```
**********************************************************
   71946 Fri Feb  3 11:00:51 2012
new/usr/src/cmd/man/src/man.c
*** NO COMMENTS ***
**********************************************************
______unchanged_portion_omitted_
 154         {'c',   "cw",                           "cw",           "-"},
 155         {'e',   "/usr/bin/neqn /usr/share/lib/pub/eqnchar",
 156                         "/usr/bin/eqn /usr/share/lib/pub/eqnchar",      "-"},
 155         {'e',   "neqn /usr/share/lib/pub/eqnchar",
 156                         "eqn /usr/share/lib/pub/eqnchar",       "-"},
 157         {'p',   "gpic",                         "gpic",         "-"},
 158         {'r',   "refer",                        "refer",        "-"},
 159         {'t',   "tbl",                          "tbl",          ""},
 160         {'v',   "vgrind -f",                    "vgrind -f",    "-"},
 161         {0,     0,                              0,              0}
 162 };
______unchanged_portion_omitted_


2320 /*
2321  * Format a man page and follow .so references
2322  * if necessary.
2323  */

2325 static int
2326 format(char *path, char *dir, char *name, char *pg)
2327 {
2328         char manpname[MAXPATHLEN+1], catpname[MAXPATHLEN+1];
2329         char manpname_sgml[MAXPATHLEN+1], smantmpname[MAXPATHLEN+1];
2330         char soed[MAXPATHLEN+1], soref[MAXPATHLEN+1];
2331         char manbuf[BUFSIZ], cmdbuf[BUFSIZ], tmpbuf[BUFSIZ];
2332         char tmpdir[MAXPATHLEN+1];
2333         int socount, updatedcat, regencat;
2334         struct stat mansb, catsb, smansb;
2335         char *tmpname;
2336         int catonly = 0;
2337         struct stat statb;
2338         int plen = PLEN;
2339         FILE *md;
2340         int tempfd;
2341         ssize_t count;
2342         int     temp, sgml_flag = 0, check_flag = 0;
2343         char prntbuf[BUFSIZ + 1];
2344         char *ptr;
2345         char *new_m;
2346         char    *tmpsubdir;

2348         found++;

2350         if (*dir != 'm' && *dir != 's')
2351                 catonly++;


2354         if (*dir == 's') {
2355                 tmpsubdir = SGMLDIR;
2356                 ++plen;
2357                 (void) sprintf(manpname_sgml, "%s/man%s/%s",
2358                     path, dir+plen, pg);
2359         } else
2360                 tmpsubdir = MANDIRNAME;

2362         if (list) {
2363                 (void) printf(gettext("%s (%s)\t-M %s\n"),
2364                     name, dir+plen, path);
2365                 return (-1);
```



```
2366         }

2368         (void) sprintf(manpname, "%s/%s%s/%s", path, tmpsubdir, dir+plen, pg);
2369         (void) sprintf(catpname, "%s/%s%s/%s", path, subdirs[1], dir+plen, pg);

2371         (void) sprintf(smantmpname, "%s/%s%s/%s", path, SGMLDIR, dir+plen, pg);

2373 /*
2374  * TRANSLATION_NOTE - message for man -d or catman -p
2375  * ex.  unformatted = /usr/share/man/ja/man3s/printf.3s
2376  */
2377         DPRINTF(gettext(
2378             "      unformatted = %s\n"), catonly ? "" : manpname);
2379 /*
2380  * TRANSLATION_NOTE - message for man -d or catman -p
2381  * ex.  formatted = /usr/share/man/ja/cat3s/printf.3s
2382  */
2383         DPRINTF(gettext(
2384             "      formatted = %s\n"), catpname);

2386         /*
2387          * Take care of indirect references to other man pages;
2388          * i.e., resolve files containing only ".so manx/file.x".
2389          * We follow .so chains, replacing title with the .so'ed
2390          * file at each stage, and keeping track of how many times
2391          * we've done so, so that we can avoid looping.
2392          */
2393         *soed = 0;
2394         socount = 0;
2395         for (;;) {
2396                 FILE *md;
2397                 char *cp;
2398                 char *s;
2399                 char *new_s;

2401                 if (catonly)
2402                         break;
2403                 /*
2404                  * Grab manpname's first line, stashing it in manbuf.
