivered is an exception event, the cached
 * entries get removed. An exception event on a file or directory means its
 * identity got changed(rename to/from, delete, mounted over, file system
 * unmount).
 *
 * If the event port gets closed, all the associated file event watches will be
 * removed and discarded.
 *
 *
 * Data structures:
 * ----------------
 *
 * The list of file event watches per file are managed by the data structure
 * portfop_vp_t. The first time a file events watch is registered for a file,
 * a portfop_vp_t is installed on the vnode_t's member v_fopdata. This gets
 * removed and freed only when the vnode becomes inactive. The FEM hooks are
 * also installed when the first watch is registered on a file. The FEM hooks
 * get un-installed when all the watches are removed.
 *
 * Each file events watch is represented by the structure portfop_t. They
```

```
 127  * get added to a list of portfop_t's on the vnode(portfop_vp_t). After
 128  * delivering an event, the portfop_t is marked inactive but retained. It is
 129  * moved to the end of the list. All the active portfop_t's are maintained at
 130  * the beginning. In case of exception events, the portfop_t will be removed
 131  * and discarded.
 132  *
 133  * To intercept unmount operations, FSEM hooks are added to the file system
 134  * under which files are being watched. A hash table('portfop_vfs_hash_t') of
 135  * active file systems is maintained. Each file system that has active watches
 136  * is represented by 'portfop_vfs_t' and is added to the hash table.
 137  * The vnode's 'portfop_vp_t' structure is added to the list of files(vnodes)
 138  * being watched on the portfop_vfs_t structure.
 139  *
 140  *
 141  * File system support:
 142  * --------------------
 143  *
 144  * The file system implementation has to provide vnode event notifications
 145  * (vnevents) in order to support watching any files on that file system.
 146  * The vnode events(vnevents) are notifications provided by the file system
 147  * for name based file operations like rename, remove etc, which do not go
 148  * thru the VOP_** interfaces. If the file system does not implement vnode
 149  * notifications, watching for file events on such file systems is not
 150  * supported. The vnode event notifications support is determined by the call
 151  * vnevent_support(vp) (VOP_VNEVENT(vp, VE_SUPPORT)), which the file system
 152  * has to implement.
 153  *
 154  *
 155  * Locking order:
 156  * --------------
 157  *
 158  * A file(vnode) can have file event watches registered by different processes.
 159  * There is one portfop_t per watch registered. These are on the vnode's list
 160  * protected by the mutex 'pvp_mutex' in 'portfop_vp_t'. The portfop_t's are
 161  * also on the per port cache. The cache is protected by the pfc_lock of
 162  * portfop_cache_t. The lock order here is 'pfc_lock' -> 'pvp_mutex'.
 163  *
 164  */

 166 #include <sys/types.h>
 167 #include <sys/systm.h>
 168 #include <sys/stat.h>
 169 #include <sys/errno.h>
 170 #include <sys/kmem.h>
 171 #include <sys/sysmacros.h>
 172 #include <sys/debug.h>
 173 #include <sys/vnode.h>
 174 #include <sys/poll_impl.h>
 175 #include <sys/port_impl.h>
 176 #include <sys/fem.h>
 177 #include <sys/vfs_opreg.h>
 178 #include <sys/atomic.h>
 179 #include <sys/mount.h>
 180 #include <sys/mntent.h>

 182 /*
 183  * For special case support of mnttab (/etc/mnttab).
 184  */
 185 extern struct vnode *vfs_mntdummyvp;
 186 extern int mntfstype;

 188 #define PORTFOP_PVFSH(vfsp)     (&portvfs_hash[PORTFOP_PVFSHASH(vfsp)])
 189 portfop_vfs_hash_t       portvfs_hash[PORTFOP_PVFSHASH_SZ];

 191 #define PORTFOP_NVP     20
 192 /*
```



```
 193  * Inactive file event watches(portfop_t) are retained on the vnode's list
 194  * for performance reason. If the applications re-registers the file, the
 195  * inactive entry is made active and moved up the list.
 196  *
 197  * If there are greater then the following number of watches on a vnode,
 198  * it will attempt to discard an oldest inactive watch(pfp) at the time
 199  * a new watch is being registered and when events get delivered. We
 200  * do this to avoid accumulating inactive watches on a file.
 201  */
 202 int     port_fop_maxpfps = 20;

