

```
*****
64508 Sun Oct 27 14:16:03 2013
new/usr/src/cmd/sgs/libld/common/args.c
4270 ld(1) argument error reporting is still pretty bad
*****
_____ unchanged_portion_omitted_


966 static int      opttitle = 0;
967 /*
968 * Parsing options pass1 for process_flags().
969 */
970 static uintptr_t
971 parseopt_pass1(ofl_desc *ofl, int argc, char **argv, int *usage)
972 {
973     int      c, ndx = optind;

975     /*
976     * The -32, -64 and -ztarget options are special, in that we validate
977     * them, but otherwise ignore them. libld.so (this code) is called
978     * from the ld front end program. ld has already examined the
979     * arguments to determine the output class and machine type of the
980     * output object, as reflected in the version (32/64) of ld_main()
981     * that was called and the value of the 'mach' argument passed.
982     * By time execution reaches this point, these options have already
983     * been seen and acted on.
984     */
985     while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {

987         switch (c) {
988             case '3':
989                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
990
991                 /*
992                 * -32 is processed by ld to determine the output class.
993                 * Here we sanity check the option incase some other
994                 * -3* option is mistakenly passed to us.
995                 */
996                 if (optarg[0] != '2')
997                     ld_eprintf(ofl, ERR_FATAL,
998                               MSG_INTL(MSG_ARG_ILLEGAL),
999                               MSG_ORIG(MSG_ARG_3), optarg);
1000                continue;

1002             case '6':
1003                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1004
1005                 /*
1006                 * -64 is processed by ld to determine the output class.
1007                 * Here we sanity check the option incase some other
1008                 * -6* option is mistakenly passed to us.
1009                 */
1010                 if (optarg[0] != '4')
1011                     ld_eprintf(ofl, ERR_FATAL,
1012                               MSG_INTL(MSG_ARG_ILLEGAL),
1013                               MSG_ORIG(MSG_ARG_6), optarg);
1014                continue;

1016             case 'a':
1017                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1018                 aflag = TRUE;
1019                 break;

1021             case 'b':
1022                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1023                 bflag = TRUE;
1024
1025         }
1026     }
1027
1028     /*
1029     * This is a hack, and may be undone later.
1030     * The -b option is only used to build the Unix
1031     * kernel and its related kernel-mode modules.
1032     * We do not want those files to get a .SUNW_ldynsym
1033     * section. At least for now, the kernel makes no
1034     * use of .SUNW_ldynsym, and we do not want to use
1035     * the space to hold it. Therefore, we overload
1036     * the use of -b to also imply -znoldynsym.
1037     */
1038     ofl->ofl_flags |= FLG_OF_NOLDYNSYM;
1039     break;
1040
1041     case 'c':
1042         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1043         if (ofl->ofl_config)
1044             ld_eprintf(ofl, ERR_WARNING_NF,
1045                       MSG_INTL(MSG_ARG_MTONCE),
1046                       MSG_ORIG(MSG_ARG_C));
1047         else
1048             ofl->ofl_config = optarg;
1049         break;
1050
1051     case 'C':
1052         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1053         demangle_flag = 1;
1054         break;
1055
1056     case 'd':
1057         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1058         if ((optarg[0] == 'n') && (optarg[1] == '\0')) {
1059             if (dflag != SET_UNKNOWN)
1060                 ld_eprintf(ofl, ERR_WARNING_NF,
1061                           MSG_INTL(MSG_ARG_MTONCE),
1062                           MSG_ORIG(MSG_ARG_D));
1063             else
1064                 dflag = SET_FALSE;
1065         } else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
1066             if (dflag != SET_UNKNOWN)
1067                 ld_eprintf(ofl, ERR_WARNING_NF,
1068                           MSG_INTL(MSG_ARG_MTONCE),
1069                           MSG_ORIG(MSG_ARG_D));
1070             else
1071                 dflag = SET_TRUE;
1072         } else {
1073             ld_eprintf(ofl, ERR_FATAL,
1074                           MSG_INTL(MSG_ARG_ILLEGAL),
1075                           MSG_ORIG(MSG_ARG_D), optarg);
1076         }
1077         break;
1078
1079     case 'e':
1080         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1081         if (ofl->ofl_entry)
1082             ld_eprintf(ofl, ERR_WARNING_NF,
1083                           MSG_INTL(MSG_MARG_MTONCE),
1084                           MSG_INTL(MSG_MARG_ENTRY));
1085         else
1086             ofl->ofl_entry = (void *)optarg;
1087         break;
1088
1089     case 'f':
1090         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1091         if ((ofl->ofl_filtrees &&
1092             (!!(ofl->ofl_flags & FLG_OF_AUX))) {
1093             ld_eprintf(ofl, ERR_FATAL,
1094
1095         }
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
18010
18011
18012
18013
18014
18015
18016
18017
18018
18019
18020
18021
18022
18023
18024
18025
18026
18027
18028
18029
18030
18031
18032
18033
18034
18035
18036
18037
18038
18039
18040
18041
18042
18043
18044
18045
18046
18047
18048
18049
18050
18051
18052
18053
18054
18055
18056
18057
18058
18059
18060
18061
18062
18063
18064
18065
18066
18067
18068
18069
18070
18071
18072
18073
18074
18075
18076
18077
18078
18079
18080
18081
18082
18083
18084
18085
18086
18087
18088
18089
18090
18091
18092
18093
18094
18095
18096
18097
18098
18099
180100
180101
180102
180103
180104
180105
180106
180107
180108
180109
180110
180111
180112
180113
180114
180115
180116
180117
180118
180119
180120
180121
180122
180123
180124
180125
180126
180127
180128
180129
180130
180131
180132
180133
180134
180135
180136
180137
180138
180139
180140
180141
180142
180143
180144
180145
180146
180147
180148
180149
180150
180151
180152
180153
180154
180155
180156
180157
180158
180159
180160
180161
180162
180163
180164
180165
180166
180167
180168
180169
180170
180171
180172
180173
180174
180175
180176
180177
180178
180179
180180
180181
180182
180183
180184
180185
180186
180187
180188
180189
180190
180191
180192
180193
180194
180195
180196
180197
180198
180199
180200
180201
180202
180203
180204
180205
180206
180207
180208
180209
180210
180211
180212
180213
180214
180215
180216
180217
180218
180219
180220
180221
180222
180223
180224
180225
180226
180227
180228
180229
180230
180231
180232
180233
180234
180235
180236
180237
180238
180239
180240
180241
180242
180243
180244
180245
180246
180247
180248
180249
180250
180251
180252
180253
180254
180255
180256
180257
180258
180259
180260
180261
180262
180263
180264
180265
180266
180267
180268
180269
180270
180271
180272
180273
180274
180275
180276
180277
180278
180279
180280
180281
180282
180283
180284
180285
180286
180287
180288
180289
180290
180291
180292
180293
180294
180295
180296
180297
180298
180299
180300
180301
180302
180303
180304
180305
180306
180307
180308
180309
180310
180311
180312
180313
180314
180315
180316
180317
180318
180319
180320
180321
180322
180323
180324
180325
180326
180327
180328
180329
180330
180331
180332
180333
180334
180335
180336
180337
180338
180339
180340
180341
180342
180343
180344
180345
180346
180347
180348
180349
180350
180351
180352
180353
180354
180355
180356
180357
180358
180359
180360
180361
180362
180363
180364
180365
180366
180367
180368
180369
180370
180371
180372
180373
180374
180375
180376
180377
180378
180379
180380
180381
180382
180383
180384
180385
180386
180387
180388
180389
180390
180391
180392
180393
180394
180395
180396
180397
180398
180399
180400
180401
180402
180403
180404
180405
180406
180407
180408
180409
180410
180411
180412
180413
180414
180415
180416
180417
180418
180419
180420
180421
180422
180423
180424
180425
180426
180427
180428
180429
180430
180431
180432
180433
180434
180435
180436
180437
180438
180439
180440
180441
180442
180443
180444
180445
180446
180447
180448
180449
180450
180451
180452
180453
180454
180455
180456
180457
180458
180459
180460
180461
180462
180463
180464
180465
180466
180467
180468
180469
180470
180471
180472
180473
180474
180475
180476
180477
180478
180479
180480
180481
180482
180483
180484
180485
180486
180487
180488
180489
180490
180491
180492
180493
180494
180495
180496
180497
180498
180499
180500
180501
180502
180503
180504
180505
180506
180507
180508
180509
180510
180511
180512
180513
180514
180515
180516
180517
180518
180519
180520
180521
180522
180523
180524
180525
180526
180527
180528
180529
180530
180531
180532
180533
180534
180535
180536
180537
180538
180539
180540
180541
180542
180543
180544
180545
180546
180547
180548
180549
180550
180551
180552
180553
180554
180555
180556
180557
180558
180559
180560
180561
180562
180563
180564
180565
180566
180567
180568
180569
180570
180571
180572
180573
180574
180575
180576
180577
180578
180579
180580
180581
180582
180583
180584
180585
180586
180587
180588
180589
180590
180591
180592
180593
180594
180595
180596
180597
180598
180599
180600
180601
180602
180603
180604
180605
180606
180607
180608
180609
180610
180611
180612
180613
180614
180615
180616
180617
180618
180619
180620
180621
180622
180623
180624
180625
180626
180627
180628
180629
180630
180631
180632
180633
180634
180635
180636
180637
180638
180639
180640
180641
180642
180643
180644
180645
180646
180647
180648
180649
180650
180651
180652
180653
180654
180655
180656
180657
180658
180659
180660
180661
180662
180663
180664
180665
180666
180667
180668
180669
180670
180671
180672
180673
180674
180675
180676
180677
180678
180679
180680
180681
180682
180683
180684
180685
180686
180687
180688
180689
180690
180691
180692
180693
180694
180695
180696
180697
180698
180699
180700
180701
180702
180703
180704
180705
180706
180707
180708
180709
180710
180711
180712
180713
180714
180715
180716
180717
180718
180719
180720
180721
180722
180723
180724
180725
180726
180727
180728
180729
180730
180731
180732
180733
180734
180735
180736
180737
180738
180739
180740
180741
180742
180743
180744
180745
180746
180747
180748
180749
180750
180751
180752
180753
180754
180755
180756
180757
180758
180759
180760
180761
180762
180763
180764
180765
180766
180767
180768
180769
180770
180771
180772
180773
180774
180775
180776
180777
180778
180779
180780
180781
180782
180783
180784
180785
180786
180787
180788
180789
180790
180791
180792
180793
180794
180795
180796
180797
180798
180799
180800
180801
180802
180803
180804
180805
180806
180807
180808
180809
180810
180811
180812
180813
180814
180815
180816
180817
180818
180819
180820
180821
180822
180823
180824
180825
180826
180827
180828
180829
180830
180831
180832
180833
180834
180835
180836
180837
180838
180839
180840
180841
180842
180843
180844
180845
180846
180847
180848
180849
180850
180851
180852
180853
180854
180855
180856
180857
180858
180859
180860
180861
180862
180863
180864
180865
180866
180867
180868
180869
180870
180871
180872
180873
180874

```

```

1091                         MSG_INTL(MSG_MARG_INCOMP),
1092                         MSG_INTL(MSG_MARG_FILTER_AUX),
1093                         MSG_INTL(MSG_MARG_FILTER));
1094
1095             } else {
1096                 if ((ofl->ofl_filtees =
1097                     add_string(ofl->ofl_filtees, optarg)) ==
1098                         (const char *)S_ERROR)
1099                     return (S_ERROR);
1100                 ofl->ofl_flags |= FLG_OF_AUX;
1101             }
1102             break;
1103
1104         case 'F':
1105             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1106             if (ofl->ofl_filtees &&
1107                 (ofl->ofl_flags & FLG_OF_AUX)) {
1108                 ld_eprintf(ofl, ERR_FATAL,
1109                         MSG_INTL(MSG_MARG_INCOMP),
1110                         MSG_INTL(MSG_MARG_FILTER),
1111                         MSG_INTL(MSG_MARG_FILTER_AUX));
1112             } else {
1113                 if ((ofl->ofl_filtees =
1114                     add_string(ofl->ofl_filtees, optarg)) ==
1115                         (const char *)S_ERROR)
1116                     return (S_ERROR);
1117             }
1118             break;
1119
1120         case 'h':
1121             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1122             if (ofl->ofl_soname)
1123                 ld_eprintf(ofl, ERR_WARNING_NF,
1124                         MSG_INTL(MSG_MARG_MTONCE),
1125                         MSG_INTL(MSG_MARG SONAME));
1126             else
1127                 ofl->ofl_soname = (const char *)optarg;
1128             break;
1129
1130         case 'i':
1131             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1132             ofl->ofl_flags |= FLG_OF_IGNENV;
1133             break;
1134
1135         case 'I':
1136             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1137             if (ofl->ofl_interp)
1138                 ld_eprintf(ofl, ERR_WARNING_NF,
1139                         MSG_INTL(MSG_ARG_MTONCE),
1140                         MSG_ORIG(MSG_ARG_CI));
1141             else
1142                 ofl->ofl_interp = (const char *)optarg;
1143             break;
1144
1145         case 'l':
1146             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1147             /*
1148             * For now, count any library as a shared object. This
1149             * is used to size the internal symbol cache. This
1150             * value is recalculated later on actual file processing
1151             * to get an accurate shared object count.
1152             */
1153             ofl->ofl_soscnt++;
1154             break;
1155
1156         case 'm':
1157             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));