2405                  */


2408                 if ((md = fopen(manpname, "r")) == NULL) {
2409                         if (*soed && errno == ENOENT) {
2410                                 (void) fprintf(stderr,
2411                                     gettext("Can't find referent of "
2412                                     ".so in %s\n"), soed);
2413                                 (void) fflush(stderr);
2414                                 return (-1);
2415                         }
2416                         perror(manpname);
2417                         return (-1);
2418                 }

2420                 /*
2421                  * If this is a directory, just ignore it.
2422                  */
2423                 if (fstat(fileno(md), &statb) == NULL) {
2424                         if (S_ISDIR(statb.st_mode)) {
2425                                 if (debug) {
2426                                         (void) fprintf(stderr,
2427                                             "\tignoring directory %s\n",
2428                                             manpname);
2429                                         (void) fflush(stderr);
2430                                 }
2431                                 (void) fclose(md);
```

```
2432                                    return (-1);
2433                            }
2434                    }

2436                    if (fgets(manbuf, BUFSIZ-1, md) == NULL) {
2437                            (void) fclose(md);
2438                            (void) fprintf(stderr, gettext("%s: null file\n"),
2439                                manpname);
2440                            (void) fflush(stderr);
2441                            return (-1);
2442                    }
2443                    (void) fclose(md);

2445                    if (strncmp(manbuf, DOT_SO, sizeof (DOT_SO) - 1))
2446                            break;
2447 so_again:          if (++socount > SOLIMIT) {
2448                            (void) fprintf(stderr, gettext(".so chain too long\n"));
2449                            (void) fflush(stderr);
2450                            return (-1);
2451                    }
2452                    s = manbuf + sizeof (DOT_SO) - 1;
2453                    if ((check_flag == 1) && ((new_s = strrchr(s, '/')) != NULL)) {
2454                            new_s++;
2455                            (void) sprintf(s, "%s%s/%s",
2456                                tmpsubdir, dir+plen, new_s);
2457                    }

2459                    cp = strrchr(s, '\n');
2460                    if (cp)
2461                            *cp = '\0';
2462                    /*
2463                     * Compensate for sloppy typists by stripping
2464                     * trailing white space.
2465                     */
2466                    cp = s + strlen(s);
2467                    while (--cp >= s && (*cp == ' ' || *cp == '\t'))
2468                            *cp = '\0';

2470                    /*
2471                     * Go off and find the next link in the chain.
2472                     */
2473                    (void) strcpy(soed, manpname);
2474                    (void) strcpy(soref, s);
2475                    (void) sprintf(manpname, "%s/%s", path, s);
2476 /*
2477  * TRANSLATION_NOTE - message for man -d or catman -p
2478  * ex.  .so ref = man3c/string.3c
2479  */
2480                    DPRINTF(gettext(".so ref = %s\n"), s);
2481            }

2483            /*
2484             * Make symlinks if so'ed and cattin'
2485             */
2486            if (socount && catmando) {
2487                    (void) sprintf(cmdbuf, "cd %s; rm -f %s; ln -s ../%s%s %s",
2488                        path, catpname, subdirs[1], soref+plen, catpname);
2489                    (void) sys(cmdbuf);
2490                    return (1);
2491            }

2493            /*
2494             * Obtain the cat page that corresponds to the man page.
2495             * If it already exists, is up to date, and if we haven't
2496             * been told not to use it, use it as it stands.
2497             */
```



```
2498            regencat = updatedcat = 0;
2499            if (compargs || (!catonly && stat(manpname, &mansb) >= 0 &&
2500                (stat(catpname, &catsb) < 0 || catsb.st_mtime < mansb.st_mtime)) ||
2501                (access(catpname, R_OK) != 0)) {
2502                    /*
2503                     * Construct a shell command line for formatting manpname.
2504                     * The resulting file goes initially into /tmp.  If possible,
2505                     * it will later be moved to catpname.
