```
**********************************************************
   72156 Thu Feb  9 21:13:53 2012
new/usr/src/cmd/man/src/man.c
man outputs "geqn should have been given a `-Tutf8' option"
**********************************************************
/*
 * 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) 2012, Josef 'Jeff' Sipek <jeffpc@josefsipek.net>. All rights re
 */

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

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */


/*
 * man
 * links to apropos, whatis, and catman
 * This version uses more for underlining and paging.
 */

#include <stdio.h>
#include <ctype.h>
#include <sgtty.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <signal.h>
#include <string.h>
#include <malloc.h>
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <locale.h>
#include <stdlib.h>
#include <unistd.h>
#include <memory.h>
```

```
#include <limits.h>
#include <wchar.h>

#define MACROF  "tmac.an"               /* name of <locale> macro file */
#define TMAC_AN "-man"          /* default macro file */

/*
 * The default search path for man subtrees.
 */

#define MANDIR          "/usr/share/man"        /* default mandir */
#define MAKEWHATIS      "/usr/lib/makewhatis"
#define WHATIS          "windex"
#define TEMPLATE        "/tmp/mpXXXXXX"
#define CONFIG          "man.cf"

/*
 * Names for formatting and display programs.  The values given
 * below are reasonable defaults, but sites with source may
 * wish to modify them to match the local environment.  The
 * value for TCAT is particularly problematic as there's no
 * accepted standard value available for it.  (The definition
 * below assumes C.A.T. troff output and prints it).
 */

#define MORE    "more -s"               /* default paging filter */
#define CAT_S   "/usr/bin/cat -s"       /* for '-' opt (no more) */
#define CAT_    "/usr/bin/cat"          /* for when output is not a tty */
#define TROFF   "troff"                 /* local name for troff */
#define TCAT    "lp -c -T troff"        /* command to "display" troff output */

#define SOLIMIT         10      /* maximum allowed .so chain length */
#define MAXDIRS         128     /* max # of subdirs per manpath */
#define MAXPAGES        128     /* max # for multiple pages */
#define PLEN            3       /* prefix length {man, cat, fmt} */
#define TMPLEN          7       /* length of tmpfile prefix */
#define MAXTOKENS       64

#define DOT_SO          ".so "
#define PREPROC_SPEC    "'\\\" "

#define DPRINTF         if (debug && !catmando) \
                                (void) printf

#define sys(s)          (debug ? ((void)puts(s), 0) : system(s))
#define eq(a, b)        (strcmp(a, b) == 0)
#define match(a, b, c)  (strncmp(a, b, c) == 0)

#define ISDIR(A)        ((A.st_mode & S_IFMT) == S_IFDIR)

#define SROFF_CMD       "/usr/lib/sgml/sgml2roff" /* sgml converter */
#define MANDIRNAME      "man"                     /* man directory */
#define SGMLDIR         "sman"                    /* sman directory */
#define SGML_SYMBOL     "<!DOCTYPE"     /* a sgml file should contain this */
#define SGML_SYMBOL_LEN         9       /* length of SGML_SYMBOL */

/*
 * Directory mapping of old directories to new directories
 */

typedef struct {
        char *old_name;
        char *new_name;
} map_entry;
______unchanged_portion_omitted_
```

```c
static char     *pages[MAXPAGES];
static char     **endp = pages;

/*
 * flags (options)
 */
static int      nomore;
static int      troffit;
static int      debug;
static int      Tflag;
static int      sargs;
static int      margs;
static int      force;
static int      found;
static int      list;
static int      all;
static int      whatis;
static int      apropos;
static int      catmando;
static int      nowhatis;
static int      whatonly;
static int      compargs;       /* -c option for catman */
static int      printmp;

static char     *CAT    = CAT_;
static char     macros[MAXPATHLEN];
static char     *mansec;
static char     *pager;
static char     *troffcmd;
static char     *troffcat;
static char     **subdirs;

static char *check_config(char *);
static struct man_node *build_manpath(char **, int);
static void getpath(struct man_node *, char **);
static void getsect(struct man_node *, char **);
static void get_all_sect(struct man_node *);
static void catman(struct man_node *, char **, int);
static int makecat(char *, char **, int);
static int getdirs(char *, char ***, short);
static void whatapro(struct man_node *, char *, int);
static void lookup_windex(char *, char *, char **);
static int icmp(wchar_t *, wchar_t *);
static void more(char **, int);
static void cleanup(char **);
static void bye(int);
static char **split(char *, char);
static void freev(char **);
static void fullpaths(struct man_node **);
static void lower(char *);
static int cmp(const void *, const void *);
static int manual(struct man_node *, char *);
static void mandir(char **, char *, char *);
static void sortdir(DIR *, char ***);
static int searchdir(char *, char *, char *);
static int windex(char **, char *, char *);
static void section(struct suffix *, char *);
static int bfsearch(FILE *, char **, char *, char **);
static int compare(char *, char *, char **);
static int format(char *, char *, char *, char *);
static char *addlocale(char *);
static int get_manconfig(FILE *, char *);
static void     malloc_error(void);
static int      sgmlcheck(const char *);
static char *map_section(char *, char *);
static void free_manp(struct man_node *manp);
```

```c
static void init_bintoman(void);
static char *path_to_manpath(char *);
static int dupcheck(struct man_node *, struct dupnode **);
static void free_dupnode(struct dupnode *);
static void print_manpath(struct man_node *, char *);

/*
 * This flag is used when the SGML-to-troff converter
 * is absent - all the SGML searches are bypassed.