```

```

1157
1158             ofl->ofl_flags |= FLG_OF_GENMAP;
1159             break;
1160
1161         case 'o':
1162             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1163             if (ofl->ofl_name)
1164                 ld_eprintf(ofl, ERR_WARNING_NF,
1165                         MSG_INTL(MSG_MARG_MTONCE),
1166                         MSG_INTL(MSG_MARG_OUTFILE));
1167             else
1168                 ofl->ofl_name = (const char *)optarg;
1169             break;
1170
1171         case 'p':
1172             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1173
1174             /*
1175             * Multiple instances of this option may occur. Each
1176             * additional instance is effectively concatenated to
1177             * the previous separated by a colon.
1178             */
1179             if (*optarg != '\0') {
1180                 if ((ofl->ofl_audit =
1181                     add_string(ofl->ofl_audit,
1182                         optarg)) ==
1183                         (const char *)S_ERROR)
1184                     return (S_ERROR);
1185             }
1186             break;
1187
1188         case 'P':
1189             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1190
1191             /*
1192             * Multiple instances of this option may occur. Each
1193             * additional instance is effectively concatenated to
1194             * the previous separated by a colon.
1195             */
1196             if (*optarg != '\0') {
1197                 if ((ofl->ofl_deaudit =
1198                     add_string(ofl->ofl_deaudit,
1199                         optarg)) ==
1200                         (const char *)S_ERROR)
1201                     return (S_ERROR);
1202             }
1203             break;
1204
1205         case 'r':
1206             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1207             rflag = TRUE;
1208             break;
1209
1210         case 'R':
1211             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1212
1213             /*
1214             * Multiple instances of this option may occur. Each
1215             * additional instance is effectively concatenated to
1216             * the previous separated by a colon.
1217             */
1218             if (*optarg != '\0') {
1219                 if ((ofl->ofl_rpath =
1220                     add_string(ofl->ofl_rpath,
1221                         optarg)) ==
1222                         (const char *)S_ERROR)
1223                     return (S_ERROR);
1224             }
1225             break;

```

```

1223      case 's':
1224          DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1225          sflag = TRUE;
1226          break;
1227
1228      case 't':
1229          DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1230          ofl->ofl_flags |= FLG_OF_NOWARN;
1231          break;
1232
1233      case 'u':
1234          DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1235          break;
1236
1237      case 'z':
1238          DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1239
1240          /*
1241          * For specific help, print our usage message and exit
1242          * immediately to ensure a 0 return code.
1243          */
1244          if (strncmp(optarg, MSG_ORIG(MSG_ARG_HELP),
1245                      MSG_ARG_HELP_SIZE) == 0) {
1246              usage_msg(TRUE);
1247              exit(0);
1248          }
1249
1250          /*
1251          * For some options set a flag - further consistency
1252          * checks will be carried out in check_flags().
1253          */
1254          if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
1255                      MSG_ARG_LD32_SIZE) == 0) ||
1256              (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
1257                      MSG_ARG_LD64_SIZE) == 0)) {
1258              if (createargv(ofl, usage) == S_ERROR)
1259                  return (S_ERROR);
1260
1261          } else if (
1262              strncmp(optarg, MSG_ORIG(MSG_ARG_DEFS)) == 0) {
1263              if (zdflag != SET_UNKNOWN)
1264                  ld_eprintf(ofl, ERR_WARNING_NF,
1265                             MSG_INTL(MSG_ARG_MTONCE),
1266                             MSG_ORIG(MSG_ARG_ZDEFNODEF));
1267              else
1268                  zdflag = SET_TRUE;
1269              ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
1270          } else if (strncmp(optarg,
1271                      MSG_ORIG(MSG_ARG_NODEFS)) == 0) {
1272              if (zdflag != SET_UNKNOWN)
1273                  ld_eprintf(ofl, ERR_WARNING_NF,
1274                             MSG_INTL(MSG_ARG_MTONCE),
1275                             MSG_ORIG(MSG_ARG_ZDEFNODEF));
1276              else
1277                  zdflag = SET_FALSE;
1278              ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
1279          } else if (strncmp(optarg,
1280                      MSG_ORIG(MSG_ARG_TEXT)) == 0) {
1281              if (ztflag &&
1282                  (ztflag != MSG_ORIG(MSG_ARG_ZTEXT)))
1283                  ld_eprintf(ofl, ERR_FATAL,
1284                             MSG_INTL(MSG_ARG_INCOMP),
1285                             MSG_ORIG(MSG_ARG_ZTEXT),
1286                             ztflag);
1287              ztflag = MSG_ORIG(MSG_ARG_ZTEXT);
1288          } else if (strcmp(optarg,

```

```

1289          MSG_ORIG(MSG_ARG_TEXTOFF)) == 0) {
1290              if (ztflag &&
1291                  (ztflag != MSG_ORIG(MSG_ARG_ZTEXTOFF)))
1292                  ld_eprintf(ofl, ERR_FATAL,
1293                             MSG_INTL(MSG_ARG_INCOMP),
1294                             MSG_ORIG(MSG_ARG_ZTEXTOFF),
1295                             ztflag);
1296              ztflag = MSG_ORIG(MSG_ARG_ZTEXTOFF);
1297          } else if (strcmp(optarg,
1298                      MSG_ORIG(MSG_ARG_TEXTWARN)) == 0) {
1299              if (ztflag &&
1300                  (ztflag != MSG_ORIG(MSG_ARG_ZTEXTWARN)))
1301                  ld_eprintf(ofl, ERR_FATAL,
1302                             MSG_INTL(MSG_ARG_INCOMP),
1303                             MSG_ORIG(MSG_ARG_ZTEXTWARN),
1304                             ztflag);
1305              ztflag = MSG_ORIG(MSG_ARG_ZTEXTWARN);
1306
1307          /*
1308          * For other options simply set the ofl flags directly.
1309          */
1310
1311      } else if (strcmp(optarg,
1312                      MSG_ORIG(MSG_ARG_RESCAN)) == 0) {
1313          ofl->ofl_flags1 |= FLG_OF1_RESCAN;
1314      } else if (strcmp(optarg,
1315                      MSG_ORIG(MSG_ARG_ABSEXEC)) == 0) {
1316          ofl->ofl_flags1 |= FLG_OF1_ABSEXEC;
1317      } else if (strcmp(optarg,
1318                      MSG_ORIG(MSG_ARG_LOADFLTR)) == 0) {
1319          ziflag = TRUE;
1320      } else if (strcmp(optarg,
1321                      MSG_ORIG(MSG_ARG_NORELOC)) == 0) {
1322          ofl->ofl_dtflags_1 |= DF_1_NORELOC;
1323      } else if (strcmp(optarg,
1324                      MSG_ORIG(MSG_ARG_NOVERSION)) == 0) {
1325          ofl->ofl_flags |= FLG_OF_NOVERSEC;
1326      } else if (strcmp(optarg,
1327                      MSG_ORIG(MSG_ARG_MULDEFS)) == 0) {
1328          ofl->ofl_flags |= FLG_OF_MULDEFS;
1329      } else if (strcmp(optarg,
1330                      MSG_ORIG(MSG_ARG_REDLOCSYM)) == 0) {
1331          ofl->ofl_flags |= FLG_OF_REDLSYM;
1332      } else if (strcmp(optarg,
1333                      MSG_ORIG(MSG_ARG_INITFIRST)) == 0) {
1334          ofl->ofl_dtflags_1 |= DF_1_INITFIRST;
1335      } else if (strcmp(optarg,
1336                      MSG_ORIG(MSG_ARG_NODELETE)) == 0) {
1337          ofl->ofl_dtflags_1 |= DF_1_NODELETE;
1338      } else if (strcmp(optarg,
1339                      MSG_ORIG(MSG_ARG_NOPARTIAL)) == 0) {
1340          ofl->ofl_flags1 |= FLG_OF1_NOPARTI;
1341      } else if (strcmp(optarg,
1342                      MSG_ORIG(MSG_ARG_NOOPEN)) == 0) {
1343          ofl->ofl_dtflags_1 |= DF_1_NOOPEN;
1344      } else if (strcmp(optarg,
1345                      MSG_ORIG(MSG_ARG_NOW)) == 0) {
1346          ofl->ofl_dtflags_1 |= DF_1_NOW;
1347          ofl->ofl_dtflags |= DF_BIND_NOW;
1348      } else if (strcmp(optarg,
1349                      MSG_ORIG(MSG_ARG_ORIGIN)) == 0) {
1350          ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
1351          ofl->ofl_dtflags |= DF_ORIGIN;
1352      } else if (strcmp(optarg,
1353                      MSG_ORIG(MSG_ARG_NODEFAULTLIB)) == 0) {
1354          ofl->ofl_dtflags_1 |= DF_1_NODEFLIB;
1355      } else if (strcmp(optarg,

```

```

1355             MSG_ORIG(MSG_ARG_NODUMP)) == 0) {
1356                 ofl->ofl_dtflags_1 |= DF_1_NODUMP;
1357         } else if (strcmp(optarg,
1358                         MSG_ORIG(MSG_ARG_ENDFILTEE)) == 0) {
1359                 ofl->ofl_dtflags_1 |= DF_1_ENDFILTEE;
1360         } else if (strcmp(optarg,
1361                         MSG_ORIG(MSG_ARG_VERBOSE)) == 0) {
1362                 ofl->ofl_flags |= FLG_OF_VERBOSE;
1363         } else if (strcmp(optarg,
1364                         MSG_ORIG(MSG_ARG_COMBRELOC)) == 0) {
1365                 ofl->ofl_flags |= FLG_OF_COMREL;
1366         } else if (strcmp(optarg,
1367                         MSG_ORIG(MSG_ARG_NOCOMBRELOC)) == 0) {
1368                 ofl->ofl_flags |= FLG_OF_NOCOMREL;
1369         } else if (strcmp(optarg,
1370                         MSG_ORIG(MSG_ARG_NOCOMPSTRTAB)) == 0) {
1371                 ofl->ofl_flags1 |= FLG_OF1_NCSTTAB;
1372         } else if (strcmp(optarg,
1373                         MSG_ORIG(MSG_ARG_NOINTTRP)) == 0) {
1374                 ofl->ofl_flags1 |= FLG_OF1_NOINTTRP;
1375         } else if (strcmp(optarg,
1376                         MSG_ORIG(MSG_ARG_INTERPOSE)) == 0) {
1377                 zinflag = TRUE;
1378         } else if (strcmp(optarg,
1379                         MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
1380                 ofl->ofl_flags1 |= FLG_OF1_IGNPRC;
1381         } else if (strcmp(optarg,
1382                         MSG_ORIG(MSG_ARG_RELAXRELOC)) == 0) {
1383                 ofl->ofl_flags1 |= FLG_OF1_RLXREL;
1384         } else if (strcmp(optarg,
1385                         MSG_ORIG(MSG_ARG_NORELAXRELOC)) == 0) {
1386                 ofl->ofl_flags1 |= FLG_OF1_NRLXREL;
1387         } else if (strcmp(optarg,
1388                         MSG_ORIG(MSG_ARG_NOLDYNNSYM)) == 0) {
1389                 ofl->ofl_flags |= FLG_OF_NOLDYNNSYM;
1390         } else if (strcmp(optarg,
1391                         MSG_ORIG(MSG_ARG_GLOBAUDIT)) == 0) {
1392                 ofl->ofl_dtflags_1 |= DF_1_GLOBAUDIT;
1393         } else if (strcmp(optarg,
1394                         MSG_ORIG(MSG_ARG_NOSIGHANDLER)) == 0) {
1395                 ofl->ofl_flags1 |= FLG_OF1_NOSGHND;
1396         } else if (strcmp(optarg,
1397                         MSG_ORIG(MSG_ARG_SYMBOLCAP)) == 0) {
1398                 ofl->ofl_flags |= FLG_OF_OTOSCAP;
1399
1400         /*
1401         * Check archive group usage
1402         *      -z rescan-start ... -z rescan-end
1403         * to ensure they don't overlap and are well formed
1404         */
1405     } else if (strcmp(optarg,
1406                     MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {
1407         if (ofl->ofl_ars_gsndx == 0) {
1408             ofl->ofl_ars_gsndx = ndx;
1409         } else if (ofl->ofl_ars_gsndx > 0) {
1410             /* Another group is still open */
1411             ld_eprintf(ofl, ERR_FATAL,
1412                         MSG_INTL(MSG_ARG_AR_GRP_OLAP),
1413                         MSG_INTL(MSG_MARG_AR_GRPS));
1414             /* Don't report cascading errors */
1415             ofl->ofl_ars_gsndx = -1;
1416         }
1417     } else if (strcmp(optarg,
1418                     MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {
1419         if (ofl->ofl_ars_gsndx > 0) {
1420             ofl->ofl_ars_gsndx = 0;