2506                     */

2508                    int pipestage = 0;
2509                    int needcol = 0;
2510                    char *cbp = cmdbuf;

2512                    regencat = updatedcat = 1;

2514                    if (!catmando && !debug && !check_flag) {
2515                            (void) fprintf(stderr, gettext(
2516                                "Reformatting page.  Please Wait..."));
2517                            if (sargs && (newsection != NULL) &&
2518                                (*newsection != '\0')) {
2519                                    (void) fprintf(stderr, gettext(
2520                                        "\nThe directory name has been changed "
2521                                        "to %s\n"), newsection);
2522                            }
2523                            (void) fflush(stderr);
2524                    }

2526                    /*
2527                     * in catman command, if the file exists in sman dir already,
2528                     * don't need to convert the file in man dir to cat dir
2529                     */

2531                    if (!no_sroff && catmando &&
2532                        match(tmpsubdir, MANDIRNAME, PLEN) &&
2533                        stat(smantmpname, &smansb) >= 0)
2534                            return (1);

2536                    /*
2537                     * cd to path so that relative .so commands will work
2538                     * correctly
2539                     */
2540                    (void) sprintf(cbp, "cd %s; ", path);
2541                    cbp += strlen(cbp);


2544                    /*
2545                     * check to see whether it is a sgml file
2546                     * assume sgml symbol(>!DOCTYPE) can be found in the first
2547                     * BUFSIZ bytes
2548                     */

2550                    if ((temp = open(manpname, 0)) == -1) {
2551                                    perror(manpname);
2552                                    return (-1);
2553                    }

2555                    if ((count = read(temp, prntbuf, BUFSIZ)) <= 0) {
2556                                    perror(manpname);
2557                                    return (-1);
2558                    }

2560                    prntbuf[count] = '\0';  /* null terminate */
2561                    ptr = prntbuf;
2562                    if (sgmlcheck((const char *)ptr) == 1) {
2563                            sgml_flag = 1;
```

```
2564                        if (defaultmandir && *localedir) {
2565                                (void) sprintf(cbp, "LC_MESSAGES=C %s %s ",
2566                                    SROFF_CMD, manpname);
2567                        } else {
2568                                (void) sprintf(cbp, "%s %s ",
2569                                    SROFF_CMD, manpname);
2570                        }
2571                        cbp += strlen(cbp);
2572                } else if (*dir == 's') {
2573                        (void) close(temp);
2574                        return (-1);
2575                }
2576                (void) close(temp);

2578                /*
2579                 * Check for special formatting requirements by examining
2580                 * manpname's first line preprocessor specifications.
2581                 */

2583                if (strncmp(manbuf, PREPROC_SPEC,
2584                    sizeof (PREPROC_SPEC) - 1) == 0) {
2585                        char *ptp;

2587                        ptp = manbuf + sizeof (PREPROC_SPEC) - 1;
2588                        while (*ptp && *ptp != '\n') {
2589                                const struct preprocessor *pp;

2591                                /*
2592                                 * Check for a preprocessor we know about.
2593                                 */
2594                                for (pp = preprocessors; pp->p_tag; pp++) {
2595                                        if (pp->p_tag == *ptp)
2596                                                break;
2597                                }
2598                                if (pp->p_tag == 0) {
2599                                        (void) fprintf(stderr,
2600                                            gettext("unknown preprocessor "
2601                                            "specifier %c\n"), *ptp);
2602                                        (void) fflush(stderr);
2603                                        return (-1);
2604                                }

2606                                /*
2607                                 * Add it to the pipeline.
2608                                 */
2609                                (void) sprintf(cbp, "%s %s |",
2610                                    troffit ? pp->p_troff : pp->p_nroff,
2611                                    pipestage++ == 0 ? manpname :
2612                                    pp->p_stdin_char);
2613                                cbp += strlen(cbp);

2615                                /*
2616                                 * Special treatment: if tbl is among the
2617                                 * preprocessors and we'll process with
2618                                 * nroff, we have to pass things through
2619                                 * col at the end of the pipeline.
