```
**********************************************************
   55594 Wed Jan  7 13:34:12 2015
new/usr/src/cmd/sgs/elfdump/common/corenote.c
5510 elfdump doesn't print the member header before pr_reg
**********************************************************
______unchanged_portion_omitted_


 851 /*
 852  * Dump register contents
 853  */
 854 static void
 855 dump_prgregset(note_state_t *state, const char *title)
 856 {
 857         sl_field_t      fdesc1, fdesc2;
 858         sl_fmtbuf_t     buf1, buf2;
 859         Conv_inv_buf_t  inv_buf1, inv_buf2;
 860         Word            w;

 862         fdesc1 = fdesc2 = state->ns_arch->prgregset->elt0;
 863 #endif /* ! codereview */
 864         indent_enter(state, title, &fdesc1);

 862         fdesc1 = fdesc2 = state->ns_arch->prgregset->elt0;
 866         for (w = 0; w < fdesc1.slf_nelts; ) {
 867                 if (w == (fdesc1.slf_nelts - 1)) {
 868                         /* One last register is left */
 869                         if (!data_present(state, &fdesc1))
 870                                 break;
 871                         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE),
 872                             INDENT, state->ns_vcol - state->ns_indent,
 873                             conv_cnote_pr_regname(state->ns_mach, w,
 874                             CONV_FMT_DECIMAL, &inv_buf1),
 875                             fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1));
 876                         fdesc1.slf_offset += fdesc1.slf_eltlen;
 877                         w++;
 878                         continue;
 879                 }

 881                 /* There are at least 2 more registers left. Show 2 up */
 882                 fdesc2.slf_offset = fdesc1.slf_offset + fdesc1.slf_eltlen;
 883                 if (!(data_present(state, &fdesc1) &&
 884                     data_present(state, &fdesc2)))
 885                         break;
 886                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
 887                     state->ns_vcol - state->ns_indent,
 888                     conv_cnote_pr_regname(state->ns_mach, w,
 889                     CONV_FMT_DECIMAL, &inv_buf1),
 890                     state->ns_t2col - state->ns_vcol,
 891                     fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1),
 892                     state->ns_v2col - state->ns_t2col,
 893                     conv_cnote_pr_regname(state->ns_mach, w + 1,
 894                     CONV_FMT_DECIMAL, &inv_buf2),
 895                     fmt_num(state, &fdesc2, SL_FMT_NUM_ZHEX, buf2));
 896                 fdesc1.slf_offset += 2 * fdesc1.slf_eltlen;
 897                 w += 2;
 898         }

 900         indent_exit(state);
 901 }

 903 /*
 904  * Output information from lwpstatus_t structure.
 905  */
 906 static void
 907 dump_lwpstatus(note_state_t *state, const char *title)
```



```
 908 {
 909         const sl_lwpstatus_layout_t     *layout = state->ns_arch->lwpstatus;
 910         Word            w, w2;
 911         int32_t         i;
 912         union {
 913                 Conv_inv_buf_t                  inv;
 914                 Conv_cnote_pr_flags_buf_t       flags;
 915         } conv_buf;

 917         indent_enter(state, title, &layout->pr_flags);

 919         if (data_present(state, &layout->pr_flags)) {
 920                 w = extract_as_word(state, &layout->pr_flags);
 921                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
 922                     conv_cnote_pr_flags(w, 0, &conv_buf.flags));
 923         }

 925         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_LWPID), pr_lwpid);

 927         if (data_present(state, &layout->pr_why)) {
 928                 w = extract_as_word(state, &layout->pr_why);
 929                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHY),
 930                     conv_cnote_pr_why(w, 0, &conv_buf.inv));
 928         }

 932                 if (data_present(state, &layout->pr_what)) {
 933                         w2 = extract_as_word(state, &layout->pr_what);
 934                         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHAT),
 935                             conv_cnote_pr_what(w, w2, 0, &conv_buf.inv));
 936                 }
 937         }
 938 #endif /* ! codereview */

 940         if (data_present(state, &layout->pr_cursig)) {
 941                 w = extract_as_word(state, &layout->pr_cursig);
 942                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_CURSIG),
 943                     conv_cnote_signal(w, CONV_FMT_DECIMAL, &conv_buf.inv));
 944         }

 946         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_INFO), pr_info, dump_siginfo);
 947         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPPEND), pr_lwppend,
 948             dump_sigset);
 949         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPHOLD), pr_lwphold,
 950             dump_sigset);
 951         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ACTION), pr_action,
 952             dump_sigaction);
 953         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ALTSTACK), pr_altstack,
 954             dump_stack);

 956         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_OLDCONTEXT), pr_oldcontext);

 958         if (data_present(state, &layout->pr_syscall)) {
 959                 w = extract_as_word(state, &layout->pr_syscall);
 960                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
 961                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
 962         }

 964         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSYSARG), pr_nsysarg);

 966         if (data_present(state, &layout->pr_errno)) {
 967                 w = extract_as_word(state, &layout->pr_errno);
 968                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_ERRNO),
 969                     conv_cnote_errno(w, CONV_FMT_DECIMAL, &conv_buf.inv));
 970         }

 972         if (data_present(state, &layout->pr_nsysarg)) {
```

```
973                 w2 = extract_as_word(state, &layout->pr_nsysarg);
974                 print_array(state, &layout->pr_sysarg, SL_FMT_NUM_ZHEX, w2, 1,
975                     MSG_ORIG(MSG_CNOTE_T_PR_SYSARG));
976         }

978         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RVAL1), pr_rval1,
979             MSG_ORIG(MSG_CNOTE_T_PR_RVAL2), pr_rval2);
980         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
981         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TSTAMP), pr_tstamp,
982             dump_timestruc);
983         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
984         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);

986         if (data_present(state, &layout->pr_errpriv)) {
987                 i = extract_as_sword(state, &layout->pr_errpriv);
988                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_ERRPRIV),
989                     conv_cnote_priv(i, CONV_FMT_DECIMAL, &conv_buf.inv));
990         }

992         PRINT_ZHEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_USTACK), pr_ustack,
993             MSG_ORIG(MSG_CNOTE_T_PR_INSTR), pr_instr);

995         /*
996          * In order to line up all the values in a single column,
997          * we would have to set vcol to a very high value, which results
998          * in ugly looking output that runs off column 80. So, we use
999          * two levels of vcol, one for the contents so far, and a
1000         * higher one for the pr_reg sub-struct.