```

```

1421     } else if (ofl->ofl_ars_gsndx == 0) {
1422         /* There was no matching begin */
1423         ld_eprintf(ofl, ERR_FATAL,
1424             MSG_INTL(MSG_ARG_AR_GRP_BAD),
1425             MSG_INTL(MSG_MARG_AR_GRP_END),
1426             MSG_INTL(MSG_MARG_AR_GRP_START));
1427         /* Don't report cascading errors */
1428         ofl->ofl_ars_gsndx = -1;
1429     }
1430
1431     /*
1432      * If -z wrap is seen, enter the symbol to be wrapped
1433      * into the wrap AVL tree.
1434      */
1435     } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_WRAP),
1436                         MSG_ARG_WRAP_SIZE) == 0) {
1437         if (ld_wrap_enter(ofl,
1438                           optarg + MSG_ARG_WRAP_SIZE) == NULL)
1439             return (S_ERROR);
1440     } else if ((strncmp(optarg, MSG_ORIG(MSG_ARG_GUIDE),
1441                         MSG_ARG_GUIDE_SIZE) == 0) &&
1442                ((optarg[MSG_ARG_GUIDE_SIZE] == '=') ||
1443                 (optarg[MSG_ARG_GUIDE_SIZE] == '\0'))) {
1444         if (!guidance_parse(ofl, optarg))
1445             return (S_ERROR);
1446     } else if (strcmp(optarg,
1447                         MSG_ORIG(MSG_ARG_FATWARN)) == 0) {
1448         if (zfwfflag == SET_FALSE) {
1449             ld_eprintf(ofl, ERR_WARNING_NF,
1450                         MSG_INTL(MSG_ARG_MTONCE),
1451                         MSG_ORIG(MSG_ARG_ZFATWNOFATW));
1452         } else {
1453             zfwfflag = SET_TRUE;
1454             ofl->ofl_flags |= FLG_OF_FATWARN;
1455         }
1456     } else if (strcmp(optarg,
1457                         MSG_ORIG(MSG_ARG_NOFATWARN)) == 0) {
1458         if (zfwfflag == SET_TRUE)
1459             ld_eprintf(ofl, ERR_WARNING_NF,
1460                         MSG_INTL(MSG_ARG_MTONCE),
1461                         MSG_ORIG(MSG_ARG_ZFATWNOFATW));
1462     else
1463         zfwfflag = SET_FALSE;
1464
1465     /*
1466      * Process everything related to -z assert-deflib. This
1467      * must be done in pass 1 because it gets used in pass
1468      * 2.
1469      */
1470     } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_ASSDEFLIB),
1471                         MSG_ARG_ASSDEFLIB_SIZE) == 0) {
1472         if (assdeflib_parse(ofl, optarg) != TRUE)
1473             return (S_ERROR);
1474
1475     /*
1476      * The following options just need validation as they
1477      * are interpreted on the second pass through the
1478      * command line arguments.
1479      */
1480     } else if (
1481         strncmp(optarg, MSG_ORIG(MSG_ARG_INITARRAY),
1482                         MSG_ARG_INITARRAY_SIZE) &&
1483         strncmp(optarg, MSG_ORIG(MSG_ARG_FINIARRAY),
1484                         MSG_ARG_FINIARRAY_SIZE) &&
1485         strncmp(optarg, MSG_ORIG(MSG_ARG_PREINITARRAY),
1486                         MSG_ARG_PREINITARRAY_SIZE) &&
1487         strncmp(optarg, MSG_ORIG(MSG_ARG_RTLDINFO),
1488                         MSG_ARG_RTLDINFO));

```

new/usr/src/cmd/sgs/libld/common/args.c

9

```

1487 MSG_ARG_RTLDINFO_SIZE) &&
1488 strncmp(optarg, MSG_ORIG(MSG_ARG_DTRACE), 
1489 MSG_ARG_DTRACE_SIZE) &&
1490 strcmp(optarg, MSG_ORIG(MSG_ARG_ALLEXTRT)) &&
1491 strcmp(optarg, MSG_ORIG(MSG_ARG_DFLEXTRT)) &&
1492 strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) &&
1493 strcmp(optarg, MSG_ORIG(MSG_ARG_NODIRECT)) &&
1494 strcmp(optarg, MSG_ORIG(MSG_ARG_GROUPPERM)) &&
1495 strcmp(optarg, MSG_ORIG(MSG_ARG_LAZYLOAD)) &&
1496 strcmp(optarg, MSG_ORIG(MSG_ARG_NOGROUPPERM)) &&
1497 strcmp(optarg, MSG_ORIG(MSG_ARG_NOLAZYLOAD)) &&
1498 strcmp(optarg, MSG_ORIG(MSG_ARG_NODEFERRED)) &&
1499 strcmp(optarg, MSG_ORIG(MSG_ARG_RECORD)) &&
1500 strcmp(optarg, MSG_ORIG(MSG_ARG_ALTEXEC64)) &&
1501 strcmp(optarg, MSG_ORIG(MSG_ARG_WEAKEXT)) &&
1502 strncmp(optarg, MSG_ORIG(MSG_ARG_TARGET),
1503 MSG_ARG_TARGET_SIZE) &&
1504 strcmp(optarg, MSG_ORIG(MSG_ARG_RESCAN_NOW)) &&
1505 strcmp(optarg, MSG_ORIG(MSG_ARG_DEFERRED))) {
1506     ld_eprintf(ofl, ERR_FATAL,
1507                 MSG_INTL(MSG_ARG_ILLEGAL),
1508                 MSG_ORIG(MSG_ARG_Z), optarg);
1509 }

1511     break;

1513 case 'D':
1514 /*
1515 * If we have not yet read any input files go ahead
1516 * and process any debugging options (this allows any
1517 * argument processing, entrance criteria and library
1518 * initialization to be displayed). Otherwise, if an
1519 * input file has been seen, skip interpretation until
1520 * process_files (this allows debugging to be turned
1521 * on and off around individual groups of files).
1522 */
1523 Dflag = 1;
1524 if (ofl->ofl_objscnt == 0) {
1525     if (dbg_setup(ofl, optarg, 2) == 0)
1526         return (S_ERROR);
1527 }

1529 /*
1530 * A diagnostic can only be provided after dbg_setup().
1531 * As this is the first diagnostic that can be produced
1532 * by ld(1), issue a title for timing and basic output.
1533 */
1534 if ((optitle == 0) && DBG_ENABLED) {
1535     optitle++;
1536     DBG_CALL(Dbg_basic_options(ofl->ofl_lml));
1537 }
1538 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1539 break;

1541 case 'B':
1542     DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1543     if (strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
1544         if (Bdflag == SET_FALSE) {
1545             ld_eprintf(ofl, ERR_FATAL,
1546                         MSG_INTL(MSG_ARG_INCOMP),
1547                         MSG_ORIG(MSG_ARG_BNODIRECT),
1548                         MSG_ORIG(MSG_ARG_BDIRECT));
1549     } else {
1550         Bdflag = SET_TRUE;
1551         ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1552     }

```

new/usr/src/cmd/sgs/libld/common/args.c

```

} else if (strcmp(optarg,
    MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
    if (Bdflag == SET_TRUE) {
        ld_eprintf(ofl, ERR_FATAL,
            MSG_INTL(MSG_ARG_INCOMP),
            MSG_ORIG(MSG_ARG_BDIRECT),
            MSG_ORIG(MSG_ARG_BNODIRECT));
    } else {
        Bdflag = SET_FALSE;
        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
    }
} else if (strcmp(optarg,
    MSG_ORIG(MSG_STR_SYMBOLIC)) == 0)
    Bsflag = TRUE;
else if (strcmp(optarg, MSG_ORIG(MSG_ARG_REDUCE)) == 0)
    ofl->ofl_flags |= FLG_OF_PROCRED;
else if (strcmp(optarg, MSG_ORIG(MSG_STR_LOCAL)) == 0)
    Blflag = TRUE;
else if (strcmp(optarg, MSG_ORIG(MSG_ARG_GROUP)) == 0)
    Bgflag = TRUE;
else if (strcmp(optarg,
    MSG_ORIG(MSG_STR_ELIMINATE)) == 0)
    Beflag = TRUE;
else if (strcmp(optarg,
    MSG_ORIG(MSG_ARG_TRANSLATOR)) == 0) {
    ld_eprintf(ofl, ERR_WARNING,
        MSG_INTL(MSG_ARG_UNSUPPORTED),
        MSG_ORIG(MSG_ARG_BTRANSLATOR));
}
else if (strcmp(optarg,
    MSG_ORIG(MSG_STR_LD_DYNAMIC)) &&
    strcmp(optarg, MSG_ORIG(MSG_ARG_STATIC))) {
    ld_eprintf(ofl, ERR_FATAL,
        MSG_INTL(MSG_ARG_ILLEGAL),
        MSG_ORIG(MSG_ARG_CB), optarg);
}
break;

G'':
DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
Gflag = TRUE;
break;

L'':
DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
break;

M'':
DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
if (aplist_append(&(ofl->ofl_maps), optarg,
    AL_CNT_OFIL_MAPFILES) == NULL)
    return (S_ERROR);
break;

N'':
DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
break;

Q'':
DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
if ((optarg[0] == 'n') && (optarg[1] == '\0')) {
    if (Qflag != SET_UNKNOWN)
        ld_eprintf(ofl, ERR_WARNING_NF,
            MSG_INTL(MSG_ARG_MTONCE),
            MSG_ORIG(MSG_ARG_CQ));
    else
        Qflag = SET_FALSE;
}

```

```

1619
1620     } else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
1621         if (Qflag != SET_UNKNOWN)
1622             ld_eprintf(ofl, ERR_WARNING_NF,
1623                         MSG_INTL(MSG_ARG_MTONCE),
1624                         MSG_ORIG(MSG_ARG_CQ));
1625         else
1626             Qflag = SET_TRUE;
1627     } else {
1628         ld_eprintf(ofl, ERR_FATAL,
1629                         MSG_INTL(MSG_ARG_ILLEGAL),
1630                         MSG_ORIG(MSG_ARG_CQ), optarg);
1631     }
1632     break;
1633
1634 case 'S':
1635     DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1636     if (aplist_append(&lib_support, optarg,
1637                         AL_CNT_SUPPORT) == NULL)
1638         return (S_ERROR);
1639     break;
1640
1641 case 'V':
1642     DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1643     if (!Vflag)
1644         (void) fprintf(stderr, MSG_ORIG(MSG_STR_STRNL),
1645                         ofl->ofl_sgsid);
1646     Vflag = TRUE;
1647     break;
1648
1649 case 'Y':
1650     DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
1651     if (strncmp(optarg, MSG_ORIG(MSG_ARG_LCOM), 2) == 0) {
1652         if (Llibdir)
1653             ld_eprintf(ofl, ERR_WARNING_NF,
1654                         MSG_INTL(MSG_ARG_MTONCE),
1655                         MSG_ORIG(MSG_ARG_CYL));
1656         else
1657             Llibdir = optarg + 2;
1658     } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_UCOM), 2) == 0) {
1659         if (Ulibdir)
1660             ld_eprintf(ofl, ERR_WARNING_NF,
1661                         MSG_INTL(MSG_ARG_MTONCE),
1662                         MSG_ORIG(MSG_ARG_CYU));
1663         else
1664             Ulibdir = optarg + 2;
1665     } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_PCOM), 2) == 0) {
1666         if (Plibpath)
1667             ld_eprintf(ofl, ERR_WARNING_NF,
1668                         MSG_INTL(MSG_ARG_MTONCE),
1669                         MSG_ORIG(MSG_ARG_CYP));
1670         else
1671             Plibpath = optarg + 2;
1672     } else {
1673         ld_eprintf(ofl, ERR_FATAL,
1674                         MSG_INTL(MSG_ARG_ILLEGAL),
1675                         MSG_ORIG(MSG_ARG_CY), optarg);
1676     }
1677     break;
1678
1679 case '?':
1680     DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
1681     /*
1682      * If the option character is '--', we're looking at a
1683      * long option which couldn't be translated, display a

```

```

1685                                     * more useful error.
1686                                     */
1687                                     if (optopt == '-') {
1688                                         eprintf(ofl->ofl_lml, ERR_FATAL,
1689                                             MSG_INTL(MSG_ARG_LONG_UNKNOWN),
1690                                             argv[optind-1]);
1691                                     } else {
1692 #endif /* ! codereview */
1693                                         eprintf(ofl->ofl_lml, ERR_FATAL,
1694                                             MSG_INTL(MSG_ARG_UNKNOWN), optopt);
1695                                     }
1696 #endif /* ! codereview */
1697                                     (*usage)++;
1698                                     break;
1699
1700     default:
1701         break;
1702     }
1703
1704     /*
1705      * Update the argument index for the next getopt() iteration.
1706      */
1707     ndx = optind;
1708 }
1709 return (1);
1710 }
1711
1712 /*
1713  * Parsing options pass2 for
1714  */
1715 static uintptr_t
1716 parseopt_pass2(Ofl_desc *ofl, int argc, char **argv)
1717 {
1718     int c, ndx = optind;
1719
1720     while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {
1721         Ifl_desc *ifl;
1722         Sym_desc *sdp;
1723
1724         switch (c) {
1725             case '1':
1726                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1727                                         optarg));
1728                 if (ld_find_library(optarg, ofl) == S_ERROR)
1729                     return (S_ERROR);
1730                 break;
1731             case 'B':
1732                 DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1733                                         optarg));
1734                 if (strcmp(optarg,
1735                         MSG_ORIG(MSG_STR_LD_DYNAMIC)) == 0) {
1736                     if (ofl->ofl_flags & FLG_OF_DYNAMIC)
1737                         ofl->ofl_flags |=
1738                             FLG_OF_DYNLIBS;
1739                 } else {
1740                     ld_eprintf(ofl, ERR_FATAL,
1741                         MSG_INTL(MSG_ARG_ST_INCOMP),
1742                         MSG_ORIG(MSG_ARG_BDYNAMIC));
1743                 }
1744             } else if (strcmp(optarg,
1745                         MSG_ORIG(MSG_ARG_STATIC)) == 0)
1746                 ofl->ofl_flags &= ~FLG_OF_DYNLIBS;
1747             break;
1748         case 'L':
1749             DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1750                                         optarg));

