

```
*****
```

```
26024 Thu Jun 28 19:00:56 2012
```

```
new/usr/src/cmd/sgs/elfdump/common/main.c
```

```
2940 elfdump should return non-null on invalid file type
```

```
*****
```

```
_____unchanged_portion_omitted_____
```

```

726 int
727 main(int argc, char **argv, char **envp)
728 {
729     Elf          *elf;
730     int          var, fd, wfd = 0;
731     char         *wname = NULL;
732     uint_t      flags = 0;
733     match_rec_t match_data;
734     int         ret;
735     uchar_t     osabi;

737     /*
738      * If we're on a 64-bit kernel, try to exec a full 64-bit version of
739      * the binary.  If successful, conv_check_native() won't return.
740      */
741     (void) conv_check_native(argv, envp);

743     /*
744      * Establish locale.
745      */
746     (void) setlocale(LC_MESSAGES, MSG_ORIG(MSG_STR_EMPTY));
747     (void) textdomain(MSG_ORIG(MSG_SUNW_OST_SGS));

749     (void) setvbuf(stdout, NULL, _IOLBF, 0);
750     (void) setvbuf(stderr, NULL, _IOLBF, 0);

752     opterr = 0;
753     while ((var = getopt(argc, argv, MSG_ORIG(MSG_STR_OPTIONS))) != EOF) {
754         switch (var) {
755             case 'C':
756                 flags |= FLG_CTL_DEMANGLE;
757                 break;
758             case 'c':
759                 flags |= FLG_SHOW_SHDR;
760                 break;
761             case 'd':
762                 flags |= FLG_SHOW_DYNAMIC;
763                 break;
764             case 'e':
765                 flags |= FLG_SHOW_EHDR;
766                 break;
767             case 'G':
768                 flags |= FLG_SHOW_GOT;
769                 break;
770             case 'g':
771                 flags |= FLG_SHOW_GROUP;
772                 break;
773             case 'H':
774                 flags |= FLG_SHOW_CAP;
775                 break;
776             case 'h':
777                 flags |= FLG_SHOW_HASH;
778                 break;
779             case 'I':
780                 if (!process_index_opt(optarg, &match_data))
781                     goto usage_brief;
782                 if (!add_match_record(argv[0], &match_data))
783                     return (1);
784                 flags |= FLG_CTL_MATCH;

```

```

785         break;
786     case 'i':
787         flags |= FLG_SHOW_INTERP;
788         break;
789     case 'k':
790         flags |= FLG_CALC_CHECKSUM;
791         break;
792     case 'l':
793         flags |= FLG_CTL_LONGNAME;
794         break;
795     case 'm':
796         flags |= FLG_SHOW_MOVE;
797         break;
798     case 'N':
799         match_data.opt_type = MATCH_OPT_NAME;
800         match_data.value.name = optarg;
801         if (!add_match_record(argv[0], &match_data))
802             return (1);
803         flags |= FLG_CTL_MATCH;
804         break;
805     case 'n':
806         flags |= FLG_SHOW_NOTE;
807         break;
808     case 'O':
809         {
810             uint32_t val;

812             /*
813              * osabi is a uchar_t in the ELF header.
814              * Don't accept any value that exceeds
815              * that range.
816              */
817             if ((atoi(optarg, ATOU_OSABI, &val) == 0) ||
818                 (val > 255)) {
819                 (void) fprintf(stderr,
820                     MSG_INTL(MSG_ERR_BAD_T_OSABI),
821                     basename(argv[0]), optarg);
822                 return (1);
823             }
824             osabi = val;
825         }
826         flags |= FLG_CTL_OSABI;
827         break;
828     case 'P':
829         flags |= FLG_CTL_FAKESHDR;
830         break;
831     case 'p':
832         flags |= FLG_SHOW_PHDR;
833         break;
834     case 'r':
835         flags |= FLG_SHOW_RELOC;
836         break;
837     case 'S':
838         flags |= FLG_SHOW_SORT;
839         break;
840     case 's':
841         flags |= FLG_SHOW_SYMBOLS;
842         break;
843     case 'T':
844         /*
845          * We can't evaluate the value yet, because
846          * we need to know if -p is used or not in
847          * order to tell if we're seeing section header
848          * or program header types. So, we save the
849          * string in the name field, and then convert
850          * it to a type integer in a following pass.