 204 /* local functions */
 205 static int      port_fop_callback(void *, int *, pid_t, int, void *);

 207 static void     port_pcache_insert(portfop_cache_t *, portfop_t *);
 208 static void     port_pcache_delete(portfop_cache_t *, portfop_t *);
 209 static void     port_close_fop(void *arg, int port, pid_t pid, int lastclose);

 211 /*
 212  * port fop functions that will be the fem hooks.
 213  */
 214 static int port_fop_open(femarg_t *vf, int mode, cred_t *cr,
 215     caller_context_t *);
 216 static int port_fop_read(femarg_t *vf, uio_t *uiop, int ioflag, cred_t *cr,
 217     struct caller_context *ct);
 218 static int port_fop_write(femarg_t *vf, uio_t *uiop, int ioflag, cred_t *cr,
 219     caller_context_t *ct);
 220 static int port_fop_map(femarg_t *vf, offset_t off, struct as *as,
 221     caddr_t *addrp, size_t len, uchar_t prot, uchar_t maxport,
 222     uint_t flags, cred_t *cr, caller_context_t *ct);
 223 static int port_fop_setattr(femarg_t *vf, vattr_t *vap, int flags, cred_t *cr,
 224     caller_context_t *ct);
 225 static int port_fop_create(femarg_t *vf, char *name, vattr_t *vap,
 226     vcexcl_t excl, int mode, vnode_t **vpp, cred_t *cr, int flag,
 227     caller_context_t *ct, vsecattr_t *vsecp);
 228 static int port_fop_remove(femarg_t *vf, char *nm, cred_t *cr,
 229     caller_context_t *ct, int flags);
 230 static int port_fop_link(femarg_t *vf, vnode_t *svp, char *tnm, cred_t *cr,
 231     caller_context_t *ct, int flags);
 232 static int port_fop_rename(femarg_t *vf, char *snm, vnode_t *tdvp, char *tnm,
 233     cred_t *cr, caller_context_t *ct, int flags);
 234 static int port_fop_mkdir(femarg_t *vf, char *dirname, vattr_t *vap,
 235     vnode_t **vpp, cred_t *cr, caller_context_t *ct, int flags,
 236     vsecattr_t *vsecp);
 237 static int port_fop_rmdir(femarg_t *vf, char *nm, vnode_t *cdir, cred_t *cr,
 238     caller_context_t *ct, int flags);
 239 static int port_fop_readdir(femarg_t *vf, uio_t *uiop, cred_t *cr, int *eofp,
 240     caller_context_t *ct, int flags);
 241 static int port_fop_symlink(femarg_t *vf, char *linkname, vattr_t *vap,
 242     char *target, cred_t *cr, caller_context_t *ct, int flags);
 243 static int port_fop_setsecattr(femarg_t *vf, vsecattr_t *vsap, int flag,
 244     cred_t *cr, caller_context_t *ct);

 246 static int port_fop_vnevent(femarg_t *vf, vnevent_t vnevent, vnode_t *dvp,
 247     char *cname, caller_context_t *ct);

 249 static int port_fop_unmount(fsemarg_t *vf, int flag, cred_t *cr);


 252 /*
 253  * Fem hooks.
 254  */
 255 const fs_operation_def_t        port_vnodesrc_template[] = {
 256         VOPNAME_OPEN,           { .femop_open = port_fop_open },
 257         VOPNAME_READ,           { .femop_read = port_fop_read },
 258         VOPNAME_WRITE,          { .femop_write = port_fop_write },
```