```

```

1751             if (ld_add_libdir(ofl, optarg) == S_ERROR)
1752                 return (S_ERROR);
1753         break;
1754     case 'N':
1755         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1756                                  optarg));
1757         /*
1758         * Record DT_NEEDED string
1759         */
1760         if (!(ofl->ofl_flags & FLG_OF_DYNAMIC))
1761             ld_eprintf(ofl, ERR_FATAL,
1762                        MSG_INTL(MSG_ARG_ST_INCOMP),
1763                        MSG_ORIG(MSG_ARG_CN));
1764         if (((ifl = libld_malloc(1,
1765                                sizeof (ifl_desc))) == NULL) ||
1766             (aplist_append(&ofl->ofl_sos, ifl,
1767                           AL_CNT_OFL_LIBS) == NULL)))
1768             return (S_ERROR);
1769
1770         ifl->ifl_name = MSG_INTL(MSG_STR_COMMAND);
1771         ifl->ifl_soname = optarg;
1772         ifl->ifl_flags = (FLG_IF_NEEDSTR |
1773                            FLG_IF_FILEREF | FLG_IF_DEPREQD);
1774
1775         break;
1776     case 'D':
1777         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1778                                  optarg));
1779         (void) dbg_setup(ofl, optarg, 3);
1780         break;
1781     case 'u':
1782         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1783                                  optarg));
1784         if (ld_sym_add_u(optarg, ofl,
1785                           MSG_STR_COMMAND) == (Sym_desc *)S_ERROR)
1786             return (S_ERROR);
1787         break;
1788     case 'z':
1789         DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
1790                                  optarg));
1791         if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
1792                      MSG_ARG_LD32_SIZE) == 0) ||
1793             (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
1794                      MSG_ARG_LD64_SIZE) == 0)) {
1795             if (createargv(ofl, 0) == S_ERROR)
1796                 return (S_ERROR);
1797         } else if (strcmp(optarg,
1798                           MSG_ORIG(MSG_ARG_ALLEXRT)) == 0) {
1799             ofl->ofl_flags1 |= FLG_OF1_ALLEXRT;
1800             ofl->ofl_flags1 &= ~FLG_OF1_WEAKEXT;
1801         } else if (strcmp(optarg,
1802                           MSG_ORIG(MSG_ARG_WEAKEXT)) == 0) {
1803             ofl->ofl_flags1 |= FLG_OF1_WEAKEXT;
1804             ofl->ofl_flags1 &= ~FLG_OF1_ALLEXRT;
1805         } else if (strcmp(optarg,
1806                           MSG_ORIG(MSG_ARG_DFLEXTRT)) == 0) {
1807             ofl->ofl_flags1 &=
1808                         ~(FLG_OF1_ALLEXRT |
1809                           FLG_OF1_WEAKEXT);
1810         } else if (strcmp(optarg,
1811                           MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
1812             ofl->ofl_flags1 |= FLG_OF1_ZDIRECT;
1813             ofl->ofl_guideflags |= FLG_OFG_NO_DB;
1814         } else if (strcmp(optarg,
1815                           MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
1816             ofl->ofl_flags1 &= ~FLG_OF1_ZDIRECT;

```

```

1817             ofl->ofl_guidedeflags |= FLG_OFG_NO_DB;
1818
1819         } else if (strcmp(optarg,
1820                         MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
1821             ofl->ofl_flags1 |= FLG_OF1_IGNORE;
1822         } else if (strcmp(optarg,
1823                         MSG_ORIG(MSG_ARG_RECORD)) == 0) {
1824             ofl->ofl_flags1 &= ~FLG_OF1_IGNORE;
1825         } else if (strcmp(optarg,
1826                         MSG_ORIG(MSG_ARG_LAZYLOAD)) == 0) {
1827             ofl->ofl_flags1 |= FLG_OF1_LAZYL;
1828             ofl->ofl_guidedeflags |= FLG_OFG_NO_LAZY;
1829         } else if (strcmp(optarg,
1830                         MSG_ORIG(MSG_ARG_NOLAZYLOAD)) == 0) {
1831             ofl->ofl_flags1 &= ~ FLG_OF1_LAZYL;
1832             ofl->ofl_guidedeflags |= FLG_OFG_NO_LAZY;
1833         } else if (strcmp(optarg,
1834                         MSG_ORIG(MSG_ARG_GROUPPERM)) == 0) {
1835             ofl->ofl_flags1 |= FLG_OF1_GRPPRM;
1836         } else if (strcmp(optarg,
1837                         MSG_ORIG(MSG_ARG_NOGROUPPERM)) == 0) {
1838             ofl->ofl_flags1 &= ~FLG_OF1_GRPPRM;
1839         } else if (strncmp(optarg,
1840                         MSG_ARG_INITARRAY),
1841                         MSG_ARG_INITARRAY_SIZE) == 0) {
1842             if (((sdp = ld_sym_add_u(optarg +
1843                                         MSG_ARG_INITARRAY_SIZE, ofl,
1844                                         MSG_STR_COMMAND)) ==
1845                                         (Sym_desc *)S_ERROR) ||
1846                 (aplist_append(&ofl->ofl_initarray,
1847                               sdp, AL_CNT_OFL_ARRAYS) == NULL))
1848                 return (S_ERROR);
1849         } else if (strncmp(optarg,
1850                         MSG_ORIG(MSG_ARG_FINIARRAY),
1851                         MSG_ARG_FINIARRAY_SIZE) == 0) {
1852             if (((sdp = ld_sym_add_u(optarg +
1853                                         MSG_ARG_FINIARRAY_SIZE, ofl,
1854                                         MSG_STR_COMMAND)) ==
1855                                         (Sym_desc *)S_ERROR) ||
1856                 (aplist_append(&ofl->ofl_finiarray,
1857                               sdp, AL_CNT_OFL_ARRAYS) == NULL))
1858                 return (S_ERROR);
1859         } else if (strncmp(optarg,
1860                         MSG_ORIG(MSG_ARG_PREINITARRAY),
1861                         MSG_ARG_PREINITARRAY_SIZE) == 0) {
1862             if (((sdp = ld_sym_add_u(optarg +
1863                                         MSG_ARG_PREINITARRAY_SIZE, ofl,
1864                                         MSG_STR_COMMAND)) ==
1865                                         (Sym_desc *)S_ERROR) ||
1866                 (aplist_append(&ofl->ofl_preiaarray,
1867                               sdp, AL_CNT_OFL_ARRAYS) == NULL))
1868                 return (S_ERROR);
1869         } else if (strncmp(optarg,
1870                         MSG_ORIG(MSG_ARG_RTLDINFO),
1871                         MSG_ARG_RTLDINFO_SIZE) == 0) {
1872             if (((sdp = ld_sym_add_u(optarg +
1873                                         MSG_ARG_RTLDINFO_SIZE, ofl,
1874                                         MSG_STR_COMMAND)) ==
1875                                         (Sym_desc *)S_ERROR) ||
1876                 (aplist_append(&ofl->ofl_rtldinfo,
1877                               sdp, AL_CNT_OFL_ARRAYS) == NULL))
1878                 return (S_ERROR);
1879         } else if (strncmp(optarg,
1880                         MSG_ORIG(MSG_ARG_DTRACE),
1881                         MSG_ARG_DTRACE_SIZE) == 0) {
1882             if (((sdp = ld_sym_add_u(optarg +

```

```

1883                         MSG_STR_COMMAND)) ==  
1884                         (Sym_desc *)S_ERROR)  
1885                         return (S_ERROR);  
1886                 ofl->ofl_dtracesym = sdp;  
1887             } else if (strcmp(optarg,  
1888                         MSG_ORIG(MSG_ARG_RESCAN_NOW)) == 0) {  
1889                 if (ld_rescan_archives(ofl, 0, ndx) ==  
1890                     S_ERROR)  
1891                     return (S_ERROR);  
1892             } else if (strcmp(optarg,  
1893                         MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {  
1894                 ofl->ofl_ars_gndx = ofl->ofl_arscnt;  
1895                 ofl->ofl_ars_gndx = ndx;  
1896             } else if (strcmp(optarg,  
1897                         MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {  
1898                 if (ld_rescan_archives(ofl, 1, ndx) ==  
1899                     S_ERROR)  
1900                     return (S_ERROR);  
1901             } else if (strcmp(optarg,  
1902                         MSG_ORIG(MSG_ARG_DEFERRED)) == 0) {  
1903                 ofl->ofl_flags1 |= FLG_OF1_DEFERRED;  
1904             } else if (strcmp(optarg,  
1905                         MSG_ORIG(MSG_ARG_NODEFERRED)) == 0) {  
1906                 ofl->ofl_flags1 &= ~FLG_OF1_DEFERRED;  
1907             }  
1908         default:  
1909             break;  
1910     }  
1911  
1912     /*  
1913      * Update the argument index for the next getopt() iteration.  
1914      */  
1915     ndx = optind;  
1916 }  
1917 return (1);  
1918 }  
1919 /*  
1920 *  
1921 * Pass 1 -- process_flags: collects all options and sets flags  
1922 */  
1923 static uintptr_t  
1924 process_flags_com(Ofl_desc *ofl, int argc, char **argv, int *usage)  
1925 {  
1926     for (; optind < argc; optind++) {  
1927         /*  
1928          * If we detect some more options return to getopt().  
1929          * Checking argv[optind][1] against null prevents a forever  
1930          * loop if an unadorned '-' argument is passed to us.  
1931          */  
1932         while ((optind < argc) && (argv[optind][0] == '-')) {  
1933             if (argv[optind][1] != '\0') {  
1934                 if (parseopt_pass1(ofl, argc, argv,  
1935                               usage) == S_ERROR)  
1936                     return (S_ERROR);  
1937             } else if (++optind < argc)  
1938                 continue;  
1939         }  
1940         if (optind >= argc)  
1941             break;  
1942         ofl->ofl_objscnt++;  
1943     }  
1944  
1945     /* Did an unterminated archive group run off the end? */  
1946     if (ofl->ofl_ars_gndx > 0) {  
1947         ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_ARG_AR_GRP_BAD),  
1948

```

```

1949                         MSG_INTL(MSG_MARG_AR_GRP_START),  
1950                         MSG_INTL(MSG_MARG_AR_GRP_END));  
1951             }  
1952         }  
1953     }  
1954     return (1);  
1955 }  
1956  
1957 uintptr_t  
1958 ld_process_flags(Ofl_desc *ofl, int argc, char **argv)  
1959 {  
1960     int      usage = 0;      /* Collect all argument errors before exit */  
1961  
1962     if (argc < 2) {  
1963         usage_mesg(FALSE);  
1964         return (S_ERROR);  
1965     }  
1966  
1967     /*  
1968      * Option handling  
1969      */  
1970     optarg = 0;  
1971     optind = 1;  
1972     if (process_flags_com(ofl, argc, argv, &usage) == S_ERROR)  
1973         return (S_ERROR);  
1974  
1975     /*  
1976      * Having parsed everything, did we have any usage errors.  
1977      */  
1978     if (usage) {  
1979         eprintf(ofl->ofl_lml, ERR_FATAL, MSG_INTL(MSG_ARG_USEHELP));  
1980         return (S_ERROR);  
1981     }  
1982  
1983     return (check_flags(ofl, argc));  
1984 }  
1985  
1986 /*  
1987  * Pass 2 -- process_files: skips the flags collected in pass 1 and processes  
1988  * files.  
1989 */  
1990 static uintptr_t  
1991 process_files_com(Ofl_desc *ofl, int argc, char **argv)  
1992 {  
1993     for (; optind < argc; optind++) {  
1994         int      fd;  
1995         uintptr_t  open_ret;  
1996         char      *path;  
1997         Rej_desc   rej = { 0 };  
1998  
1999     /*  
2000      * If we detect some more options return to getopt().  
2001      * Checking argv[optind][1] against null prevents a forever  
2002      * loop if an unadorned '-' argument is passed to us.  
2003      */  
2004         while ((optind < argc) && (argv[optind][0] == '-')) {  
2005             if (argv[optind][1] != '\0') {  
2006                 if (parseopt_pass2(ofl, argc, argv) == S_ERROR)  
2007                     return (S_ERROR);  
2008             } else if (++optind < argc)  
2009                 continue;  
2010         }  
2011         if (optind >= argc)  
2012             break;  
2013         path = argv[optind];