```

```

851     */
852     match_data.opt_type = MATCH_OPT_TYPE;
853     match_data.value.name = optarg;
854     if (!add_match_record(argv[0], &match_data))
855         return (1);
856     flags |= FLG_CTL_MATCH;
857     break;
858 case 'u':
859     flags |= FLG_SHOW_UNWIND;
860     break;
861 case 'v':
862     flags |= FLG_SHOW_VERSIONS;
863     break;
864 case 'w':
865     wname = optarg;
866     break;
867 case 'y':
868     flags |= FLG_SHOW_SYMINFO;
869     break;
870 case '?':
871     (void) fprintf(stderr, MSG_INTL(MSG_USAGE_BRIEF),
872                 basename(argv[0]));
873     detail_usage();
874     return (1);
875 default:
876     break;
877 }
878
880 /* -p and -w are mutually exclusive. -w only works with sections */
881 if ((flags & FLG_SHOW_PHDR) != 0) && (wname != NULL))
882     goto usage_brief;
883
884 /* If a match argument is present, prepare the match state */
885 if ((match_state.list != NULL) && (match_prepare(argv[0], flags) == 0))
886     return (1);
887
888 /*
889  * Decide what to do if no options specifying something to
890  * show or do are present.
891  *
892  * If there is no -w and no match options, then we will set all
893  * the show flags, causing a full display of everything in the
894  * file that we know how to handle.
895  *
896  * Otherwise, if there is no match list, we generate a usage
897  * error and quit.
898  *
899  * In the case where there is a match list, we go ahead and call
900  * regular() anyway, leaving it to decide what to do. If -w is
901  * present, regular() will use the match list to handle it.
902  * In addition, in the absence of explicit show/calc flags, regular()
903  * will compare the section headers to the match list and use
904  * that to generate the FLG_bits that will display the information
905  * specified by the match list.
906  */
907 if ((flags & ~FLG_MASK_CTL) == 0) {
908     if (!wname && (match_state.list == NULL))
909         flags |= FLG_MASK_SHOW;
910     else if (match_state.list == NULL)
911         goto usage_brief;
912 }
913
914 /* There needs to be at least 1 filename left following the options */
915 if ((var = argc - optind) == 0)
916     goto usage_brief;

```

```

918     /*
919     * If the -l/-C option is specified, set up the liblddbg.so.
920     */
921     if (flags & FLG_CTL_LONGNAME)
922         dbg_desc->d_extra |= DBG_E_LONG;
923     if (flags & FLG_CTL_DEMANGLE)
924         dbg_desc->d_extra |= DBG_E_DEMANGLE;
925
926     /*
927     * If the -w option has indicated an output file open it. It's
928     * arguable whether this option has much use when multiple files are
929     * being processed.
930     *
931     * If wname is non-NULL, we know that -p was not specified, due
932     * to the test above.
933     */
934     if (wname) {
935         if ((wfd = open(wname, (O_RDWR | O_CREAT | O_TRUNC),
936                        0666)) < 0) {
937             int err = errno;
938             (void) fprintf(stderr, MSG_INTL(MSG_ERR_OPEN),
939                             wname, strerror(err));
940             return (1);
941         }
942     }
943
944     /*
945     * Open the input file, initialize the elf interface, and
946     * process it.
947     */
948     ret = 0;
949     for (; (optind < argc) && (ret == 0); optind++) {
950         const char *file = argv[optind];
951
952         if ((fd = open(argv[optind], O_RDONLY)) == -1) {
953             int err = errno;
954             (void) fprintf(stderr, MSG_INTL(MSG_ERR_OPEN),
955                             file, strerror(err));
956             continue;
957         }
958         (void) elf_version(EV_CURRENT);
959         if ((elf = elf_begin(fd, ELF_C_READ, NULL)) == NULL) {
960             failure(file, MSG_ORIG(MSG_ELF_BEGIN));
961             (void) close(fd);
962             continue;
963         }
964
965         if (var > 1)
966             dbg_print(0, MSG_ORIG(MSG_FMT_NLSTRNL), file);
967
968         switch (elf_kind(elf)) {
969             case ELF_K_AR:
970                 ret = archive(file, fd, elf, flags, wname, wfd, osabi);
971                 break;
972             case ELF_K_ELF:
973                 ret = decide(file, fd, elf, flags, wname, wfd, osabi);
974                 break;
975             default:
976                 (void) fprintf(stderr, MSG_INTL(MSG_ERR_BADFILE), file);
977                 ret = 1;
978         }
979         #endif /* ! codereview */
980         break;
981     }
982     (void) close(fd);

```

```
983         (void) elf_end(elf);
984     }
985
986     if (wfd)
987         (void) close(wfd);
988     return (ret);
989
990 usage_brief:
991     /* Control comes here for a simple usage message and exit */
992     (void) fprintf(stderr, MSG_INTL(MSG_USAGE_BRIEF),
993                 basename(argv[0]));
994     return (1);
995
996 }
```