1001         */
1002        state->ns_vcol += 3;
1003        state->ns_t2col += 3;
1004        state->ns_v2col += 2;
1005        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_REG), pr_reg, dump_prgregset);
1006        state->ns_vcol -= 3;
1007        state->ns_t2col -= 3;
1008        state->ns_v2col -= 2;

1010        /*
1011         * The floating point register state is complex, and highly
1012         * platform dependent. For now, we simply display it as
1013         * a hex dump. This can be replaced if better information
1014         * is required.
1015         */
1016        if (data_present(state, &layout->pr_fpreg)) {
1017                indent_enter(state, MSG_ORIG(MSG_CNOTE_T_PR_FPREG),
1018                    &layout->pr_fpreg);
1019                dump_hex_bytes(layout->pr_fpreg.slf_offset + state->ns_data,
1020                    layout->pr_fpreg.slf_eltlen, state->ns_indent, 4, 3);
1021                indent_exit(state);
1022        }

1024        indent_exit(state);
1025 }


1028 /*
1029  * Output information from pstatus_t structure.
1030  */
1031 static void
1032 dump_pstatus(note_state_t *state, const char *title)
1033 {
1034        const sl_pstatus_layout_t       *layout = state->ns_arch->pstatus;
1035        Word                            w;
1036        union {
1037                Conv_inv_buf_t                  inv;
1038                Conv_cnote_pr_flags_buf_t       flags;
```



```
1039        } conv_buf;

1041        indent_enter(state, title, &layout->pr_flags);

1043        if (data_present(state, &layout->pr_flags)) {
1044                w = extract_as_word(state, &layout->pr_flags);
1045                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
1046                    conv_cnote_pr_flags(w, 0, &conv_buf.flags));
1047        }

1049        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1050        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1051            MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1052        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGID), pr_pgid,
1053            MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1054        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ASLWPID), pr_aslwpid,
1055            MSG_ORIG(MSG_CNOTE_T_PR_AGENTID), pr_agentid);
1056        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGPEND), pr_sigpend,
1057            dump_sigset);
1058        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_BRKBASE),
1059            &layout->pr_brkbase, SL_FMT_NUM_ZHEX,
1060            MSG_ORIG(MSG_CNOTE_T_PR_BRKSIZE),
1061            &layout->pr_brksize, SL_FMT_NUM_HEX);
1062        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_STKBASE),
1063            &layout->pr_stkbase, SL_FMT_NUM_ZHEX,
1064            MSG_ORIG(MSG_CNOTE_T_PR_STKSIZE),
1065            &layout->pr_stksize, SL_FMT_NUM_HEX);
1066        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
1067        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);
1068        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CUTIME), pr_cutime,
1069            dump_timestruc);
1070        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CSTIME), pr_cstime,
1071            dump_timestruc);
1072        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGTRACE), pr_sigtrace,
1073            dump_sigset);
1074        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_FLTTRACE), pr_flttrace,
1075            dump_fltset);
1076        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SYSENTRY), pr_sysentry,
1077            dump_sysset);
1078        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SYSEXIT), pr_sysexit,
1079            dump_sysset);

1081        if (data_present(state, &layout->pr_dmodel)) {
1082                w = extract_as_word(state, &layout->pr_dmodel);
1083                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1084                    conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1085        }

1087        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_TASKID), pr_taskid,
1088            MSG_ORIG(MSG_CNOTE_T_PR_PROJID), pr_projid);
1089        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_NZOMB), pr_nzomb,
1090            MSG_ORIG(MSG_CNOTE_T_PR_ZONEID), pr_zoneid);

1092        /*
1093         * In order to line up all the values in a single column,
1094         * we would have to set vcol to a very high value, which results
1095         * in ugly looking output that runs off column 80. So, we use
1096         * two levels of vcol, one for the contents so far, and a
1097         * higher one for the pr_lwp sub-struct.
1098         */
1099        state->ns_vcol += 5;
1100        state->ns_t2col += 5;
1101        state->ns_v2col += 5;
1102        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWP), pr_lwp, dump_lwpstatus);
1103        state->ns_vcol -= 5;
1104        state->ns_t2col -= 5;
```

```
1105         state->ns_v2col -= 5;

1107         indent_exit(state);
1108 }


1111 /*
1112  * Output information from prstatus_t (<sys/old_procfs.h>) structure.