```

```

2015     if ((fd = open(path, O_RDONLY)) == -1) {
2016         int err = errno;
2017
2018         ld_eprintf(ofl, ERR_FATAL,
2019             MSG_INTL(MSG_SYS_OPEN), path, strerror(err));
2020         continue;
2021     }
2022
2023     DBG_CALL(Dbg_args_file(ofl->ofl_lml, optind, path));
2024
2025     open_ret = ld_process_open(path, path, &fd, ofl,
2026         (FLG_IF_CMDLINE | FLG_IF_NEEDED), &rej, NULL);
2027     if (fd != -1)
2028         (void) close(fd);
2029     if (open_ret == S_ERROR)
2030         return (S_ERROR);
2031
2032     /*
2033      * Check for mismatched input.
2034      */
2035     if (rej.rej_type) {
2036         Conv_reject_desc_buf_t rej_buf;
2037
2038         ld_eprintf(ofl, ERR_FATAL,
2039             MSG_INTL(reject[rej.rej_type]),
2040             rej.rej_name ? rej.rej_name :
2041             MSG_INTL(MSG_STR_UNKNOWN),
2042             conv_reject_desc(&rej, &rej_buf,
2043                 ld_targ.t_m.m_mach));
2044         return (1);
2045     }
2046 }
2047
2048 }
2049
2050 uintptr_t
2051 ld_process_files(Ofl_desc *ofl, int argc, char **argv)
2052 {
2053     DBG_CALL(Dbg_basic_files(ofl->ofl_lml));
2054
2055     /*
2056      * Process command line files (taking into account any applicable
2057      * preceding flags).  Return if any fatal errors have occurred.
2058      */
2059     optarg = 0;
2060     optind = 1;
2061     if (process_files_com(ofl, argc, argv) == S_ERROR)
2062         return (S_ERROR);
2063     if (ofl->ofl_flags & FLG_OF_FATAL)
2064         return (1);
2065
2066     /*
2067      * Guidance: Use -B direct/nodirect or -z direct/nodirect.
2068      *
2069      * This is a backstop for the case where the link had no dependencies.
2070      * Otherwise, it will get caught by ld_process_ifl(). We need both,
2071      * because -z direct is positional, and its value at the time where
2072      * the first dependency is seen might be different than it is now.
2073      */
2074     if ((ofl->ofl_flags & FLG_OF_DYNAMIC) &&
2075         OFL_GUIDANCE(ofl, FLG_OFG_NO_DB)) {
2076         ld_eprintf(ofl, ERR_GUIDANCE, MSG_INTL(MSG_GUIDE_DIRECT));
2077         ofl->ofl_guideflags |= FLG_OFG_NO_DB;
2078     }
2079
2080     /*

```

```

2081         * Now that all command line files have been processed see if there are
2082         * any additional 'needed' shared object dependencies.
2083         */
2084         if (ofl->ofl_soneed)
2085             if (ld_finish_libs(ofl) == S_ERROR)
2086                 return (S_ERROR);
2087
2088         /*
2089          * If rescanning archives is enabled, do so now to determine whether
2090          * there might still be members extracted to satisfy references from any
2091          * explicit objects. Continue until no new objects are extracted. Note
2092          * that this pass is carried out *after* processing any implicit objects
2093          * (above) as they may already have resolved any undefined references
2094          * from any explicit dependencies.
2095          */
2096         if (ofl->ofl_flags1 & FLG_OF1_RESCAN) {
2097             if (ld_rescan_archives(ofl, 0, argc) == S_ERROR)
2098                 return (S_ERROR);
2099             if (ofl->ofl_flags & FLG_OF_FATAL)
2100                 return (1);
2101         }
2102
2103         /*
2104          * If debugging, provide statistics on each archive extraction, or flag
2105          * any archive that has provided no members. Note that this could be a
2106          * nice place to free up much of the archive infrastructure, as we've
2107          * extracted any members we need. However, as we presently don't free
2108          * anything under ld(1) there's not much point in proceeding further.
2109          */
2110         DBG_CALL(Dbg_statistics_ar(ofl));
2111
2112         /*
2113          * If any version definitions have been established, either via input
2114          * from a mapfile or from the input relocatable objects, make sure any
2115          * version dependencies are satisfied, and version symbols created.
2116          */
2117         if (ofl->ofl_verdesc)
2118             if (ld_vers_check_defs(ofl) == S_ERROR)
2119                 return (S_ERROR);
2120
2121         /*
2122          * If input section ordering was specified within some segment
2123          * using a mapfile, verify that the expected sections were seen.
2124          */
2125         if (ofl->ofl_flags & FLG_OF_IS_ORDER)
2126             ld_ent_check(ofl);
2127
2128     }
2129
2130     uintptr_t
2131     ld_init_strings(Ofl_desc *ofl)
2132     {
2133         uint_t stflags;
2134
2135         if (ofl->ofl_flags1 & FLG_OF1_NCSTTAB)
2136             stflags = 0;
2137         else
2138             stflags = FLG_STNEW_COMPRESS;
2139
2140         if (((ofl->ofl_shdrsttab = st_new(stflags)) == NULL) ||
2141             ((ofl->ofl_strtab = st_new(stflags)) == NULL) ||
2142             ((ofl->ofl_dynstrtab = st_new(stflags)) == NULL))
2143             return (S_ERROR);
2144
2145     }
2146

```

2147 }

new/usr/src/cmd/sgs/libld/common/libld.msg

```
*****
60246 Sun Oct 27 14:16:04 2013
new/usr/src/cmd/sgs/libld/common/libld.msg
4270 ld(1) argument error reporting is still pretty bad
*****
```

1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at [usr/src/OPENSOLARIS.LICENSE](#)
9 # or <http://www.opensolaris.org/os/licensing>.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at [usr/src/OPENSOLARIS.LICENSE](#).
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
22 # Copyright (c) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
26 # Copyright (c) 2012, Joyent, Inc. All rights reserved.
28 #
30 @_START_
32 # Message file for cmd/sgs/libld.
34 @ MSG_ID_LIBLD
36 #
37 # TRANSLATION_NOTE -- Beginning of USAGE message
38 # The following messages are the usage messages for the ld command.
39 # Tab characters (\t) are used to align the messages.
40 #
41 # Each usage message starts with \t, and if the message has more than one
42 # line, the following messages are aligned by 3 tab characters.
43 # When you see \n\t\t\t, the first \n is used to change the line,
44 # and following 3 tab characters are used to align the line.
45 #
46 # Each usage message option is surrounded by [and]. Then the
47 # description of the option follows. The descriptions should be aligned,
48 # so tab characters are padded as needed after the closing bracket].
49 #
50 # How to align the messages are up to the translators and the
51 # localization engineers.
52 #
53 # In C locale, the first 3 messages would look like the following:
54 #
55 # usage: ld [-6:abc:.....] file(s)
56 # [-a] create an absolute file
57 # [-b] do not do special PIC relocations in a.out
58 # [-c file] record configuration 'file'
59 #
60 @ MSG_ARG_USAGE "usage: ld [-%s] file(s)\n"
61 @ MSG_ARG_DETAIL_3 "\t[-32]\t\tenforce a 32-bit link-edit\n"

1

new/usr/src/cmd/sgs/libld/common/libld.msg

```
62 @ MSG_ARG_DETAIL_6      "\t[-64]\t\tenforce a 64-bit link-edit\n"  
63 @ MSG_ARG_DETAIL_A     "\t[-a]\t\tcreate an absolute file\n"  
64 @ MSG_ARG_DETAIL_B     "\t[-b]\t\tndo not do special PIC relocations in a.out\n"  
65 @ MSG_ARG_DETAIL_CBDR  "\t[-B direct | nodirect]\n"  
66                                "\t\t\ttestablish direct bindings, or inhibit direct \n"  
67                                binding\n"  
68                                "\t\t\ttto, the object being created\n"  
69 @ MSG_ARG_DETAIL_CBDY  "\t[-B dynamic | static]\n"  
70                                "\t\t\ttsearch for shared libraries|archives\n"  
71 @ MSG_ARG_DETAIL_CBE   "\t[-B eliminate]\t\teliminate unqualified global \n"  
72                                symbols from the\n\t\t\ttsymbol table\n"  
73 @ MSG_ARG_DETAIL_CBG   "\t[-B group]\t\trelocate object from within group\n"  
74 @ MSG_ARG_DETAIL_CBL   "\t[-B local]\t\treduce unqualified global symbols to \n"  
75                                local\n"  
76 @ MSG_ARG_DETAIL_CBR   "\t[-B reduce]\t\tprocess symbol reductions\n"  
77 @ MSG_ARG_DETAIL_CBS   "\t[-B symbolic]\t\tbind external references to \n"  
78                                definitions when creating\n"  
79                                "\t\t\ttshared objects\n"  
80 @ MSG_ARG_DETAIL_C      "\t[-c name]\t\trecord configuration file 'name'\n"  
81 @ MSG_ARG_DETAIL_CC    "\t[-C]\t\ttdemangle C++ symbol name diagnostics\n"  
82 @ MSG_ARG_DETAIL_D     "\t[-d y | n]\t\toperate in dynamic|static mode\n"  
83 @ MSG_ARG_DETAIL_CD    "\t[-D token,...]\t\tprint diagnostic messages\n"  
84 @ MSG_ARG_DETAIL_E     "\t[-e epsym], [--entry epsym]\n"  
85                                "\t\t\ttuse 'epsym' as entry point address\n"  
86 @ MSG_ARG_DETAIL_F     "\t[-f name], [--auxiliary name]\n"  
87                                "\t\t\ttspecify library for which this file is an \n"  
88                                auxiliary\n\t\t\ttfilter\n"  
89 @ MSG_ARG_DETAIL_CF    "\t[-F name], [--filter name]\n"  
90                                "\t\t\ttspecify library for which this file is a filter\n"  
91 @ MSG_ARG_DETAIL_CG    "\t[-G], [-shared]\n"  
92                                "\t\t\ttcreate a shared object\n"  
93 @ MSG_ARG_DETAIL_H     "\t[-h name], [--soname name]\n"  
94                                "\t\t\ttuse 'name' as internal shared object identifier\n"  
95 @ MSG_ARG_DETAIL_I     "\t[-i]\t\t\tignore LD_LIBRARY_PATH setting\n"  
96 @ MSG_ARG_DETAIL_CI    "\t[-I name]\t\tuse 'name' as path of interpreter\n"  
97 @ MSG_ARG_DETAIL_L     "\t[-l x], [-library x]\n"  
98                                "\t\t\ttsearch for libx.so or libx.a\n"  
99 @ MSG_ARG_DETAIL_CL    "\t[-L path], [--library-path path]\n"  
100 @ MSG_ARG_DETAIL_M    "\t\t\ttsearch for libraries in directory 'path'\n"  
102 @ MSG_ARG_DETAIL_CM   "\t[-m]\t\t\tprint memory map\n"  
103 @ MSG_ARG_DETAIL_CN   "\t[-M mapfile]\t\tuse processing directives contained \n"  
104 @ MSG_ARG_DETAIL_O     "\t\t\tin 'mapfile'\n"  
105 @ MSG_ARG_DETAIL_P     "\t[-N string]\t\tcreate a dynamic dependency for \n"  
106 @ MSG_ARG_DETAIL_O     "string\n"  
107 @ MSG_ARG_DETAIL_P     "\t\t\ttname the output file 'outfile'\n"  
108 @ MSG_ARG_DETAIL_P     "\t[-p auditlib]\t\tidentify audit library to accompany \n"  
109                                this object\n"  
110 @ MSG_ARG_DETAIL_CP    "\t[-P auditlib]\t\tidentify audit library for \n"  
111                                processing the dependencies\n"  
112                                "\t\t\ttthis object\n"  
113 @ MSG_ARG_DETAIL_CQ    "\t[-Q y | n]\t\tndo not place version information in \n"  
114                                output file\n"  
115 @ MSG_ARG_DETAIL_R    "\t[-r], [-relocatable]\n"  
116                                "\t\t\ttcreate a relocatable object\n"  
117 @ MSG_ARG_DETAIL_CR    "\t[-R path], [-rpath path]\n"  
118                                "\t\t\ttspecify a library search path to be used at run \n"  
119                                time\n"  
120 @ MSG_ARG_DETAIL_S     "\t[-s], [--strip-all]\n"  
121                                "\t\t\ttstrip any symbol and debugging information\n"  
122 @ MSG_ARG_DETAIL_CS    "\t[-S supportlib]\n"  
123                                "\t\t\ttspecify a link-edit support library\n"  
124 @ MSG_ARG_DETAIL_T     "\t[-T]\t\t\tndo not warn of multiply-defined symbols \n"  
125                                that have\n\t\t\ttdifferent sizes or alignments\n"  
126 @ MSG_ARG_DETAIL_U     "\t[-u symname], [--undefined symname]\n"  
127                                "\t\t\ttcreate an undefined symbol 'symname'\n"
```