```
 259         VOPNAME_MAP,            { .femop_map = port_fop_map },
 260         VOPNAME_SETATTR,        { .femop_setattr = port_fop_setattr },
 261         VOPNAME_CREATE,         { .femop_create = port_fop_create },
 262         VOPNAME_REMOVE,         { .femop_remove = port_fop_remove },
 263         VOPNAME_LINK,           { .femop_link = port_fop_link },
 264         VOPNAME_RENAME,         { .femop_rename = port_fop_rename },
 265         VOPNAME_MKDIR,          { .femop_mkdir = port_fop_mkdir },
 266         VOPNAME_RMDIR,          { .femop_rmdir = port_fop_rmdir },
 267         VOPNAME_READDIR,        { .femop_readdir = port_fop_readdir },
 268         VOPNAME_SYMLINK,        { .femop_symlink = port_fop_symlink },
 269         VOPNAME_SETSECATTR,     { .femop_setsecattr = port_fop_setsecattr },
 270         VOPNAME_VNEVENT,        { .femop_vnevent = port_fop_vnevent },
 271         NULL,   NULL
 272 };
______unchanged_portion_omitted_

1946 /*
1947  * Given the file operation, map it to the event types and send.
1948  */
1949 void
1950 port_fop(vnode_t *vp, int op, int retval)
1951 {
1952         int event = 0;
1953         /*
1954          * deliver events only if the operation was successful.
1955          */
1956         if (retval)
1957                 return;

1959         /*
1960          * These events occurring on the watched file.
1961          */
1962         if (op & FOP_MODIFIED_MASK) {
1963                 event  = FILE_MODIFIED;
1964         }
1965         if (op & FOP_ACCESS_MASK) {
1966                 event  |= FILE_ACCESS;
1967         }
1968         if (op & FOP_ATTRIB_MASK) {
1969                 event  |= FILE_ATTRIB;
1970         }
1971         if (op & FOP_TRUNC_MASK) {
1972                 event  |= FILE_TRUNC;
1973         }

1974         if (event) {
1975                 port_fop_sendevent(vp,  event, NULL, NULL);
1976         }
1977 }
______unchanged_portion_omitted_


2143 /*
2144  * AT_SIZE - is for the open(O_TRUNC) case.
2145  */
2146 int
2147 port_fop_setattr(femarg_t *vf, vattr_t *vap, int flags, cred_t *cr,
2148     caller_context_t *ct)
2149 {
2150         int             retval;
2151         vnode_t         *vp = (vnode_t *)vf->fa_fnode->fn_available;
2152         int             events = 0;

2154         retval = vnext_setattr(vf, vap, flags, cr, ct);
2155         if (vap->va_mask & AT_SIZE) {
2156                 events |= FOP_FILE_TRUNC;
```



```
2157         }
2158         if (vap->va_mask & (AT_SIZE|AT_MTIME)) {
2159                 events |= FOP_FILE_SETATTR_MTIME;
2160         }
2161         if (vap->va_mask & AT_ATIME) {
2162                 events |= FOP_FILE_SETATTR_ATIME;
2163         }
2164         events |= FOP_FILE_SETATTR_CTIME;

2166         port_fop(vp, events, retval);
2167         return (retval);
2168 }
______unchanged_portion_omitted_

2310 /*
2311  * these are events on the watched file/directory
2312  */
2313 int
2314 port_fop_vnevent(femarg_t *vf, vnevent_t vnevent, vnode_t *dvp, char *name,
2315     caller_context_t *ct)
2316 {
2317         vnode_t         *vp = (vnode_t *)vf->fa_fnode->fn_available;

2319         switch (vnevent) {
2320         case    VE_RENAME_SRC:
2321                 port_fop_sendevent(vp, FILE_RENAME_FROM, dvp, name);
2322                 break;
2323         case    VE_RENAME_DEST:
2324                 port_fop_sendevent(vp, FILE_RENAME_TO, dvp, name);
2325                 break;
2326         case    VE_REMOVE:
2327                 port_fop_sendevent(vp, FILE_DELETE, dvp, name);
2328                 break;
2329         case    VE_RMDIR:
2330                 port_fop_sendevent(vp, FILE_DELETE, dvp, name);
2331                 break;
2332         case    VE_CREATE:
2333                 port_fop_sendevent(vp,
2334                     FILE_MODIFIED|FILE_ATTRIB|FILE_TRUNC, NULL, NULL);
2325                 port_fop_sendevent(vp, FILE_MODIFIED|FILE_ATTRIB,
2326                     NULL, NULL);
2335                 break;
2336         case    VE_LINK:
2337                 port_fop_sendevent(vp, FILE_ATTRIB, NULL, NULL);
2338                 break;