1113  */
1114 static void
1115 dump_prstatus(note_state_t *state, const char *title)
1116 {
1117         const sl_prstatus_layout_t      *layout = state->ns_arch->prstatus;
1118         Word                            w, w2;
1119         int                             i;
1120         union {
1121                 Conv_inv_buf_t                  inv;
1122                 Conv_cnote_old_pr_flags_buf_t   flags;
1123         } conv_buf;

1125         indent_enter(state, title, &layout->pr_flags);

1127         if (data_present(state, &layout->pr_flags)) {
1128                 w = extract_as_word(state, &layout->pr_flags);
1129                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAGS),
1130                     conv_cnote_old_pr_flags(w, 0, &conv_buf.flags));
1131         }

1133         if (data_present(state, &layout->pr_why)) {
1134                 w = extract_as_word(state, &layout->pr_why);
1135                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHY),
1136                     conv_cnote_pr_why(w, 0, &conv_buf.inv));

 935         }

1139                 if (data_present(state, &layout->pr_what)) {
1140                         w2 = extract_as_word(state, &layout->pr_what);
1141                         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_WHAT),
1142                             conv_cnote_pr_what(w, w2, 0, &conv_buf.inv));
1143                 }
1144         }
1145 #endif /* ! codereview */

1147         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_INFO), pr_info, dump_siginfo);

1149         if (data_present(state, &layout->pr_cursig)) {
1150                 w = extract_as_word(state, &layout->pr_cursig);
1151                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_CURSIG),
1152                     conv_cnote_signal(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1153         }

1155         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1156         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGPEND), pr_sigpend,
1157             dump_sigset);
1158         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_SIGHOLD), pr_sighold,
1159             dump_sigset);
1160         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ALTSTACK), pr_altstack,
1161             dump_stack);
1162         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_ACTION), pr_action,
1163             dump_sigaction);
1164         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1165             MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1166         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGRP), pr_pgrp,
1167             MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1168         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_UTIME), pr_utime, dump_timestruc);
1169         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_STIME), pr_stime, dump_timestruc);
```



```
1170         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CUTIME), pr_cutime,
1171             dump_timestruc);
1172         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CSTIME), pr_cstime,
1173             dump_timestruc);
1174         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);

1176         if (data_present(state, &layout->pr_syscall)) {
1177                 w = extract_as_word(state, &layout->pr_syscall);
1178                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1179                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1180         }

1182         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSYSARG), pr_nsysarg);

1184         if (data_present(state, &layout->pr_nsysarg)) {
1185                 w2 = extract_as_word(state, &layout->pr_nsysarg);
1186                 print_array(state, &layout->pr_sysarg, SL_FMT_NUM_ZHEX, w2, 1,
1187                     MSG_ORIG(MSG_CNOTE_T_PR_SYSARG));
1188         }

1190         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_WHO), pr_who);
1191         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWPPEND), pr_sigpend,
1192             dump_sigset);
1193         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_OLDCONTEXT), pr_oldcontext);
1194         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_BRKBASE),
1195             &layout->pr_brkbase, SL_FMT_NUM_ZHEX,
1196             MSG_ORIG(MSG_CNOTE_T_PR_BRKSIZE),
1197             &layout->pr_brksize, SL_FMT_NUM_HEX);
1198         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_STKBASE),
1199             &layout->pr_stkbase, SL_FMT_NUM_ZHEX,
1200             MSG_ORIG(MSG_CNOTE_T_PR_STKSIZE),
1201             &layout->pr_stksize, SL_FMT_NUM_HEX);
1202         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_PROCESSOR), pr_processor);

1204         if (data_present(state, &layout->pr_bind)) {
1205                 i = extract_as_sword(state, &layout->pr_bind);
1206                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_BIND),
1207                     conv_cnote_psetid(i, CONV_FMT_DECIMAL, &conv_buf.inv));
1208         }

1210         PRINT_ZHEX(MSG_ORIG(MSG_CNOTE_T_PR_INSTR), pr_instr);
1211         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_REG), pr_reg, dump_prgregset);

1213         indent_exit(state);
1214 }


1217 /*
1218  * Print percent from 16-bit binary fraction [0 .. 1]
1219  * Round up .01 to .1 to indicate some small percentage (the 0x7000 below).
1220  *
1221  * Note: This routine was copied from ps(1) and then modified.
1222  */
1223 static const char *
1224 prtpct_value(note_state_t *state, const sl_field_t *fdesc,
1225     sl_fmtbuf_t buf)
1226 {
1227         uint_t value;           /* need 32 bits to compute with */

1229         value = extract_as_word(state, fdesc);
1230         value = ((value * 1000) + 0x7000) >> 15;        /* [0 .. 1000] */
1231         if (value >= 1000)
1232                 value = 999;

1234         (void) snprintf(buf, sizeof (sl_fmtbuf_t),
1235             MSG_ORIG(MSG_CNOTE_FMT_PRTPCT), value / 10, value % 10);
```

```
1237         return (buf);
1238 }



1242 /*
1243  * Version of prtpct() used for a 2-up display of two adjacent percentages.