 */
static int no_sroff = 0;

/*
 * This flag is used to describe the case where we’ve found
 * an SGML formatted manpage in the sman directory, we haven’t
 * found a troff formatted manpage, and we don’t have the SGML to troff
 * conversion utility on the system.
 */
static int sman_no_man_no_sroff;

static char language[PATH_MAX + 1];     /* LC_MESSAGES */
static char localedir[PATH_MAX + 1];    /* locale specific path component */

static int      defaultmandir = 1;      /* if processing default mandir, 1 */

static char *newsection = NULL;

int
main(int argc, char *argv[])
{
        int badopts = 0;
        int c;
        char **pathv;
        char *cmdname;
        char *manpath = NULL;
        static struct man_node  *manpage = NULL;
        int bmp_flags = 0;
        int err = 0;

        if (access(SROFF_CMD, F_OK | X_OK) != 0)
                no_sroff = 1;

        (void) setlocale(LC_ALL, "");
        (void) strcpy(language, setlocale(LC_MESSAGES, (char *)0));
        if (strcmp("C", language) != 0)
                (void) sprintf(localedir, "%s", language);

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

        (void) strcpy(macros, TMAC_AN);

        /*
         * get base part of command name
         */
        if ((cmdname = strrchr(argv[0], ’/’)) != NULL)
                cmdname++;
        else
                cmdname = argv[0];

        if (eq(cmdname, "apropos") || eq(cmdname, "whatis")) {
                whatis++;
                apropos = (*cmdname == ’a’);
                if ((optind = 1) == argc) {
                        (void) fprintf(stderr, gettext("%s what?\n"), cmdname);
                        exit(2);
                }
                goto doargs;
        } else if (eq(cmdname, "catman"))
                catmando++;

        opterr = 0;
        while ((c = getopt(argc, argv, opts[catmando])) != -1)
                switch (c) {

                /*
                 * man specific options
                 */
                case 'k':
                        apropos++;
                        /*FALLTHROUGH*/
                case 'f':
                        whatis++;
                        break;
                case 'F':
                        force++;        /* do lookups the hard way */
                        break;
                case 's':
                        mansec = optarg;
                        sargs++;
                        break;
                case 'r':
                        nomore++, troffit++;
                        break;
                case 'l':
                        list++;         /* implies all */
                        /*FALLTHROUGH*/
                case 'a':
                        all++;
                        break;
                case 'd':
                        debug++;
                        break;
                /*
                 * man and catman use -p differently.  In catman it
                 * enables debug mode and in man it prints the (possibly
                 * derived from PATH or name operand) MANPATH.
                 */
                case 'p':
                        if (catmando == 0) {
                                printmp++;
                        } else {
                                debug++;
                        }
                        break;
                case 'n':
                        nowhatis++;
                        break;
                case 'w':
                        whatonly++;
                        break;
                case 'c':       /* n|troff compatibility */
                        if (no_sroff)
                                (void) fprintf(stderr, gettext(
                                    "catman: SGML conversion not "
                                    "available -- -c flag ignored\n"));
                        else
                                compargs++;
                        continue;
```

```
                /*
                 * shared options
                 */
                case 'P':       /* Backwards compatibility */
                case 'M':       /* Respecify path for man pages. */
                        manpath = optarg;
                        margs++;
                        break;
                case 'T':       /* Respecify man macros */
                        (void) strcpy(macros, optarg);
                        Tflag++;
                        break;
                case 't':
                        troffit++;
                        break;
                case '?':
                        badopts++;
                }

        /*
         *  Bad options or no args?