2340         case    VE_RENAME_DEST_DIR:
2341                 port_fop_sendevent(vp, FILE_MODIFIED|FILE_ATTRIB,
2342                     NULL, NULL);
2343                 break;

2345         case    VE_MOUNTEDOVER:
2346                 port_fop_sendevent(vp, MOUNTEDOVER, NULL, NULL);
2347                 break;
2348         default:
2349                 break;
2350         }
2351         return (vnext_vnevent(vf, vnevent, dvp, name, ct));
2352 }
______unchanged_portion_omitted_
```

```
**********************************************************
   57203 Fri Jan 18 13:59:20 2013
new/usr/src/uts/common/fs/tmpfs/tmp_vnops.c
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * 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.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/t_lock.h>
#include <sys/systm.h>
#include <sys/sysmacros.h>
#include <sys/user.h>
#include <sys/time.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/vnode.h>
#include <sys/file.h>
#include <sys/fcntl.h>
#include <sys/flock.h>
#include <sys/kmem.h>
#include <sys/uio.h>
#include <sys/errno.h>
#include <sys/stat.h>
#include <sys/cred.h>
#include <sys/dirent.h>
#include <sys/pathname.h>
#include <sys/vmsystm.h>
#include <sys/fs/tmp.h>
#include <sys/fs/tmpnode.h>
#include <sys/mman.h>
#include <vm/hat.h>
#include <vm/seg_vn.h>
#include <vm/seg_map.h>
#include <vm/seg.h>
#include <vm/anon.h>
#include <vm/as.h>
```



```
#include <vm/page.h>
#include <vm/pvn.h>
#include <sys/cmn_err.h>
#include <sys/debug.h>
#include <sys/swap.h>
#include <sys/buf.h>
#include <sys/vm.h>
#include <sys/vtrace.h>
#include <sys/policy.h>
#include <fs/fs_subr.h>

static int      tmp_getapage(struct vnode *, u_offset_t, size_t, uint_t *,
        page_t **, size_t, struct seg *, caddr_t, enum seg_rw, struct cred *);
static int      tmp_putapage(struct vnode *, page_t *, u_offset_t *, size_t *,
        int, struct cred *);

/* ARGSUSED1 */
static int
tmp_open(struct vnode **vpp, int flag, struct cred *cred, caller_context_t *ct)
{
        /*
         * swapon to a tmpfs file is not supported so access
         * is denied on open if VISSWAP is set.
         */
        if ((*vpp)->v_flag & VISSWAP)
                return (EINVAL);
        return (0);
}
_____unchanged_portion_omitted_

/*ARGSUSED7*/
static int
tmp_create(
        struct vnode *dvp,
        char *nm,
        struct vattr *vap,
        enum vcexcl exclusive,
        int mode,
        struct vnode **vpp,
        struct cred *cred,
        int flag,
        caller_context_t *ct,
        vsecattr_t *vsecp)
{
        struct tmpnode *parent;
        struct tmount *tm;
        struct tmpnode *self;
        int error;
        struct tmpnode *oldtp;

again:
        parent = (struct tmpnode *)VTOTN(dvp);
        tm = (struct tmount *)VTOTM(dvp);
        self = NULL;
        error = 0;
        oldtp = NULL;

        /* device files not allowed in ext. attr dirs */
        if ((parent->tn_flags & ISXATTR) &&
            (vap->va_type == VBLK || vap->va_type == VCHR ||
            vap->va_type == VFIFO || vap->va_type == VDOOR ||
            vap->va_type == VSOCK || vap->va_type == VPORT))
                        return (EINVAL);

        if (vap->va_type == VREG && (vap->va_mode & VSVTX)) {
                /* Must be privileged to set sticky bit */
```

```
968                 if (secpolicy_vnode_stky_modify(cred))
969                         vap->va_mode &= ~VSVTX;
970         } else if (vap->va_type == VNON) {
971                 return (EINVAL);
972         }

974         /*
975          * Null component name is a synonym for directory being searched.
976          */
977         if (*nm == '\0') {
978                 VN_HOLD(dvp);
979                 oldtp = parent;
980         } else {
981                 error = tdirlookup(parent, nm, &oldtp, cred);
982         }