1244  */
1245 static void
1246 prtpct_2up(note_state_t *state, const sl_field_t *fdesc1,
1247     const char *title1, const sl_field_t *fdesc2, const char *title2)
1248 {
1249         sl_fmtbuf_t     buf1, buf2;

1251         if (!(data_present(state, fdesc1) &&
1252             data_present(state, fdesc2)))
1253                 return;

1255         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1256             state->ns_vcol - state->ns_indent, title1,
1257             state->ns_t2col - state->ns_vcol,
1258             prtpct_value(state, fdesc1, buf1),
1259             state->ns_v2col - state->ns_t2col, title2,
1260             prtpct_value(state, fdesc2, buf2));
1261 }


1264 /*
1265  * The psinfo_t and prpsinfo_t structs have pr_state and pr_sname
1266  * fields that we wish to print in a 2up format. The pr_state is
1267  * an integer, while pr_sname is a single character.
1268  */
1269 static void
1270 print_state_sname_2up(note_state_t *state,
1271     const sl_field_t *state_fdesc,
1272     const sl_field_t *sname_fdesc)
1273 {
1274         sl_fmtbuf_t     buf1, buf2;
1275         int             sname;

1277         /*
1278          * If the field slf_offset and extent fall past the end of the
1279          * available data, then return without doing anything. That note
1280          * is from an older core file that doesn't have all the fields
1281          * that we know about.
1282          */
1283         if (!(data_present(state, state_fdesc) &&
1284             data_present(state, sname_fdesc)))
1285                 return;

1287         sname = extract_as_sword(state, sname_fdesc);
1288         buf2[0] = sname;
1289         buf2[1] = '\0';

1291         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1292             state->ns_vcol - state->ns_indent, MSG_ORIG(MSG_CNOTE_T_PR_STATE),
1293             state->ns_t2col - state->ns_vcol,
1294             fmt_num(state, state_fdesc, SL_FMT_NUM_DEC, buf1),
1295             state->ns_v2col - state->ns_t2col, MSG_ORIG(MSG_CNOTE_T_PR_SNAME),
1296             buf2);
1297 }

1299 /*
1300  * Output information from lwpsinfo_t structure.
1301  */
```



```
1302 static void
1303 dump_lwpsinfo(note_state_t *state, const char *title)
1304 {
1305         const sl_lwpsinfo_layout_t      *layout = state->ns_arch->lwpsinfo;
1306         Word                    w;
1307         int32_t                 i;
1308         union {
1309                 Conv_cnote_proc_flag_buf_t      proc_flag;
1310                 Conv_inv_buf_t                  inv;
1311         } conv_buf;

1313         indent_enter(state, title, &layout->pr_flag);

1315         if (data_present(state, &layout->pr_flag)) {
1316                 w = extract_as_word(state, &layout->pr_flag);
1317                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1318                     conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1319         }

1321         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_LWPID), &layout->pr_lwpid,
1322             SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1323             SL_FMT_NUM_ZHEX);
1324         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_WCHAN), pr_wchan);

1326         if (data_present(state, &layout->pr_stype)) {
1327                 w = extract_as_word(state, &layout->pr_stype);
1328                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_STYPE),
1329                     conv_cnote_pr_stype(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1330         }

1332         print_state_sname_2up(state, &layout->pr_state, &layout->pr_sname);

1334         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NICE), pr_nice);

1336         if (data_present(state, &layout->pr_syscall)) {
1337                 w = extract_as_word(state, &layout->pr_syscall);
1338                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1339                     conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1340         }

1342         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_OLDPRI), pr_oldpri,
1343             MSG_ORIG(MSG_CNOTE_T_PR_CPU), pr_cpu);

1345         if (data_present(state, &layout->pr_pri) &&
1346             data_present(state, &layout->pr_pctcpu)) {
1347                 sl_fmtbuf_t     buf1, buf2;

1349                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1350                     state->ns_vcol - state->ns_indent,
1351                     MSG_ORIG(MSG_CNOTE_T_PR_PRI),
1352                     state->ns_t2col - state->ns_vcol,
1353                     fmt_num(state, &layout->pr_pri, SL_FMT_NUM_DEC, buf1),
1354                     state->ns_v2col - state->ns_t2col,
1355                     MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1356                     prtpct_value(state, &layout->pr_pctcpu, buf2));
1357         }

1359         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1360         PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1361         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
1362         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_NAME), pr_name);
1363         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ONPRO), pr_onpro,
1364             MSG_ORIG(MSG_CNOTE_T_PR_BINDPRO), pr_bindpro);

1366         if (data_present(state, &layout->pr_bindpset)) {
1367                 i = extract_as_sword(state, &layout->pr_bindpset);
```

```
1368                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_BINDPSET),
1369                    conv_cnote_psetid(i, CONV_FMT_DECIMAL, &conv_buf.inv));
1370        }

1372        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_LGRP), pr_lgrp);

1374        indent_exit(state);
1375 }


1378 /*
1379  * Output information from psinfo_t structure.