         *      (man -p and catman don't need args)
         */
        if (badopts || (!catmando && !printmp && optind == argc)) {
                (void) fprintf(stderr, "%s\n", catmando ?
                    gettext(CATMAN_USAGE) : gettext(MAN_USAGE));
                exit(2);
        }

        if (compargs && (nowhatis || whatonly || troffit)) {
                (void) fprintf(stderr, "%s\n", gettext(CATMAN_USAGE));
                (void) fprintf(stderr, gettext(
                    "-c option cannot be used with [-w][-n][-t]\n"));
                exit(2);
        }

        if (sargs && margs && catmando) {
                (void) fprintf(stderr, "%s\n", gettext(CATMAN_USAGE));
                exit(2);
        }

        if (troffit == 0 && nomore == 0 && !isatty(fileno(stdout)))
                nomore++;

        /*
         * Collect environment information.
         */
        if (troffit) {
                if ((troffcmd = getenv("TROFF")) == NULL)
                        troffcmd = TROFF;
                if ((troffcat = getenv("TCAT")) == NULL)
                        troffcat = TCAT;
        } else {
                if (((pager = getenv("PAGER")) == NULL) ||
                    (*pager == NULL))
                        pager = MORE;
        }

doargs:
        subdirs = troffit ? troffdirs : nroffdirs;

        init_bintoman();

        if (manpath == NULL && (manpath = getenv("MANPATH")) == NULL) {
                if ((manpath = getenv("PATH")) != NULL) {
                        bmp_flags = BMP_ISPATH | BMP_APPEND_MANDIR;
```

```
		} else {
			manpath = MANDIR;
		}
	}

	pathv = split(manpath, ':');

	manpage = build_manpath(pathv, bmp_flags);

	/* release pathv allocated by split() */
	freev(pathv);

	/*
	 * Since we can't make use of GNU troff, set the path to ensure we
	 * find the one in /usr/bin first.
	 */
	if (putenv("PATH=/usr/bin") != 0) {
		perror("putenv");
		exit(1);
	}

	fullpaths(&manpage);

	if (catmando) {
		catman(manpage, argv+optind, argc-optind);
		exit(0);
	}

	/*
	 * The manual routine contains windows during which
	 * termination would leave a temp file behind.  Thus
	 * we blanket the whole thing with a clean-up routine.
	 */
	if (signal(SIGINT, SIG_IGN) == SIG_DFL) {
		(void) signal(SIGINT, bye);
		(void) signal(SIGQUIT, bye);
		(void) signal(SIGTERM, bye);
	}

	/*
	 * "man -p" without operands
	 */
	if ((printmp != 0) && (optind == argc)) {
		print_manpath(manpage, NULL);
		exit(0);
	}

	for (; optind < argc; optind++) {
		if (strcmp(argv[optind], "-") == 0) {
			nomore++;
			CAT = CAT_S;
		} else {
			char *cmd;
			static struct man_node *mp;
			char *pv[2];

			/*
			 * If full path to command specified, customize
			 * manpath accordingly
			 */
			if ((cmd = strrchr(argv[optind], '/')) != NULL) {
				*cmd = '\0';
				if ((pv[0] = strdup(argv[optind])) == NULL) {
					malloc_error();
				}
				pv[1] = NULL;
```

```
				*cmd = '/';
				mp = build_manpath(pv,
				    BMP_ISPATH|BMP_FALLBACK_MANDIR);
			} else {
				mp = manpage;
			}

			if (whatis) {
				whatapro(mp, argv[optind], apropos);
			} else if (printmp != 0) {
				print_manpath(mp, argv[optind]);
			} else {
				err += manual(mp, argv[optind]);
			}

			if (mp != NULL && mp != manpage) {
				free(pv[0]);
				free_manp(mp);
			}
		}
	}
	return (err == 0 ? 0 : 1);
	/*NOTREACHED*/
}
```
______***unchanged_portion_omitted_***