984         if (error == 0) {       /* name found */
985                 boolean_t trunc = B_FALSE;

987                 ASSERT(oldtp);

989                 rw_enter(&oldtp->tn_rwlock, RW_WRITER);

991                 /*
992                  * if create/read-only an existing
993                  * directory, allow it
994                  */
995                 if (exclusive == EXCL)
996                         error = EEXIST;
997                 else if ((oldtp->tn_type == VDIR) && (mode & VWRITE))
998                         error = EISDIR;
999                 else {
1000                        error = tmp_taccess(oldtp, mode, cred);
1001                }

1003                if (error) {
1004                        rw_exit(&oldtp->tn_rwlock);
1005                        tmpnode_rele(oldtp);
1006                        return (error);
1007                }
1008                *vpp = TNTOV(oldtp);
1009                if ((*vpp)->v_type == VREG && (vap->va_mask & AT_SIZE) &&
1010                    vap->va_size == 0) {
1011                        rw_enter(&oldtp->tn_contents, RW_WRITER);
1012                        (void) tmpnode_trunc(tm, oldtp, 0);
1013                        rw_exit(&oldtp->tn_contents);
1014                        trunc = B_TRUE;
1015                }
1016                rw_exit(&oldtp->tn_rwlock);
1017                if (IS_DEVVP(*vpp)) {
1018                        struct vnode *newvp;

1020                        newvp = specvp(*vpp, (*vpp)->v_rdev, (*vpp)->v_type,
1021                            cred);
1022                        VN_RELE(*vpp);
1023                        if (newvp == NULL) {
1024                                return (ENOSYS);
1025                        }
1026                        *vpp = newvp;
1027                }

1029                if (trunc)
1022                if (error == 0) {
1030                        vnevent_create(*vpp, ct);

1024                }
```



```
1032                        return (0);
1033        }

1035        if (error != ENOENT)
1036                return (error);

1038        rw_enter(&parent->tn_rwlock, RW_WRITER);
1039        error = tdirenter(tm, parent, nm, DE_CREATE,
1040            (struct tmpnode *)NULL, (struct tmpnode *)NULL,
1041            vap, &self, cred, ct);
1042        rw_exit(&parent->tn_rwlock);

1044        if (error) {
1045                if (self)
1046                        tmpnode_rele(self);

1048                if (error == EEXIST) {
1049                        /*
1050                         * This means that the file was created sometime
1051                         * after we checked and did not find it and when
1052                         * we went to create it.
1053                         * Since creat() is supposed to truncate a file
1054                         * that already exits go back to the begining
1055                         * of the function. This time we will find it
1056                         * and go down the tmp_trunc() path
1057                         */
1058                        goto again;
1059                }
1060                return (error);
1061        }

1063        *vpp = TNTOV(self);

1065        if (!error && IS_DEVVP(*vpp)) {
1066                struct vnode *newvp;

1068                newvp = specvp(*vpp, (*vpp)->v_rdev, (*vpp)->v_type, cred);
1069                VN_RELE(*vpp);
1070                if (newvp == NULL)
1071                        return (ENOSYS);
1072                *vpp = newvp;
1073        }
1074        TRACE_3(TR_FAC_TMPFS, TR_TMPFS_CREATE,
1075            "tmpfs create:dvp %p nm %s vpp %p", dvp, nm, vpp);
1076        return (0);
1077 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3786 Fri Jan 18 13:59:20 2013
new/usr/src/uts/common/sys/port.h
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * 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.
 */

#ifndef _SYS_PORT_H
#define _SYS_PORT_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#ifdef  __cplusplus
extern "C" {
#endif

#include <sys/types.h>

/* port sources */
#define PORT_SOURCE_AIO         1
#define PORT_SOURCE_TIMER       2
#define PORT_SOURCE_USER        3
#define PORT_SOURCE_FD          4
#define PORT_SOURCE_ALERT       5
#define PORT_SOURCE_MQ          6
#define PORT_SOURCE_FILE        7

typedef struct port_event {
        int             portev_events;  /* event data is source specific */
        ushort_t        portev_source;  /* event source */
        ushort_t        portev_pad;     /* port internal use */
        uintptr_t       portev_object;  /* source specific object */
        void            *portev_user;   /* user cookie */
} port_event_t;
```

*_____unchanged_portion_omitted_*