1380  */
1381 static void
1382 dump_psinfo(note_state_t *state, const char *title)
1383 {
1384        const sl_psinfo_layout_t        *layout = state->ns_arch->psinfo;
1385        Word                            w;
1386        union {
1387                Conv_cnote_proc_flag_buf_t      proc_flag;
1388                Conv_inv_buf_t                  inv;
1389        } conv_buf;

1391        indent_enter(state, title, &layout->pr_flag);

1393        if (data_present(state, &layout->pr_flag)) {
1394                w = extract_as_word(state, &layout->pr_flag);
1395                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1396                    conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1397        }

1399        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NLWP), pr_nlwp);
1400        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1401            MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1402        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGID), pr_pgid,
1403            MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1404        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1405            MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid);
1406        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid,
1407            MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1408        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1409            SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_SIZE), &layout->pr_size,
1410            SL_FMT_NUM_HEX);
1411        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_RSSIZE),
1412            &layout->pr_rssize, SL_FMT_NUM_HEX, MSG_ORIG(MSG_CNOTE_T_PR_TTYDEV),
1413            &layout->pr_ttydev, SL_FMT_NUM_DEC);
1414        prtpct_2up(state, &layout->pr_pctcpu, MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1415            &layout->pr_pctmem, MSG_ORIG(MSG_CNOTE_T_PR_PCTMEM));
1416        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1417        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1418        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CTIME), pr_ctime, dump_timestruc);
1419        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_FNAME), pr_fname);
1420        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PSARGS), pr_psargs);
1421        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_WSTAT), &layout->pr_wstat,
1422            SL_FMT_NUM_HEX, MSG_ORIG(MSG_CNOTE_T_PR_ARGC), &layout->pr_argc,
1423            SL_FMT_NUM_DEC);
1424        PRINT_ZHEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ARGV), pr_argv,
1425            MSG_ORIG(MSG_CNOTE_T_PR_ENVP), pr_envp);

1427        if (data_present(state, &layout->pr_dmodel)) {
1428                w = extract_as_word(state, &layout->pr_dmodel);
1429                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1430                    conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1431        }

1433        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_TASKID), pr_taskid,
```



```
1434            MSG_ORIG(MSG_CNOTE_T_PR_PROJID), pr_projid);
1435        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_NZOMB), pr_nzomb,
1436            MSG_ORIG(MSG_CNOTE_T_PR_POOLID), pr_poolid);
1437        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ZONEID), pr_zoneid,
1438            MSG_ORIG(MSG_CNOTE_T_PR_CONTRACT), pr_contract);

1440        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_LWP), pr_lwp, dump_lwpsinfo);

1442        indent_exit(state);
1443 }

1445 /*
1446  * Output information from prpsinfo_t structure.
1447  */
1448 static void
1449 dump_prpsinfo(note_state_t *state, const char *title)
1450 {
1451        const sl_prpsinfo_layout_t      *layout = state->ns_arch->prpsinfo;
1452        Word                            w;
1453        union {
1454                Conv_cnote_proc_flag_buf_t      proc_flag;
1455                Conv_inv_buf_t                  inv;
1456        } conv_buf;

1458        indent_enter(state, title, &layout->pr_state);

1460        print_state_sname_2up(state, &layout->pr_state, &layout->pr_sname);
1461        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_ZOMB), pr_zomb,
1462            MSG_ORIG(MSG_CNOTE_T_PR_NICE), pr_nice);

1464        if (data_present(state, &layout->pr_flag)) {
1465                w = extract_as_word(state, &layout->pr_flag);
1466                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FLAG),
1467                    conv_cnote_proc_flag(w, 0, &conv_buf.proc_flag));
1468        }


1471        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1472            MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid);
1473        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PID), pr_pid,
1474            MSG_ORIG(MSG_CNOTE_T_PR_PPID), pr_ppid);
1475        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PGRP), pr_pgrp,
1476            MSG_ORIG(MSG_CNOTE_T_PR_SID), pr_sid);
1477        print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ADDR), &layout->pr_addr,
1478            SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_SIZE), &layout->pr_size,
1479            SL_FMT_NUM_HEX);
1480        PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RSSIZE), pr_rssize,
1481            MSG_ORIG(MSG_CNOTE_T_PR_WCHAN), pr_wchan);
1482        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_START), pr_start, dump_timestruc);
1483        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_TIME), pr_time, dump_timestruc);
1484        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_PRI), pr_pri,
1485            MSG_ORIG(MSG_CNOTE_T_PR_OLDPRI), pr_oldpri);
1486        PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_CPU), pr_cpu);
1487        PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_OTTYDEV), pr_ottydev,
1488            MSG_ORIG(MSG_CNOTE_T_PR_LTTYDEV), pr_lttydev);
1489        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_CLNAME), pr_clname);
1490        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_FNAME), pr_fname);
1491        PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PSARGS), pr_psargs);

1493        if (data_present(state, &layout->pr_syscall)) {
1494                w = extract_as_word(state, &layout->pr_syscall);
1495                print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_SYSCALL),
1496                    conv_cnote_syscall(w, CONV_FMT_DECIMAL, &conv_buf.inv));
1497        }

1499        PRINT_SUBTYPE(MSG_ORIG(MSG_CNOTE_T_PR_CTIME), pr_ctime, dump_timestruc);
```

```
1500         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PR_BYSIZE), pr_bysize,
1501             MSG_ORIG(MSG_CNOTE_T_PR_BYRSSIZE), pr_byrssize);
1502         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ARGC), &layout->pr_argc,
1503             SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PR_ARGV), &layout->pr_argv,
1504             SL_FMT_NUM_ZHEX);
1505         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PR_ENVP), &layout->pr_envp,
1506             SL_FMT_NUM_ZHEX, MSG_ORIG(MSG_CNOTE_T_PR_WSTAT), &layout->pr_wstat,
1507             SL_FMT_NUM_HEX);
1508         prtpct_2up(state, &layout->pr_pctcpu, MSG_ORIG(MSG_CNOTE_T_PR_PCTCPU),
1509             &layout->pr_pctmem, MSG_ORIG(MSG_CNOTE_T_PR_PCTMEM));
1510         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid,
1511             MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1512         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_ASLWPID), pr_aslwpid);

1514         if (data_present(state, &layout->pr_dmodel)) {
1515                 w = extract_as_word(state, &layout->pr_dmodel);
1516                 print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_DMODEL),
1517                     conv_cnote_pr_dmodel(w, 0, &conv_buf.inv));
1518         }

1520         indent_exit(state);
1521 }


1524 /*
1525  * Output information from prcred_t structure.