2620                                 */
2621                                if (pp->p_tag == 't' && !troffit)
2622                                        needcol++;

2624                                ptp++;
2625                        }
2626                }

2628                /*
2629                 * if catman, use the cat page name
```



```
2630                 * otherwise, dup template and create another
2631                 * (needed for multiple pages)
2632                 */
2633                if (catmando)
2634                        tmpname = catpname;
2635                else {
2636                        tmpname = strdup(TEMPLATE);
2637                        if (tmpname == NULL)
2638                                malloc_error();
2639                        (void) close(mkstemp(tmpname));
2640                }

2642                if (! Tflag) {
2643                        if (*localedir != '\0') {
2644                                (void) sprintf(macros, "%s/%s", path, MACROF);
2645 /*
2646  * TRANSLATION_NOTE - message for man -d or catman -p
2647  * ex.  locale macros = /usr/share/man/ja/tmac.an
2648  */
2649                                if (debug)
2650                                        (void) printf(gettext(
2651                                            "\nlocale macros = %s "),
2652                                            macros);
2653                                if (stat(macros, &statb) < 0)
2654                                        (void) strcpy(macros, TMAC_AN);
2655 /*
2656  * TRANSLATION_NOTE - message for man -d or catman -p
2657  * ex.  macros = /usr/share/man/ja/tman.an
2658  */
2659                                if (debug)
2660                                        (void) printf(gettext(
2661                                            "\nmacros = %s\n"),
2662                                            macros);
2663                        }
2664                }

2666                tmpdir[0] = '\0';
2667                if (sgml_flag == 1) {
2668                        if (check_flag == 0) {
2669                                strcpy(tmpdir, "/tmp/sman_XXXXXX");
2670                                if ((tempfd = mkstemp(tmpdir)) == -1) {
2671                                        (void) fprintf(stderr, gettext(
2672                                            "%s: null file\n"), tmpdir);
2673                                        (void) fflush(stderr);
2674                                        return (-1);
2675                                }

2677                                if (debug)
2678                                        close(tempfd);

2680                                (void) sprintf(tmpbuf, "%s > %s",
2681                                    cmdbuf, tmpdir);
2682                                if (sys(tmpbuf)) {
2683 /*
2684  * TRANSLATION_NOTE - message for man -d or catman -p
2685  * Error message if sys(%s) failed
2686  */
2687                                        (void) fprintf(stderr, gettext(
2688                                            "sys(%s) fail!\n"), tmpbuf);
2689                                        (void) fprintf(stderr,
2690                                            gettext(" aborted (sorry)\n"));
2691                                        (void) fflush(stderr);
2692                                        /* release memory for tmpname */
2693                                        if (!catmando) {
2694                                                (void) unlink(tmpdir);
2695                                                (void) unlink(tmpname);
```

```
2696                                            free(tmpname);
2697                                    }
2698                                    return (-1);
2699                            } else if (debug == 0) {
2700                                    if ((md = fdopen(tempfd, "r"))
2701                                        == NULL) {
2702                                            (void) fprintf(stderr, gettext(
2703                                                "%s: null file\n"), tmpdir);
2704                                            (void) fflush(stderr);
2705                                            close(tempfd);
2706                                            /* release memory for tmpname */
2707                                            if (!catmando)
2708                                                    free(tmpname);
2709                                            return (-1);
2710                                    }

2712                                    /* if the file is empty, */
2713                                    /* it's a fragment, do nothing */
2714                                    if (fgets(manbuf, BUFSIZ-1, md)
2715                                        == NULL) {
2716                                            (void) fclose(md);
2717                                            /* release memory for tmpname */
2718                                            if (!catmando)
2719                                                    free(tmpname);
2720                                            return (1);
2721                                    }
2722                                    (void) fclose(md);

2724                                    if (strncmp(manbuf, DOT_SO,
2725                                        sizeof (DOT_SO) - 1) == 0) {
2726                                            if (!compargs) {
2727                                            check_flag = 1;
2728                                            (void) unlink(tmpdir);
2729                                            (void) unlink(tmpname);
2730                                            /* release memory for tmpname */
2731                                            if (!catmando)
2732                                                    free(tmpname);
2733                                            goto so_again;
2734                                            } else {
2735                                                    (void) unlink(tmpdir);
2736                                            strcpy(tmpdir,
2737                                                "/tmp/sman_XXXXXX");
2738                                            tempfd = mkstemp(tmpdir);
2739                                            if ((tempfd == -1) ||
2740                                                (md = fdopen(tempfd, "w"))
2741                                                == NULL) {
2742                                                    (void) fprintf(stderr,
2743                                                        gettext(
2744                                                        "%s: null file\n"),
2745                                                        tmpdir);
2746                                                    (void) fflush(stderr);