```
#endif /* _SYSCALL32 */
```



```
/* port_alert() flags */
#define PORT_ALERT_SET          0x01
#define PORT_ALERT_UPDATE       0x02
#define PORT_ALERT_INVALID      (PORT_ALERT_SET | PORT_ALERT_UPDATE)

/*
 * PORT_SOURCE_FILE - events
 */

/*
 * User watchable file events
 */
#define FILE_ACCESS             0x00000001
#define FILE_MODIFIED           0x00000002
#define FILE_ATTRIB             0x00000004
#define FILE_TRUNC              0x00100000
#define FILE_NOFOLLOW           0x10000000

/*
 * exception file events
 */

/*
 * The watched file..
 */
#define FILE_DELETE             0x00000010
#define FILE_RENAME_TO          0x00000020
#define FILE_RENAME_FROM        0x00000040
/*
 * The filesystem on which the watched file resides got
 * unmounted.
 */
#define UNMOUNTED               0x20000000
/*
 * Some other file/filesystem got mounted over the
 * watched file/directory.
 */
#define MOUNTEDOVER             0x40000000

/*
 * Helper type
 */
#define FILE_EXCEPTION          (UNMOUNTED|FILE_DELETE|FILE_RENAME_TO \
                                |FILE_RENAME_FROM|MOUNTEDOVER)

#ifdef  __cplusplus
}
```

*_____unchanged_portion_omitted_*

```
**********************************************************
   11682 Fri Jan 18 13:59:21 2013
new/usr/src/uts/common/sys/port_impl.h
3484 enhance and document tail follow support
Reviewed by: Joshua M. Clulow <jmc@joyent.com>
**********************************************************
```

```
/*
 * 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.
 */

#ifndef _SYS_PORT_IMPL_H
#define _SYS_PORT_IMPL_H


#ifdef  __cplusplus
extern "C" {
#endif

/*
 * Note:
 * The contents of this file are private to the implementation of the
 * Solaris system and event ports subsystem and are subject to change
 * at any time without notice.
 */

#include <sys/poll_impl.h>
#include <sys/port.h>
#include <sys/port_kernel.h>
#include <sys/vnode.h>
#include <sys/fem.h>

/*
 * port system call codes
 */
#define PORT_CREATE     0       /* create a port */
#define PORT_ASSOCIATE  1       /* register object or object list */
#define PORT_DISSOCIATE 2       /* remove object association */
#define PORT_SEND       3       /* send user-defined event to a port */
#define PORT_SENDN      4       /* send user-defined event to a list of ports */
#define PORT_GET        5       /* receive object with events */
```



```
#define PORT_GETN       6       /* receive list of objects with events */
#define PORT_ALERT      7       /* set port in alert mode */
#define PORT_DISPATCH   8       /* dispatch object with events */

#define PORT_SYS_NOPORT         0x100   /* system call without port-id */
#define PORT_SYS_NOSHARE        0x200   /* non shareable event */
#define PORT_CODE_MASK          0xff

/* port_dispatch() flags */
#define PORT_SHARE_EVENT        0x01    /* event can be shared between procs */

/* port limits */
#define PORT_MAX_LIST   8192    /* max. # of list ent. per syscall */

#ifdef _KERNEL

#define PORT_SCACHE_SIZE        16      /* start source cache size */
#define PORT_SHASH(cookie)      (cookie & (PORT_SCACHE_SIZE-1))

/* portkev_flags masks */
#define PORT_CLEANUP_DONE       (PORT_KEV_FREE|PORT_KEV_DONEQ)
#define PORT_KEV_CACHE          (PORT_KEV_CACHED|PORT_KEV_SCACHED)
#define PORT_KEV_WIRED          (PORT_KEV_PRIVATE|PORT_KEV_CACHE)