1526  */
1527 static void
1528 dump_prcred(note_state_t *state, const char *title)
1529 {
1530         const sl_prcred_layout_t *layout = state->ns_arch->prcred;
1531         Word            ngroups;

1533         indent_enter(state, title, &layout->pr_euid);

1535         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_EUID), pr_euid,
1536             MSG_ORIG(MSG_CNOTE_T_PR_RUID), pr_ruid);
1537         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_SUID), pr_suid,
1538             MSG_ORIG(MSG_CNOTE_T_PR_EGID), pr_egid);
1539         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RGID), pr_rgid,
1540             MSG_ORIG(MSG_CNOTE_T_PR_SGID), pr_sgid);
1541         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NGROUPS), pr_ngroups);

1543         if (data_present(state, &layout->pr_ngroups)) {
1544                 ngroups = extract_as_word(state, &layout->pr_ngroups);
1545                 print_array(state, &layout->pr_groups, SL_FMT_NUM_DEC, ngroups,
1546                     0, MSG_ORIG(MSG_CNOTE_T_PR_GROUPS));
1547         }

1549         indent_exit(state);
1550 }


1553 /*
1554  * Output information from prpriv_t structure.
1555  */
1556 static void
1557 dump_prpriv(note_state_t *state, const char *title)
1558 {
1559         const sl_prpriv_layout_t *layout = state->ns_arch->prpriv;
1560         Word            nsets;

1562         indent_enter(state, title, &layout->pr_nsets);

1564         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_NSETS), pr_nsets);
1565         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_SETSIZE), pr_setsize);
```



```
1566         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PR_INFOSIZE), pr_infosize);

1568         if (data_present(state, &layout->pr_nsets)) {
1569                 nsets = extract_as_word(state, &layout->pr_nsets);
1570                 print_array(state, &layout->pr_sets, SL_FMT_NUM_ZHEX, nsets,
1571                     0, MSG_ORIG(MSG_CNOTE_T_PR_SETS));
1572         }

1574         indent_exit(state);
1575 }

1577 static void
1578 dump_prfdinfo(note_state_t *state, const char *title)
1579 {
1580         const sl_prfdinfo_layout_t *layout = state->ns_arch->prfdinfo;
1581         char buf[1024];
1582         uint32_t fileflags, mode;

1584         indent_enter(state, title, &layout->pr_fd);

1586         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_FD), pr_fd);
1587         mode = extract_as_word(state, &layout->pr_mode);

1589         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_MODE),
1590             conv_cnote_filemode(mode, 0, buf, sizeof (buf)));

1592         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_UID), pr_uid,
1593             MSG_ORIG(MSG_CNOTE_T_PR_GID), pr_gid);

1595         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_MAJOR), pr_major,
1596             MSG_ORIG(MSG_CNOTE_T_PR_MINOR), pr_minor);
1597         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_RMAJOR), pr_rmajor,
1598             MSG_ORIG(MSG_CNOTE_T_PR_RMINOR), pr_rminor);

1600         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_INO), pr_ino);

1602         PRINT_DEC_2UP(MSG_ORIG(MSG_CNOTE_T_PR_SIZE), pr_size,
1603             MSG_ORIG(MSG_CNOTE_T_PR_OFFSET), pr_offset);

1605         fileflags = extract_as_word(state, &layout->pr_fileflags);

1607         print_str(state, MSG_ORIG(MSG_CNOTE_T_PR_FILEFLAGS),
1608             conv_cnote_fileflags(fileflags, 0, buf, sizeof (buf)));

1610         PRINT_DEC(MSG_ORIG(MSG_CNOTE_T_PR_FDFLAGS), pr_fdflags);

1612         PRINT_STRBUF(MSG_ORIG(MSG_CNOTE_T_PR_PATH), pr_path);

1614         indent_exit(state);
1615 }

1617 /*
1618  * Output information from priv_impl_info_t structure.
1619  */
1620 static void
1621 dump_priv_impl_info(note_state_t *state, const char *title)
1622 {
1623         const sl_priv_impl_info_layout_t *layout;

1625         layout = state->ns_arch->priv_impl_info;
1626         indent_enter(state, title, &layout->priv_headersize);

1628         PRINT_HEX_2UP(MSG_ORIG(MSG_CNOTE_T_PRIV_HEADERSIZE), priv_headersize,
1629             MSG_ORIG(MSG_CNOTE_T_PRIV_FLAGS), priv_flags);

1631         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PRIV_NSETS),
```

```
1632             &layout->priv_nsets, SL_FMT_NUM_DEC,
1633             MSG_ORIG(MSG_CNOTE_T_PRIV_SETSIZE), &layout->priv_setsize,
1634             SL_FMT_NUM_HEX);
1635         print_num_2up(state, MSG_ORIG(MSG_CNOTE_T_PRIV_MAX), &layout->priv_max,
1636             SL_FMT_NUM_DEC, MSG_ORIG(MSG_CNOTE_T_PRIV_INFOSIZE),
1637             &layout->priv_infosize, SL_FMT_NUM_HEX);
1638         PRINT_HEX(MSG_ORIG(MSG_CNOTE_T_PRIV_GLOBALINFOSIZE),
1639             priv_globalinfosize);

1641         indent_exit(state);
1642 }


1645 /*
1646  * Dump information from an asrset_t array. This data
1647  * structure is specific to sparcv9, and does not appear
1648  * on any other platform.