2747                                                    if (tempfd != -1)
2748                                                            close(tempfd);
2749                                            /* release memory for tmpname */
2750                                                    if (!catmando)
2751                                                            free(tmpname);
2752                                                    return (-1);
2753                                            }
2754                                    if ((new_m = strrchr(manbuf, '/')) != NULL) {
2755                    (void) fprintf(md, ".so man%s%s\n", dir+plen, new_m);
2756                                                    } else {
2757 /*
2758  * TRANSLATION_NOTE - message for catman -c
2759  * Error message if unable to get file name
2760  */
2761                                    (void) fprintf(stderr,
```



```
2762                                        gettext("file not found\n"));
2763                                    (void) fflush(stderr);
2764                                    return (-1);
2765                                    }
2766                                                            (void) fclose(md);
2767                                                    }
2768                                            }
2769                                    }
2770                                    if (catmando && compargs)
2771                                            (void) sprintf(cmdbuf, "cat %s > %s",
2772                                                tmpdir, manpname_sgml);
2773                                    else
2774            (void) sprintf(cmdbuf, " cat %s | tbl | /usr/bin/eqn | %s %s - %s > %s",
2775                                        tmpdir, troffit ? troffcmd : "/usr/bin/nroff -u0 -Tlp",
2774            (void) sprintf(cmdbuf, " cat %s | tbl | eqn | %s %s - %s > %s",
2775                                        tmpdir, troffit ? troffcmd : "nroff -u0 -Tlp",
2776                                        macros, troffit ? "" : " | col -x", tmpname);
2777                                                } else
2778                                    if (catmando && compargs)
2779                                            (void) sprintf(cbp, " > %s",
2780                                                manpname_sgml);
2781                                    else
2782            (void) sprintf(cbp, " | tbl | /usr/bin/eqn | %s %s - %s > %s",
2783                                        troffit ? troffcmd : "/usr/bin/nroff -u0 -Tlp",
2782            (void) sprintf(cbp, " | tbl | eqn | %s %s - %s > %s",
2783                                        troffit ? troffcmd : "nroff -u0 -Tlp",
2784                                        macros, troffit ? "" : " | col -x", tmpname);

2786                                        } else
2787                    (void) sprintf(cbp, "%s %s %s%s > %s",
2788                                        troffit ? troffcmd : "/usr/bin/nroff -u0 -Tlp",
2788                                        troffit ? troffcmd : "nroff -u0 -Tlp",
2789                                        macros, pipestage == 0 ? manpname : "-",
2790                                        troffit ? "" : " | col -x", tmpname);

2792                                    /* Reformat the page. */
2793                                    if (sys(cmdbuf)) {
2794 /*
2795  * TRANSLATION_NOTE - message for man -d or catman -p
2796  * Error message if sys(%s) failed
2797  */
2798                                            (void) fprintf(stderr, gettext(
2799                                                "sys(%s) fail!\n"), cmdbuf);
2800                                            (void) fprintf(stderr, gettext(" aborted (sorry)\n"));
2801                                            (void) fflush(stderr);
2802                                            (void) unlink(tmpname);
2803                                            /* release memory for tmpname */
2804                                            if (!catmando)
2805                                                    free(tmpname);
2806                                            return (-1);
2807                                    }

2809                                    if (tmpdir[0] != '\0')
2810                                            (void) unlink(tmpdir);

2812                                    if (catmando)
2813                                            return (1);

2815                                    /*
2816                                     * Attempt to move the cat page to its proper home.
2817                                     */
2818                                    (void) sprintf(cmdbuf,
2819                                        "trap '' 1 15; /usr/bin/mv -f %s %s 2> /dev/null",
2820                                        tmpname,
2821                                        catpname);
2822                                    if (sys(cmdbuf))
```

```
2823                        updatedcat = 0;
2824                else if (debug == 0)
2825                        (void) chmod(catpname, 0644);

2827                if (debug) {
2828                        /* release memory for tmpname */
2829                        if (!catmando)
2830                                free(tmpname);
2831                        (void) unlink(tmpname);
2832                        return (1);
2833                }

2835                (void) fprintf(stderr, gettext(" done\n"));
2836                (void) fflush(stderr);
2837        }

2839        /*
2840         * Save file name (dup if necessary)
2841         * to view later
2842         * fix for 1123802 - don't save names if we are invoked as catman
2843         */
2844        if (!catmando) {
2845                char    **tmpp;
2846                int     dup;
2847                char    *newpage;

2849                if (regencat && !updatedcat)
2850                        newpage = tmpname;
2851                else {
2852                        newpage = strdup(catpname);
2853                        if (newpage == NULL)
2854                                malloc_error();
2855                }
2856                /* make sure we don't add a dup */
2857                dup = 0;
2858                for (tmpp = pages; tmpp < endp; tmpp++) {
2859                        if (strcmp(*tmpp, newpage) == 0) {
2860                                dup = 1;
2861                                break;
2862                        }
2863                }
2864                if (!dup)
2865                        *endp++ = newpage;
2866                if (endp >= &pages[MAXPAGES]) {
2867                        fprintf(stderr,
2868                            gettext("Internal pages array overflow!\n"));
2869                        exit(1);
2870                }
2871        }

2873        return (regencat);
2874 }
```

_____***unchanged_portion_omitted_***