2

```

128 @ MSG_ARG_DETAIL_CV      "\t[-v], [--version]\n"
129                               \t\tprint version information\n"
130 @ MSG_ARG_DETAIL_CY      "\t[-Y P,dirlist]\tuse 'dirlist' as a default path \
131                               when searching for\n"
132                               \t\tlibraries\n"
133 @ MSG_ARG_DETAIL_ZA      "\t[-z absexec]\twhen building an executable absolute \
134                               symbols\n \
135                               \t\tpreferenced in dynamic objects are promoted to\n \
136                               \t\tthe executable\n"
137 @ MSG_ARG_DETAIL_ZAE     "\t[-z allextract | defaultextract | weakextract],\n \
138                               \t[-whole-archive | --no-whole-archive]\n \
139                               \t\texttract all member files, only members that \
140                               \t\tresolve\n"
141                               \t\tpundefined or tentative symbols, or \
142                               \t\tallow extraction of\n \
143                               \t\tarchive members to resolve weak references from \
144                               \n\lt\larchive files\n"
145 @ MSG_ARG_DETAIL_ZAL     "\t[-z altexec64]\texecute the 64-bit link-editor\n"
146 @ MSG_ARG_DETAIL_ZADLIB  "\t[-z assert-deflib]\n"
147                               \t\tenables warnings for linking with libraries in \
148                               \n\lt\ldefault search path\n \
149                               \t[-z assert-deflib-libname]\n"
150                               \t\tenables warnings for linking with libraries in \
151                               \n\lt\ldefault search path, but 'libname' is exempt \
152                               \t[-z combreloc | nocombrelloc]\n \
153                               \t\tpcombine|do not combine multiple relocation \
154                               \t\tsections\n"
155 @ MSG_ARG_DETAIL_ZNC     "\t[-z nocmpstrtab]\n\t\tdisable compression of \
156                               \t\tstring tables\n"
157 @ MSG_ARG_DETAIL_ZDEF    "\t[-z deferred | nodeferred]\n \
158                               \t\tenable|disable deferred identification of \
159                               \t\tshared object\n\lt\tdependencies\n"
160 @ MSG_ARG_DETAIL_ZDFS   "\t[-z defs], [--no-undefined]\n \
161                               \t\tdisallow undefined symbol references\n"
162 @ MSG_ARG_DETAIL_ZDRS   "\t[-z direct | nodirect]\n \
163                               \t\tenable|disable direct binding to shared object\n \
164                               \t\tdependencies\n"
165 @ MSG_ARG_DETAIL_ZE     "\t[-z endfiltee]\tmarks a filtee such that it will \
166                               \t\tterminate a filters\n\lt\lsearch\n"
167 @ MSG_ARG_DETAIL_ZFATW  "\t[-z fatal-warnings | nofatal-warnings],\n \
168                               \t[--fatal-warnings | --no-fatal-warnings]\n \
169                               \t\tenable|disable treatment of warnings as fatal\n"
170 @ MSG_ARG_DETAIL_ZFA    "\t[-z finiarray-function]\n \
171                               \t\tname of function to be appended to the \
172                               \t\t.fini_array\n"
173 @ MSG_ARG_DETAIL_ZGP    "\t[-z groupperm | nogroupperm]\n \
174                               \t\tenable|disable setting of group permissions\n \
175                               \t\tion dynamic dependencies\n"
176 @ MSG_ARG_DETAIL_ZGUIDE "\t[-z guidance | -z guidance=item1,item2,...]\n \
177                               \t\tenable guidance warnings. items: \
178                               \nnoall, nodefs,\n \
179                               \t\tindirect, nolazyload, nomapfile, notext, \
180                               \t\tounused\n"
181 @ MSG_ARG_DETAIL_ZH     "\t[-z help], [--help]\n \
182                               \t\tprint this usage message\n"
183 @ MSG_ARG_DETAIL_ZIG    "\t[-z ignore | record]\n \
184                               \t\tpignore|record unused dynamic dependencies\n"
185 @ MSG_ARG_DETAIL_ZINA   "\t[-z initarray=function]\n \
186                               \t\tname of function to be appended to the \
187                               \t\t.init_array\n"
188 @ MSG_ARG_DETAIL_ZINI   "\t[-z initfirst]\tmark object to indicate that its \
189                               \t\tinit section should\n \
190                               \t\tbe executed before the .init section of any \
191                               \t\tother\n\lt\t\tobjects\n"
192 @ MSG_ARG_DETAIL_ZINT   "\t[-z interpose]\n \
193                               \t\tdynamic object is to be an 'interposer' on direct\n"

```

```

194                               \t\t\tbindings\n"
195 @ MSG_ARG_DETAIL_ZLAZY  "\t[-z lazyload | nolazyload]\n \
196                               \t\tenable|disable delayed loading of shared \
197                               \t\tobject\n\lt\t\tdependencies\n"
198 @ MSG_ARG_DETAIL_ZLD32  "\t[-z ld32=arg1,arg2,...]\n \
199                               \t\tdefine arguments applicable to the \
200                               \t\t\t32-bit class of ld(1)\n"
201 @ MSG_ARG_DETAIL_ZLD64  "\t[-z ld64=arg1,arg2,...]\n \
202                               \t\tdefine arguments applicable to the \
203                               \t\t\t64-bit class of ld(1)\n"
204 @ MSG_ARG_DETAIL_ZLO    "\t[-z loadfltr]\tmark filter as requiring immediate \
205                               \t\tloading of its\n \
206                               \t\t\tfiltees at runtime\n"
207 @ MSG_ARG_DETAIL_ZM    "\t[-z muldefs], [--allow-multiple-definition]\n \
208                               \t\tallow multiply-defined symbols\n"
209 @ MSG_ARG_DETAIL_ZNDFS  "\t[-z nodefs]\tallow undefined symbol references\n"
210 @ MSG_ARG_DETAIL_ZNDEF  "\t[-z defaultlib]\n \
211                               \t\tmark object to ignore any default library \
212                               \t\t\tsearch path\n"
213 @ MSG_ARG_DETAIL_ZNDEL  "\t[-z nodelete]\tmark object as non-deletable\n"
214 @ MSG_ARG_DETAIL_ZNDO   "\t[-z nodlopen]\tmark object as non-dlopen()'able\n"
215 @ MSG_ARG_DETAIL_ZNDU   "\t[-z nodump]\tmark object as non-dldump()'able\n"
216 @ MSG_ARG_DETAIL_ZNLD   "\t[-z noldynsym]\tdo not add a .SUNW_ldynsym section\n"
217 @ MSG_ARG_DETAIL_ZNPA   "\t[-z nopartial]\texpand any partially initialized \
218                               \t\tsymbols\n"
219 @ MSG_ARG_DETAIL_ZNV    "\t[-z noversion]\tdo not record any version sections\n"
220 @ MSG_ARG_DETAIL_ZNOW   "\t[-z now]\tmark object as requiring non-lazy \
221                               \t\tbinding\n"
222 @ MSG_ARG_DETAIL_ZO    "\t[-z origin]\tmark object as requiring $ORIGIN \
223                               \t\tprocessing\n"
224 @ MSG_ARG_DETAIL_ZPIA   "\t[-z preinitarray=function]\n \
225                               \t\tname of function to be appended to the \
226                               \t\t\t.preinit_array\n"
227 @ MSG_ARG_DETAIL_ZRL    "\t[-z redlocsym]\treduce local syms in .symtab to \
228                               \t\t\tminimum\n"
229 @ MSG_ARG_DETAIL_ZRREL  "\t[-z relaxreloc]\trelax rules used for relocations \
230                               \t\tagainst COMDAT sections\n"
231 @ MSG_ARG_DETAIL_ZRS   "\t[-z rescans]\tafter processing all arguments, rescan \
232                               \t\tarchive list\n"
233 @ MSG_ARG_DETAIL_ZRSN  "\t[-z until no further member extraction occurs]\n"
234 @ MSG_ARG_DETAIL_ZRSN  "\t[-z rescan-now]\timmediately rescan archive list \
235                               \t\tuntil\n \
236                               \t\t\tno further member extraction occurs\n"
237 @ MSG_ARG_DETAIL_ZRSGRP "\t[-z rescan-start archives... -z rescan-end],\n \
238                               \t\t\tstart-group archives... --end-group], \
239                               \t\t\t(- archives... -)]\n \
240                               \t\t\trescan specified archive group upon reaching\n \
241                               \t\t\tthe end of the group, until no further\n \
242                               \t\t\tmember extraction occurs\n"
243 @ MSG_ARG_DETAIL_ZSCAP  "\t[-z symbolcap]\tconvert object capabilities to \
244                               \t\t\tsymbol capabilities\n"
245 @ MSG_ARG_DETAIL_ZTARG  "\t[-z target=platform]\n \
246                               \t\ttarget machine for cross linking\n"
247 @ MSG_ARG_DETAIL_ZT    "\t[-z text]\tdisallow output relocations against \
248                               \t\t\ttext\n"
249 @ MSG_ARG_DETAIL_ZTO   "\t[-z textoff]\tallow output relocations against \
250                               \t\t\ttext\n"
251 @ MSG_ARG_DETAIL_ZTW   "\t[-z textwarn]\twarn if there are relocations \
252                               \t\tagainst text\n"
253 @ MSG_ARG_DETAIL_ZWRAP  "\t[-z wrap=symbol], [-wrap=symbol], [--wrap=symbol]\n \
254                               \t\twrap symbol references\n"
255 @ MSG_ARG_DETAIL_ZVER   "\t[-z verbose]\t\
256                               \t\tgenerate warnings for suspicious processings\n"
258 #
259 # TRANSLATION_NOTE -- End of USAGE message

```

```

260 #
261 @ MSG_GRP_INVALNDX "file %s: group section [%u]s: entry %d: \
262           invalid section index: %d"
263 @ MSG_GRP_INVALSYM "file %s: group section [%u]s: invalid group symbol %s"

265 # Relocation processing messages (some of these are required to satisfy
266 # do_reloc(), which is common code used by cmd/sgs/rtld - make sure both
267 # message files remain consistent).

269 @ MSG_REL_NOFIT      "relocation error: %s: file %s: symbol %s: \
270           value 0x%llx does not fit"
271 @ MSG_REL_NONALIGN   "relocation error: %s: file %s: symbol %s: \
272           offset 0x%llx is non-aligned"
273 @ MSG_REL_NULL       "relocation error: file %s: section [%u]s: \
274           skipping null relocation record"
275 @ MSG_REL_NOTSUP     "relocation error: %s: file %s: section [%u]s: \
276           relocation not currently supported"
277 @ MSG_REL_PICREDLOC "relocation error: %s: file %s symbol %s: \
278           -z relocsym may not be used for pic code"
279 @ MSG_REL_TLSLE      "relocation error: %s: file %s: symbol %s: \
280           relocation illegal when building a shared object"
281 @ MSG_REL_TLSBND     "relocation error: %s: file %s: symbol %s: \
282           bound to: %s: relocation illegal when not bound \
283           to object being created"
284 @ MSG_REL_TLSSTAT    "relocation error: %s: file %s: symbol %s: \
285           relocation illegal when building a static object"
286 @ MSG_REL_TLSBADSYM  "relocation error: %s: file %s: symbol %s: \
287           bad symbol type %s: symbol type must be TLS"
288 @ MSG_REL_BADTLS     "relocation error: %s: file %s: symbol %s: \
289           relocation illegal for TLS symbol"
290 @ MSG_REL_BADGOTBASED "relocation error: %s: file %s: symbol %s: a GOT \
291           relative relocation must reference a local symbol"
292 @ MSG_REL_UNKNWSYM   "relocation error: %s: file %s: section [%u]s: \
293           attempt to relocate with respect to unknown \
294           symbol %s: offset 0x%llx, symbol index %d"
295 @ MSG_REL_UNSUPSZ    "relocation error: %s: file %s: symbol %s: \
296           offset size (%d bytes) is not supported"
297 @ MSG_REL_INVALOFFSET "relocation error: %s: file %s section [%u]s: \
298           invalid offset symbol '%s': offset 0x%llx"
299 @ MSG_REL_INVALRELT  "relocation error: file %s: section [%u]s: \
300           invalid relocation type: 0x%x"
301 @ MSG_REL_EMPTYSEC   "relocation error: %s: file %s: symbol %s: \
302           attempted against empty section [%u]s"
303 @ MSG_REL_EXTERNSYM  "relocation error: %s: file %s: symbol %s: \
304           external symbolic relocation against non-allocatable \
305           section %s; cannot be processed at runtime: \
306           relocation ignored"
307 @ MSG_REL_UNEXPREL   "relocation error: %s: file %s: symbol %s: \
308           unexpected relocation; generic processing performed"
309 @ MSG_REL_UNEXPSSYM  "relocation error: %s: file %s: symbol %s: \
310           unexpected symbol referenced from file %s"
311 @ MSG_REL_SYMDISC    "relocation error: %s: file %s: section [%u]s: \
312           symbol %s: symbol has been discarded with discarded \
313           section: [%u]s"
314 @ MSG_REL_NOSYMBOL   "relocation error: %s: file %s: section: [%u]s: \
315           offset: 0x%llx: relocation requires reference symbol"
316 @ MSG_REL_DISPREL1   "relocation error: %s: file %s: symbol %s: \
317           displacement relocation applied to the symbol \
318           %s at 0x%llx: symbol %s is a copy relocated symbol"
319 @ MSG_REL_UNSUPSIZE   "relocation error: %s: file %s: section [%u]s: \
320           relocation against section symbol unsupported"

322 @ MSG_REL_DISPREL2   "relocation warning: %s: file %s: symbol %s: \
323           may contain displacement relocation"
324 @ MSG_REL_DISPREL3   "relocation warning: %s: file %s: symbol %s: \
325           displacement relocation applied to the symbol \