#define PORT_FREE_EVENT(pev)    (((pev)->portkev_flags & PORT_KEV_CACHE) == 0)

typedef struct port_alert {
        int     portal_events;          /* passed to alert event */
        pid_t   portal_pid;             /* owner of the alert mode */
        uintptr_t portal_object;        /* passed to alert event */
        void    *portal_user;           /* passed to alert event */
} port_alert_t;
______unchanged_portion_omitted_

#define PORTFOP_PVFSHASH_SZ     256
#define PORTFOP_PVFSHASH(vfsp)  (((uintptr_t)(vfsp) >> 4) % PORTFOP_PVFSHASH_SZ)

/*
 * file operations flag.
 */

/*
 * PORT_SOURCE_FILE - vnode operations
 */

#define FOP_FILE_OPEN           0x00000001
#define FOP_FILE_READ           0x00000002
#define FOP_FILE_WRITE          0x00000004
#define FOP_FILE_MAP            0x00000008
#define FOP_FILE_IOCTL          0x00000010
#define FOP_FILE_CREATE         0x00000020
#define FOP_FILE_MKDIR          0x00000040
#define FOP_FILE_SYMLINK        0x00000080
#define FOP_FILE_LINK           0x00000100
#define FOP_FILE_RENAME         0x00000200
#define FOP_FILE_REMOVE         0x00000400
#define FOP_FILE_RMDIR          0x00000800
#define FOP_FILE_READDIR        0x00001000
#define FOP_FILE_RENAMESRC      0x00002000
#define FOP_FILE_RENAMEDST      0x00004000
#define FOP_FILE_REMOVEFILE     0x00008000
#define FOP_FILE_REMOVEDIR      0x00010000
#define FOP_FILE_SETSECATTR     0x00020000
#define FOP_FILE_SETATTR_ATIME  0x00040000
#define FOP_FILE_SETATTR_MTIME  0x00080000
#define FOP_FILE_SETATTR_CTIME  0x00100000
```

```
317 #define FOP_FILE_LINK_SRC       0x00200000
318 #define FOP_FILE_TRUNC          0x00400000

320 /*
321  * File modification event.
322  */
323 #define FOP_MODIFIED_MASK       (FOP_FILE_WRITE|FOP_FILE_CREATE \
324                                 |FOP_FILE_REMOVE|FOP_FILE_LINK \
325                                 |FOP_FILE_RENAMESRC|FOP_FILE_RENAMEDST \
326                                 |FOP_FILE_MKDIR|FOP_FILE_RMDIR \
327                                 |FOP_FILE_SYMLINK|FOP_FILE_SETATTR_MTIME)

329 /*
330  * File access event
331  */
332 #define FOP_ACCESS_MASK         (FOP_FILE_READ|FOP_FILE_READDIR \
333                                 |FOP_FILE_MAP|FOP_FILE_SETATTR_ATIME)

335 /*
336  * File attrib event
337  */
338 #define FOP_ATTRIB_MASK         (FOP_FILE_WRITE|FOP_FILE_CREATE \
339                                 |FOP_FILE_REMOVE|FOP_FILE_LINK \
340                                 |FOP_FILE_RENAMESRC|FOP_FILE_RENAMEDST \
341                                 |FOP_FILE_MKDIR|FOP_FILE_RMDIR \
342                                 |FOP_FILE_SYMLINK|FOP_FILE_SETATTR_CTIME \
343                                 |FOP_FILE_LINK_SRC|FOP_FILE_SETSECATTR)


346 /*
347  * File trunc event
348  */
349 #define FOP_TRUNC_MASK          (FOP_FILE_TRUNC|FOP_FILE_CREATE)

351 /*
352  * valid watchable events
353  */
354 #define FILE_EVENTS_MASK        (FILE_ACCESS|FILE_MODIFIED|FILE_ATTRIB \
355                                 |FILE_NOFOLLOW|FILE_TRUNC)
345                                 |FILE_NOFOLLOW)
356 /* --- End file events --- */

358 /*
359  * port_kstat_t contains the event port kernel values which are
360  * exported to kstat.
361  * Currently only the number of active ports is exported.
362  */
363 typedef struct port_kstat {
364         kstat_named_t   pks_ports;
365 } port_kstat_t;
```

*unchanged_portion_omitted_*