```
**********************************************************
   16597 Sun Oct 14 08:17:13 2012
new/usr/src/cmd/tput/tput.c
702 tput calls gets()
**********************************************************
```

```
/*
 * 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) 2012 Gary Mills
 */

/*      Copyright (c) 1988 AT&T */
/*        All Rights Reserved   */

/*
 *      tput - print terminal attribute
 *
 *  return-codes - command line arguments:
 *      0: ok if boolean capname -> TRUE
 *      1: for boolean capname -> FALSE
 *
 *  return-codes - standard input arguments:
 *      0: ok; tput for all lines was successful
 *
 *  return-codes - both cases:
 *      2       usage error
 *      3       bad terminal type given or no terminfo database
 *      4       unknown capname
 *      -1      capname is a numeric variable that is not specified in the
 *              terminfo database(E.g. tpu -T450 lines).
 *
 *  tput printfs a value if an INT capname was given; e.g. cols.
 *      putp's a string if a STRING capname was given; e.g. clear. and
 *  for BOOLEAN capnames, e.g. hard-copy, just returns the boolean value.
 */

#include <curses.h>
#include <term.h>
#include <fcntl.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <locale.h>
```



```
/* externs from libcurses */
extern int tigetnum();

static int outputcap(char *cap, int argc, char **argv);
static int allnumeric(char *string);
static int getpad(char *cap);
static void setdelay();
static void settabs();
static void cat(char *file);
static void initterm();
static void reset_term();

static char *progname;          /* argv[0] */
static int CurrentBaudRate;     /* current baud rate */
static int reset = 0;           /* called as reset_term */
static int fildes = 1;

int
main(int argc, char **argv)
{
        int i, std_argc;
        char *term = getenv("TERM");
        char *cap, std_input = FALSE;
        int setuperr;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)
#define TEXT_DOMAIN "SYS_TEST"
#endif
        (void) textdomain(TEXT_DOMAIN);

        progname = argv[0];

        while ((i = getopt(argc, argv, "ST:")) != EOF) {
                switch (i) {
                case 'T':
                        fildes = -1;
                        (void) putenv("LINES=");
                        (void) putenv("COLUMNS=");
                        term = optarg;
                        break;

                case 'S':
                        std_input = TRUE;
                        break;

                case '?':                       /* FALLTHROUGH          */
                usage:                          /* FALLTHROUGH          */
                default:
                        (void) fprintf(stderr, gettext(
                            "usage:\t%s [-T [term]] capname "
                            "[parm argument...]\n"), progname);
                        (void) fprintf(stderr, gettext("OR:\t%s -S <<\n"),
                            progname);
                        exit(2);
                }
        }

        if (!term || !*term) {
                (void) fprintf(stderr,
                    gettext("%s: No value for $TERM and no -T specified\n"),
                    progname);
                exit(2);
        }
```

```
128         (void) setupterm(term, fildes, &setuperr);

130         switch (setuperr) {
131         case -2:
132                 (void) fprintf(stderr,
133                     gettext("%s: unreadable terminal descriptor \"%s\"\n"),
134                     progname, term);
135                 exit(3);
136                 break;

138         case -1:
139                 (void) fprintf(stderr,
140                     gettext("%s: no terminfo database\n"), progname);
141                 exit(3);
142                 break;

144         case 0:
145                 (void) fprintf(stderr,
146                     gettext("%s: unknown terminal \"%s\"\n"),
147                     progname, term);
148                 exit(3);
149         }

151         reset_shell_mode();

153         /* command line arguments */
154         if (!std_input) {
155                 if (argc == optind)
156                         goto usage;

158                 cap = argv[optind++];

160                 if (strcmp(cap, "init") == 0)
161                         initterm();
162                 else if (strcmp(cap, "reset") == 0)
163                         reset_term();
164                 else if (strcmp(cap, "longname") == 0)
165                         (void) printf("%s\n", longname());
166                 else
167                         exit(outputcap(cap, argc, argv));
168                 return (0);
169         } else {                        /* standard input argumets      */
170                 char buff[128];
171                 char **v;

173                 /* allocate storage for the 'faked' argv[] array        */
174                 v = (char **)malloc(10 * sizeof (char *));
175                 for (i = 0; i < 10; i++)
176                         v[i] = (char *)malloc(32 * sizeof (char));

178                 while (fgets(buff, sizeof (buff), stdin) != NULL) {
177                 while (gets(buff) != NULL) {
179                         /* read standard input line; skip over empty lines */
180                         if ((std_argc =
181                             sscanf(buff,
182                             "%31s %31s %31s %31s %31s %31s %31s %31s "
183                             "%31s %31s",
180                             sscanf(buff, "%s %s %s %s %s %s %s %s %s %s",
184                             v[0], v[1], v[2], v[3], v[4], v[5], v[6], v[7],
185                             v[8], v[9])) < 1) {
186                                 continue;
187                         }

189                         cap = v[0];
190                         optind = 1;
```



```
192                         if (strcmp(cap, "init") == 0) {
193                                 initterm();
194                         } else if (strcmp(cap, "reset") == 0) {
195                                 reset_term();
196                         } else if (strcmp(cap, "longname") == 0) {
197                                 (void) printf("%s\n", longname());
198                         } else {
199                                 (void) outputcap(cap, std_argc, v);
200                         }
201                         (void) fflush(stdout);
202                 }

204                 return (0);
205         }
206 }
```

_____***unchanged_portion_omitted_***