```

```

326           %s: at 0x%llx: displacement relocation will not be \
327           visible in output image"
328 @ MSG_REL_DISPREL4   "relocation warning: %s: file %s: symbol %s: \
329           displacement relocation to be applied to the symbol \
330           %s: at 0x%llx: displacement relocation will be \
331           visible in output image"
332 @ MSG_REL_COPY       "relocation warning: %s: file %s: symbol %s: \
333           relocation bound to a symbol with STV_PROTECTED \
334           visibility"
335 @ MSG_RELINVSEC     "relocation warning: %s: file %s: section: [%u]s: \
336           against suspicious section [%u]s; relocation ignored"
337 @ MSG_REL_TLSIE      "relocation warning: %s: file %s: symbol %s: \
338           relocation has restricted use when building a shared \
339           object"

341 @ MSG_REL_SLOPCDATNONAM "relocation warning: %s: file %s: section [%u]s: \
342           relocation against discarded COMDAT section [%u]s: \
343           redirected to file %s"
344 @ MSG_REL_SLOPCDATNAM "relocation warning: %s: file %s: section [%u]s: \
345           symbol %s: relocation against discarded COMDAT \
346           section [%u]s: redirected to file %s"
347 @ MSG_REL_SLOPCDATNOSYM "relocation warning: %s: file %s: section [%u]s: \
348           symbol %s: relocation against discarded COMDAT \
349           section [%u]s: symbol not found, relocation ignored"

351 @ MSG_REL_NOREG     "relocation error: REGISTER relocation not supported \
352           on target architecture"

354 #
355 # TRANSLATION_NOTE
356 #   The following 7 messages are the message to print the
357 #   following example messages.
358 #
359 #Text relocation remains
360 #   against symbol
361 #str
362 #printf
363 #
364 #   The first two lines are the header, and the next msgid
365 #   is the format string for the header.
366 #   Tabs and spaces are used for alignment.
367 #   The first and third %s are for: "Text relocation remains against symbol"
368 #   The second %s and fourth %s are for: "referenced in file"
369 #   The third %s is for: "offset"
370 #
371 @ MSG_REL_REMAIN_FMT_1 "%-40s\t%s\n    %s\t\t\t %s\t%s"
372 #
373 # TRANSLATION_NOTE
374 #   The next two msdid make a sentence. So translate:
375 #   "Text relocation remain against symbol"
376 #   And separate them into two msgstr considering the proper
377 #   alignment.
378 @ MSG_REL_RMN_ITM_11   "Text relocation remains"
379 @ MSG_REL_RMN_ITM_12   "against symbol"
380 @ MSG_REL_RMN_ITM_13   "warning: Text relocation remains"

382 @ MSG_REL_RMN_ITM_2   "offset"

384 #
385 # TRANSLATION_NOTE
386 #   The next two msdid make a sentence. So translate:
387 #   "referenced in file"
388 #   And separate them into two msgstr considering the proper
389 #   alignment.
390 @ MSG_REL_RMN_ITM_31   "referenced"
391 @ MSG_REL_RMN_ITM_32   "in file"

```

```

392 @ MSG_REL_REMAIN_2      "%-35s 0x%-8llx\t%s"
393 @ MSG_REL_REMAIN_3      "relocations remain against allocatable but \
                           non-writable sections"

396 # Files processing messages

398 @ MSG_FIL_MULINC_1      "file %s: attempted multiple inclusion of file"
399 @ MSG_FIL_MULINC_2      "file %s: linked to %s: attempted multiple inclusion \
                           of file"
400 @ MSG_FIL_SOINSTAT      "input of shared object '%s' in static mode"
402 @ MSG_FIL_INVALSEC      "file %s: section [%u]s has invalid type %s"
403 @ MSG_FIL_NOTFOUND      "file %s: required by %s, not found"
404 @ MSG_FIL_MALSTR        "file %s: section [%u]s: malformed string table, \
                           initial or final byte"
405 @ MSG_FIL_PTHTOLONG     "'%s' pathname too long"
407 @ MSG_FIL_EXCLUDE       "file %s: section [%u]s contains both SHF_EXCLUDE and \
                           SHF_ALLOC flags: SHF_EXCLUDE ignored"
409 @ MSG_FIL_INTERRUPT     "file %s: creation interrupted: %s"
410 @ MSG_FIL_INVRELOC1     "file %s: section [%u]s: relocations can not be \
                           applied against section [%u]s"
412 @ MSG_FIL_INVSHINFO     "file %s: section [%u]s: has invalid sh_info: %lld"
413 @ MSG_FIL_INVSHLINK     "file %s: section [%u]s: has invalid sh_link: %lld"
414 @ MSG_FIL_INVSHENTSIZE   "file %s: section [%u]s: has invalid sh_entsize: %lld"
415 @ MSG_FIL_NOSTRTABLE    "file %s: section [%u]s: symbol[%d]: specifies string \
                           table offset 0x%llx: no string table is available"
417 @ MSG_FIL_EXCSTRTABLE   "file %s: section [%u]s: symbol[%d]: specifies string \
                           table offset 0x%llx: exceeds string table %s: \
                           size 0x%llx"
420 @ MSG_FIL_NONAMESYM    "file %s: section [%u]s: symbol[%d]: global symbol has \
                           no name"
422 @ MSG_FIL_UNKCAP        "file %s: section [%u]s: unknown capability tag: %d"
423 @ MSG_FIL_BADSF1        "file %s: section [%u]s: unknown software \
                           capabilities: 0x%llx; ignored"
425 @ MSG_FIL_INADDR32SF1   "file %s: section [%u]s: software capability ADDR32: is \
                           ineffective when building 32-bit object; ignored"
427 @ MSG_FIL_EXADDR32SF1   "file %s: section [%u]s: software capability ADDR32: \
                           requires executable be built with ADDR32 capability"
430 @ MSG_FIL_BADORDREF    "file %s: section [%u]s: contains illegal reference \
                           to discarded section: [%u]s"

433 # Recording name conflicts

435 @ MSG_REC_OPTCNFLT    "recording name conflict: file '%s' and %s provide \
                           identical dependency names: %s"
437 @ MSG_REC_OBJCNFLT    "recording name conflict: file '%s' and file '%s' \
                           provide identical dependency names: %s %s"
439 @ MSG_REC_CNFTHINT     "(possible multiple inclusion of the same file)"

441 # System call messages

443 @ MSG_SYS_OPEN          "file %s: open failed: %s"
444 @ MSG_SYS_UNLINK        "file %s: unlink failed: %s"
445 @ MSG_SYS_MMAPANON      "mmap anon failed: %s"
446 @ MSG_SYS_MALLOC         "malloc failed: %s"

449 # Messages related to platform support

451 @ MSG_TARG_UNSUPPORTED   "unsupported ELF machine type: %s"

454 # ELF processing messages

456 @ MSG_ELF_LIBELF        "libelf: version not supported: %d"

```

```

458 @ MSG_ELF_ARMEM        "file %s: unable to locate archive member:\n\t\
                           offset=%x, symbol=%s"
459
461 @ MSG_ELF_ARSYM        "file %s ignored: unable to locate archive symbol table"
463 @ MSG_ELF_VERSYM        "file %s: version symbol section entry mismatch:\n\t\
                           (section [%u]s entries=%d; section [%u]s entries=%d)"
464
466 @ MSG_ELF_NOGROUPSECT   "file %s: section [%u]s: SHF_GROUP flag set, but no \
                           corresponding SHT_GROUP section found"
467
469 # Section processing errors
471 @ MSG_SCN_NONALLOC      "%s: non-allocatable section '%s' directed to a \
                           loadable segment: %s"
472
474 @ MSG_SCN_MULTICOMDAT   "file %s: section [%u]s: cannot be susceptible to multi \
                           COMDAT mechanisms: %s"
475
477 # Symbol processing errors
479 @ MSG_SYM_NOSECDEF      "symbol '%s' in file %s has no section definition"
480 @ MSG_SYM_INVSEC        "symbol '%s' in file %s associated with invalid \
                           section[%lld]"
481
482 @ MSG_SYM_TLS            "symbol '%s' in file %s (STT_TLS), is defined \
                           in a non-SHF_TLS section"
483
484 @ MSG_SYM_BADADDR        "symbol '%s' in file %s: section [%u]s: size %#llx: \
                           symbol (address %#llx, size %#llx) lies outside \
                           of containing section"
485
486 @ MSG_SYM_BADADDR_ROTXT  "symbol '%s' in file %s: readonly text section \
                           [%u]s: size %#llx: symbol (address %#llx, \
                           size %#llx) lies outside of containing section"
487 @ MSG_SYM_BADADDR_ROTXT  "symbol '%s' is multiply-defined"
488
489 @ MSG_SYM_MULDEF         "symbol '%s' has conflicting visibilities"
490 @ MSG_SYM_CONFVIS        "symbol '%s' has differing types"
492 @ MSG_SYM_DIFFTYPE       "symbol '%s' has differing %s:\n\t\
                           (%file %s value=0x%llx; file %s value=0x%llx);"
493 @ MSG_SYM_DIFFATTR       "symbol '%s' has differing %s:\n\t\
                           (%file %s type=%s; file %s type=%s);"
494
495 @ MSG_SYM_FILETYPES      "symbol '%s' visibility=%s; file %s visibility=%s);"
496 @ MSG_SYM_VISTYPES       "\t%s definition taken"
497 @ MSG_SYM_DEFTAKEN       "\t%s definition taken and updated with larger size"
498 @ MSG_SYM_DEFUPDATE      "\tlargest value applied"
499 @ MSG_SYM_LARGER         "\tentative symbol cannot override defined symbol \
                           of smaller size"
500 @ MSG_SYM_TENTERR        "symbol %s has invalid section index; \
                           ignored:\n\t(file %s value=%s);"
501
503 @ MSG_SYM_INVSHNDX      "global symbol %s has non-global binding:\n\t\
                           (%file %s value=%s);"
504
505 @ MSG_SYM_NONGLOB        "reserved symbol '%s' already defined in file %s"
506
507 @ MSG_SYM_RESERVE        "undefined symbol '%s' with non-zero value encountered \
                           from file %s"
508 @ MSG_SYM_NOTNULL        "section %s: symbol '%s' and symbol '%s' have the \
                           same address: %#llx: remove duplicate with \
                           NOSORTSYM mapfile directive"
509
510 @ MSG_SYM_DUPSORTADDR   "file %s: section [%u]s: entry[%d] has invalid m_info: \
                           0x%llx for symbol index"
511
512
514 @ MSG_PSYM_INVMINFO1    "file %s: section [%u]s: entry[%d] has invalid m_info: \
                           0x%llx for size"
515
516 @ MSG_PSYM_INVMINFO2    "file %s: section [%u]s: entry[%d] has invalid m_info: \
                           0x%llx for size"
517
518 @ MSG_PSYM_INVMREPEAT   "file %s: section [%u]s: entry[%d] has invalid m_repeat \
                           0x%llx"
519
520 @ MSG_PSYM_CANNOTEXPND  "file %s: section [%u]s: entry[%d] can not be expanded: \
                           associated symbol size is unknown %s"
521
522 @ MSG_PSYM_NOSTATIC     "and partial initialization cannot be deferred to \
                           a static object"
523

```

```

524 @ MSG_MOVE_OVERLAP      "file %s: section [%u] %s: symbol '%s' overlapping move \
525                                initialization: start=0x%llx, length=0x%llx: \
526                                start=0x%llx, length=0x%llx"
527 @ MSG_PSYM_EXPREASON1   "output file is static object"
528 @ MSG_PSYM_EXPREASON2   "-z nopartial option in effect"
529 @ MSG_PSYM_EXPREASON3   "move infrastructure size is greater than move data"

531 #
532 # Support library failures
533 #
534 @ MSG_SUP_NOLOAD        "dlopen() of support library (%s) failed with \
535                                error: %s"
536 @ MSG_SUP_BADVERSION    "initialization of support library (%s) failed with \
537                                bad version. supported: %d returned: %d"

540 #
541 # TRANSLATION_NOTE
542 #     The following 7 messages are the message to print the
543 #     following example messages.
544 #
545 #Undefined                  first referenced
546 # symbol                   in file
547 #inquire                   halt_hold.o
548 #
549 @ MSG_SYM_FMT_UNDEF      "%s\t\t\t%s\
550          \n %s \t\t\t %s"

552 #
553 # TRANSLATION_NOTE
554 #     The next two msdid make a sentence. So translate:
555 #         "Undefined symbol"
556 #     And separate them into two msgstr considering the proper
557 #     alignment.
558 @ MSG_SYM_UNDEF_ITM_11   "Undefined"
559 @ MSG_SYM_UNDEF_ITM_12   "symbol"
560 #
561 # TRANSLATION_NOTE
562 #     The next two msdid make a sentence. So translate:
563 #         "first referenced in file"
564 #     And separate them into two msgstr considering the proper
565 #     alignment.
566 @ MSG_SYM_UNDEF_ITM_21   "first referenced"
567 @ MSG_SYM_UNDEF_ITM_22   "in file"
568 #

570 @ MSG_SYM_UND_UNDEF      "%-35s %s"
571 @ MSG_SYM_UND_NEVER      "%-35s %s (symbol has no version assigned)"
572 @ MSG_SYM_UND_IMPL       "%-35s %s (symbol belongs to implicit dependency %s)"
573 @ MSG_SYM_UND_NOTA       "%-35s %s (symbol belongs to unavailable version %s \
574                                (%s))"
575 @ MSG_SYM_UND_BNDLOCAL   "%-35s %s (symbol scope specifies local binding)"

577 @ MSG_SYM_ENTRY          "entry point"
578 @ MSG_SYM_UNDEF          "%s symbol '%s' is undefined"
579 @ MSG_SYM_EXTERN          "%s symbol '%s' is undefined (symbol belongs to \
580                                dependency %s)"
581 @ MSG_SYM_NOCRT          "symbol '%s' not found, but %s section exists - \
582                                possible link-edit without using the compiler driver"

584 # Output file update messages

586 @ MSG_UPD_NOREADSEG     "No read-only segments found. Setting '_etext' to 0"
587 @ MSG_UPD_NORDWRSEGP    "No read-write segments found. Setting '_edata' to 0"
588 @ MSG_UPD_NOSEG          "Setting 'end' and '_end' to 0"