1649  *
1650  * asrset_t is a simple array, defined in <sys/regset.h> as
1651  *      typedef int64_t asrset_t[16];    %asr16 - > %asr31
1652  *
1653  * As such, we do not make use of the struct_layout facilities
1654  * for this routine.
1655  */
1656 static void
1657 dump_asrset(note_state_t *state, const char *title)
1658 {
1659         static const sl_field_t ftemplate = { 0, sizeof (int64_t), 16, 0 };
1660         sl_field_t      fdesc1, fdesc2;
1661         sl_fmtbuf_t     buf1, buf2;
1662         char            index1[MAXNDXSIZE * 2], index2[MAXNDXSIZE * 2];
1663         Word            w, nelts;

1665         fdesc1 = fdesc2 =  ftemplate;

1667         /* We expect 16 values, but will print whatever is actually there */
1668         nelts = state->ns_len / ftemplate.slf_eltlen;
1669         if (nelts == 0)
1670                 return;

1672         indent_enter(state, title, &fdesc1);

1674         for (w = 0; w < nelts; ) {
1675                 (void) snprintf(index1, sizeof (index1),
1676                     MSG_ORIG(MSG_FMT_ASRINDEX), w + 16);

1678                 if (w == (nelts - 1)) {
1679                         /* One last register is left */
1680                         dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE),
1681                             INDENT, state->ns_vcol - state->ns_indent, index1,
1682                             fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1));
1683                         fdesc1.slf_offset += fdesc1.slf_eltlen;
1684                         w++;
1685                         continue;
1686                 }

1688                 /* There are at least 2 more registers left. Show 2 up */
1689                 (void) snprintf(index2, sizeof (index2),
1690                     MSG_ORIG(MSG_FMT_ASRINDEX), w + 17);

1692                 fdesc2.slf_offset = fdesc1.slf_offset + fdesc1.slf_eltlen;
1693                 dbg_print(0, MSG_ORIG(MSG_CNOTE_FMT_LINE_2UP), INDENT,
1694                     state->ns_vcol - state->ns_indent, index1,
1695                     state->ns_t2col - state->ns_vcol,
1696                     fmt_num(state, &fdesc1, SL_FMT_NUM_ZHEX, buf1),
1697                     state->ns_v2col - state->ns_t2col, index2,
```



```
1698                     fmt_num(state, &fdesc2, SL_FMT_NUM_ZHEX, buf2));
1699                 fdesc1.slf_offset += 2 * fdesc1.slf_eltlen;
1700                 w += 2;
1701         }

1703         indent_exit(state);
1704 }

1706 corenote_ret_t
1707 corenote(Half mach, int do_swap, Word type,
1708     const char *desc, Word descsz)
1709 {
1710         note_state_t            state;

1712         /*
1713          * Get the per-architecture layout definition
1714          */
1715         state.ns_mach = mach;
1716         state.ns_arch = sl_mach(state.ns_mach);
1717         if (sl_mach(state.ns_mach) == NULL)
1718                 return (CORENOTE_R_BADARCH);

1720         state.ns_swap = do_swap;
1721         state.ns_indent = 4;
1722         state.ns_t2col = state.ns_v2col = 0;
1723         state.ns_data = desc;
1724         state.ns_len = descsz;

1726         switch (type) {
1727         case NT_PRSTATUS:               /* prstatus_t <sys/old_procfs.h> */
1728                 state.ns_vcol = 26;
1729                 state.ns_t2col = 46;
1730                 state.ns_v2col = 60;
1731                 dump_prstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_PRSTATUS_T));
1732                 return (CORENOTE_R_OK);

1734         case NT_PRFPREG:                /* prfpregset_t <sys/procfs_isa.h> */
1735                 return (CORENOTE_R_OK_DUMP);

1737         case NT_PRPSINFO:               /* prpsinfo_t   <sys/old_procfs.h> */
1738                 state.ns_vcol = 20;
1739                 state.ns_t2col = 41;
1740                 state.ns_v2col = 54;
1741                 dump_prpsinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PRPSINFO_T));
1742                 return (CORENOTE_R_OK);

1744         case NT_PRXREG:                 /* prxregset_t <sys/procfs_isa.h> */
1745                 return (CORENOTE_R_OK_DUMP);

1747         case NT_PLATFORM:               /* string from sysinfo(SI_PLATFORM) */
1748                 dbg_print(0, MSG_ORIG(MSG_NOTE_DESC));
1749                 dbg_print(0, MSG_ORIG(MSG_FMT_INDENT), safe_str(desc, descsz));
1750                 return (CORENOTE_R_OK);

1752         case NT_AUXV:                   /* auxv_t array <sys/auxv.h> */
1753                 state.ns_vcol = 18;
1754                 dump_auxv(&state, MSG_ORIG(MSG_CNOTE_DESC_AUXV_T));
1755                 return (CORENOTE_R_OK);

1757         case NT_GWINDOWS:               /* gwindows_t SPARC only */
1758                 return (CORENOTE_R_OK_DUMP);

1760         case NT_ASRS:                   /* asrset_t <sys/regset> sparcv9 only */
1761                 state.ns_vcol = 18;
1762                 state.ns_t2col = 38;
1763                 state.ns_v2col = 46;
```

```
1764                dump_asrset(&state, MSG_ORIG(MSG_CNOTE_DESC_ASRSET_T));
1765                return (CORENOTE_R_OK);

1767        case NT_LDT:                    /* ssd array <sys/sysi86.h> IA32 only */
1768                return (CORENOTE_R_OK_DUMP);

1770        case NT_PSTATUS:                /* pstatus_t <sys/procfs.h> */
1771                state.ns_vcol = 22;
1772                state.ns_t2col = 42;
1773                state.ns_v2col = 54;
1774                dump_pstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_PSTATUS_T));
1775                return (CORENOTE_R_OK);

1777        case NT_PSINFO:                 /* psinfo_t <sys/procfs.h> */
1778                state.ns_vcol = 25;
1779                state.ns_t2col = 45;
1780                state.ns_v2col = 58;
1781                dump_psinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PSINFO_T));
1782                return (CORENOTE_R_OK);

1784        case NT_PRCRED:                 /* prcred_t <sys/procfs.h> */
1785                state.ns_vcol = 20;
1786                state.ns_t2col = 34;
1787                state.ns_v2col = 44;
1788                dump_prcred(&state, MSG_ORIG(MSG_CNOTE_DESC_PRCRED_T));
1789                return (CORENOTE_R_OK);

1791        case NT_UTSNAME:                /* struct utsname <sys/utsname.h> */
1792                state.ns_vcol = 18;
1793                dump_utsname(&state, MSG_ORIG(MSG_CNOTE_DESC_STRUCT_UTSNAME));
1794                return (CORENOTE_R_OK);

1796        case NT_LWPSTATUS:              /* lwpstatus_t <sys/procfs.h> */
1797                state.ns_vcol = 24;
1798                state.ns_t2col = 44;
1799                state.ns_v2col = 54;
1800                dump_lwpstatus(&state, MSG_ORIG(MSG_CNOTE_DESC_LWPSTATUS_T));
1801                return (CORENOTE_R_OK);

1803        case NT_LWPSINFO:               /* lwpsinfo_t <sys/procfs.h> */
1804                state.ns_vcol = 22;
1805                state.ns_t2col = 42;
1806                state.ns_v2col = 54;
1807                dump_lwpsinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_LWPSINFO_T));
1808                return (CORENOTE_R_OK);

1810        case NT_PRPRIV:                 /* prpriv_t <sys/procfs.h> */
1811                state.ns_vcol = 21;
1812                state.ns_t2col = 34;
1813                state.ns_v2col = 38;
1814                dump_prpriv(&state, MSG_ORIG(MSG_CNOTE_DESC_PRPRIV_T));
1815                return (CORENOTE_R_OK);

1817        case NT_PRPRIVINFO:             /* priv_impl_info_t <sys/priv.h> */
1818                state.ns_vcol = 29;
1819                state.ns_t2col = 41;
1820                state.ns_v2col = 56;
1821                dump_priv_impl_info(&state,
1822                    MSG_ORIG(MSG_CNOTE_DESC_PRIV_IMPL_INFO_T));
1823                return (CORENOTE_R_OK);

1825        case NT_CONTENT:                /* core_content_t <sys/corectl.h> */
1826                if (sizeof (core_content_t) > descsz)
1827                        return (CORENOTE_R_BADDATA);
1828                {
1829                        static sl_field_t fdesc = { 0, 8, 0, 0 };
```



```
1830                        Conv_cnote_cc_content_buf_t conv_buf;
1831                        core_content_t content;

1833                        state.ns_vcol = 8;
1834                        indent_enter(&state,
1835                            MSG_ORIG(MSG_CNOTE_DESC_CORE_CONTENT_T),
1836                            &fdesc);
1837                        content = extract_as_lword(&state, &fdesc);
1838                        print_str(&state, MSG_ORIG(MSG_STR_EMPTY),
1839                            conv_cnote_cc_content(content, 0, &conv_buf));
1840                        indent_exit(&state);
1841                }
1842                return (CORENOTE_R_OK);

1844        case NT_ZONENAME:               /* string from getzonenamebyid(3C) */
1845                dbg_print(0, MSG_ORIG(MSG_NOTE_DESC));
1846                dbg_print(0, MSG_ORIG(MSG_FMT_INDENT), safe_str(desc, descsz));
1847                return (CORENOTE_R_OK);


1850        case NT_FDINFO:
1851                state.ns_vcol = 22;
1852                state.ns_t2col = 41;
1853                state.ns_v2col = 54;
1854                dump_prfdinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PRFDINFO_T));
1855                return (CORENOTE_R_OK);

1857        case NT_SPYMASTER:
1858                state.ns_vcol = 25;
1859                state.ns_t2col = 45;
1860                state.ns_v2col = 58;
1861                dump_psinfo(&state, MSG_ORIG(MSG_CNOTE_DESC_PSINFO_T));
1862                return (CORENOTE_R_OK);
1863        }

1865        return (CORENOTE_R_BADTYPE);
1866 }
```