```

```

590 @ MSG_UPD_SEGOVERLAP    "%s: segment address overlap:\n \
591                                \tprevious segment ending at address 0x%llx overlaps\n \
592                                \tuser defined segment '%s' starting at address 0x%llx"
593 @ MSG_UPD_LARGSIZE       "%s: segment %s calculated size 0x%llx\n \
594                                \tis larger than user-defined size 0x%llx"

596 @ MSG_UPD_NOBITS         "NOBITS section found before end of initialized data"
597 @ MSG_SEG_FIRNOTLOAD    "First segment has type %s, PT_LOAD required: %s"
598 @ MSG_UPD_MULEHFRAME    "file %s: section [%u] %s and file %s: section [%u] %s \
599                                have incompatible attributes and cannot \
600                                be merged into a single output section"

603 # Version processing messages

605 @ MSG_VER_HIGHER         "file %s: version revision %d is higher than \
606                                expected %d"
607 @ MSG_VER_NOEXIST        "file %s: version '%s' does not exist:\n \
608                                \trequired by file %s"
609 @ MSG_VER_UNDEF           "version '%s' undefined, referenced by version '%s':\n \
610                                \trequired by file %s"
611 @ MSG_VER_UNAVAIL        "file %s: version '%s' is unavailable:\n \
612                                \trequired by file %s"
613 @ MSG_VER_DEFINED         "version symbol '%s' already defined in file %s"
614 @ MSG_VER_INVALNDX       "version symbol '%s' from file %s has an invalid \
615                                version index (%d)"
616 @ MSG_VER_ADDVERS        "unused $ADDVERS specification from file '%s' \
617                                for object '%s'\nversion(s):"
618 @ MSG_VER_ADDVER         "\t%s"
619 @ MSG_VER_CYCLIC         "following versions generate cyclic dependency:"

621 # Capabilities messages

623 @ MSG_CAP_MULDEF         "capabilities symbol '%s' has multiply-defined members: \
624 @ MSG_CAP_MULDEFSYMS     "\t(file %s symbol '%s'; file %s symbol '%s');"
625 @ MSG_CAP_REDUNDANT      "file %s: section [%u] %s: symbol capabilities \
626                                redundant, as object capabilities are more restrictive"
627 @ MSG_CAP_NOSYMSFOUND    "no global symbols have been found that are associated \
628                                with capabilities identified relocatable objects: \
629                                -z symbolcap has no effect"

631 @ MSG_CAPINFO_INVALSYM   "file %s: capabilities info section [%u] %s: index %d: \
632                                \tfamily member symbol '%s': invalid"
633 @ MSG_CAPINFO_INVALIDLEAD "file %s: capabilities info section [%u] %s: index %d: \
634                                \tfamily lead symbol '%s': invalid symbol index %d"

636 # Basic strings

638 @ MSG_STR_ALIGNMENTS     "alignments"
639 @ MSG_STR_COMMAND        "(command line)"
640 @ MSG_STR_TLSREL          "(internal TLS relocation requirement)"
641 @ MSG_STR_SIZES           "sizes"
642 @ MSG_STR_UNKNOWN         "<unknown>"
643 @ MSG_STR_SECTION         "%s (section)"
644 @ MSG_STR_SECTION_MSTR   "%s (merged string section)"

646 #
647 # TRANSLATION_NOTE
648 #     The elf_function name represents a man page reference and should not
649 #     be translated.
650 @ MSG_ELF_BEGIN            "file %s: elf_begin"
651 @ MSG_ELF_CNTL             "file %s: elf_cntl"
652 @ MSG_ELF_GETARHDR         "file %s: elf_getarhdr"
653 @ MSG_ELF_GETARSYM         "file %s: elf_getarsym"
654 @ MSG_ELF_GETDATA          "file %s: elf_getdata"
655 @ MSG_ELF_GETEHDR          "file %s: elf_getehdr"

```

```

656 @ MSG_ELF_GETPHDR      "file %s: elf_getphdr"
657 @ MSG_ELF_GETSCN       "file %s: elf_getscn: scnndx: %d"
658 @ MSG_ELF_GETSHDR      "file %s: elf_getshdr"
659 @ MSG_ELF_MEMORY        "file %s: elf_memory"
660 @ MSG_ELF_NDXSCN       "file %s: elf_ndxscn"
661 @ MSG_ELF_NEWDATA      "file %s: elf_newdata"
662 @ MSG_ELF_NEWEHDR      "file %s: elf_newehdr"
663 @ MSG_ELF_NEWSCN       "file %s: elf_newscn"
664 @ MSG_ELF_NEWPHDR      "file %s: elf_newphdr"
665 @ MSG_ELF_STRPTR       "file %s: elf_strptr"
666 @ MSG_ELF_UPDATE        "file %s: elf_update"
667 @ MSG_ELF_SWAP_WRIMAGE "file %s: _elf_swap_wrimage"

670 @ MSG_REJ_MACH          "file %s: wrong ELF machine type: %s"
671 @ MSG_REJ_CLASS         "file %s: wrong ELF class: %s"
672 @ MSG_REJ_DATA          "file %s: wrong ELF data format: %s"
673 @ MSG_REJ_TYPE          "file %s: bad ELF type: %s"
674 @ MSG_REJ_BADFLAG       "file %s: bad ELF flags value: %s"
675 @ MSG_REJ_MISFLAG       "file %s: mismatched ELF flags value: %s"
676 @ MSG_REJ_VERSION       "file %s: mismatched ELF/lib version: %s"
677 @ MSG_REJ_HAL            "file %s: HAL R1 extensions required"
678 @ MSG_REJ_US3            "file %s: Sun UltraSPARC III extensions required"
679 @ MSG_REJ_STR            "file %s"
680 @ MSG_REJ_UNKFILE       "file %s: unknown file type"
681 @ MSG_REJ_UNKCAP         "file=%s: unknown capability: %d"
682 @ MSG_REJ_HWCAP_1        "file %s: hardware capability (CA_SUNW_HW_1) \
683                                unsupported: %s"
684 @ MSG_REJ_SFCAP_1        "file %s: software capability (CA_SUNW_SF_1) \
685                                unsupported: %s"
686 @ MSG_REJ_MACHCAP       "file %s: machine capability (CA_SUNW_MACH) \
687                                unsupported: %s"
688 @ MSG_REJ_PLATCAP       "file %s: platform capability (CA_SUNW_PLAT) \
689                                unsupported: %s"
690 @ MSG_REJ_HWCAP_2        "file %s: hardware capability (CA_SUNW_HW_2) \
691                                unsupported: %s"
692 @ MSG_REJ_ARCHIVE        "file %s: invalid archive use"

694 # Guidance messages
695 @ MSG_GUIDE_SUMMARY     "see ld(1) -z guidance for more information"
696 @ MSG_GUIDE_DEFS        "-z defs option recommended for shared objects"
697 @ MSG_GUIDE_DIRECT       "-B direct or -z direct option recommended before \
698                                first dependency"
699 @ MSG_GUIDE_LAZYLOAD     "-z lazyload option recommended before \
700                                first dependency"
701 @ MSG_GUIDE_MAPFILE      "version 2 mapfile syntax recommended: %s"
702 @ MSG_GUIDE_TEXT         "position independent (PIC) code recommended for \
703                                shared objects"
704 @ MSG_GUIDE_UNUSED       "removal of unused dependency recommended: %s"

706 @_END_

```

709 # The following strings represent reserved names. Reference to these strings
710 # is via the MSG_ORIG() macro, and thus translations are not required.

```

712 @ MSG_STR_EOF           "<eof>"
713 @ MSG_STR_ERROR         "<error>"
714 @ MSG_STR_EMPTY         ""
715 @ MSG_QSTR_BANG         "'!''"
716 @ MSG_STR_COLON         ":""
717 @ MSG_QSTR_COLON        "'::''"
718 @ MSG_QSTR_SEMICOLON    "'='''"
719 @ MSG_QSTR_EQUAL         "'+='''"
720 @ MSG_QSTR_PLUSEQ        "'-='''"

```

```

722 @ MSG_QSTR_ATSIGN      "'@''"
723 @ MSG_QSTR_DASH         "'-'"
724 @ MSG_QSTR_LEFTBKT     "'{'"
725 @ MSG_QSTR_RIGHTBKT    "'}'"
726 @ MSG_QSTR_PIPE         "'|''"
727 @ MSG_QSTR_STAR         "'*'"
728 @ MSG_STR_DOT           "'.'"
729 @ MSG_STR_SLASH         "'/'"
730 @ MSG_STR_DYNAMIC       "(.dynamic)"
731 @ MSG_STR_ORIGIN         "$ORIGIN"
732 @ MSG_STR_MACHINE        "$MACHINE"
733 @ MSG_STR_PLATFORM       "$PLATFORM"
734 @ MSG_STR_ISALIST       "$ISALIST"
735 @ MSG_STR_OSNAME        "$OSNAME"
736 @ MSG_STR_OSREL         "$OSREL"
737 @ MSG_STR_UU_REAL_U     "__real__"
738 @ MSG_STR_UU_WRAP_U     "__wrap__"
739 @ MSG_STR_UELF32         "_ELF32"
740 @ MSG_STR_UELF64         "_ELF64"
741 @ MSG_STR_USPARC        "_sparc"
742 @ MSG_STR_UX86           "_x86"
743 @ MSG_STR_TRUE           "true"

745 @ MSG_STR_CDIR_ADD      "$add"
746 @ MSG_STR_CDIR_CLEAR    "$clear"
747 @ MSG_STR_CDIR_ERROR    "$error"
748 @ MSG_STR_CDIR_MFVER    "$mapfile_version"
749 @ MSG_STR_CDIR_IF        "$if"
750 @ MSG_STR_CDIR_ELIF     "$elif"
751 @ MSG_STR_CDIR_ELSE     "$else"
752 @ MSG_STR_CDIR_ENDIF    "$endif"

754 @ MSG_STR_GROUP         "GROUP"
755 @ MSG_STR_SUNW_COMDAT   "SUNW_COMDAT"

757 @ MSG_FMT_ARMEM         "%s(%s)"
758 @ MSG_FMT_COLPATH       "%s:%s"
759 @ MSG_FMT_SYMNAM        "%s[%d]"
760 @ MSG_FMT_NULLSYNMAM    "%s%s"
761 @ MSG_FMT_STRCAT        "%s%s"

763 @ MSG_PTH_RTLD          "/usr/lib/ld.so.1"

765 @ MSG_SUNW_OST_SGS      "SUNW_OST_SGS"

768 # Section strings

770 @ MSG_SCN_BSS           ".bss"
771 @ MSG_SCN_DATA          ".data"
772 @ MSG_SCN_COMMENT        ".comment"
773 @ MSG_SCN_DEBUG          ".debug"
774 @ MSG_SCN_DEBUG_INFO     ".debug_info"
775 @ MSG_SCN_DYNAMIC        ".dynamic"
776 @ MSG_SCN_DYNNSMORT     ".SUNW_dynnsmort"
777 @ MSG_SCN_DYNTLSSORT    ".SUNW_dyntlssort"
778 @ MSG_SCN_DYNSTR         ".dynstr"
779 @ MSG_SCN_DYNSYM         ".dynsym"
780 @ MSG_SCN_DYNNSYM_SHNDX ".dynsym_shndx"
781 @ MSG_SCN_LDYNNSYM      ".SUNW_ldynsym"
782 @ MSG_SCN_LDYNNSYM_SHNDX ".SUNW_ldynsym_shndx"
783 @ MSG_SCN_EX_SHARED      ".ex_shared"
784 @ MSG_SCN_EX_RANGES      ".exception_ranges"
785 @ MSG_SCN_EXCL           ".excl"
786 @ MSG_SCN_FINI           ".fini"
787 @ MSG_SCN_FINIARRAY      ".fini_array"

```

```

788 @ MSG_SCN_GOT      ".got"
789 @ MSG_SCN_GNU_LINKONCE ".gnu.linkonce."
790 @ MSG_SCN_HASH      ".hash"
791 @ MSG_SCN_INDEX     ".index"
792 @ MSG_SCN_INIT      ".init"
793 @ MSG_SCN_INITARRAY ".init_array"
794 @ MSG_SCN_INTERP   ".interp"
795 @ MSG_SCN_LBSS      ".lbss"
796 @ MSG_SCN_LDATA     ".ldata"
797 @ MSG_SCN_LINE      ".line"
798 @ MSG_SCN_LRODATA   ".lrodata"
799 @ MSG_SCN_PLT       ".plt"
800 @ MSG_SCN_PREINITARRAY ".preinit_array"
801 @ MSG_SCN_REL       ".rel"
802 @ MSG_SCN_RELA      ".rela"
803 @ MSG_SCN_RODATA   ".rodata"
804 @ MSG_SCN_SBSS     ".sbss"
805 @ MSG_SCN_SBSS2    ".sbss2"
806 @ MSG_SCN_SDATA    ".sdata"
807 @ MSG_SCN_SDATA2   ".sdata2"
808 @ MSG_SCN_SHSTRTAB ".shstrtab"
809 @ MSG_SCN_STAB     ".stab"
810 @ MSG_SCN_STABEXCL ".stab.exclstr"
811 @ MSG_SCN_STRTAB   ".sttab"
812 @ MSG_SCN_SUNWMOVE ".SUNW_move"
813 @ MSG_SCN_SUNWRELOC ".SUNW_reloc"
814 @ MSG_SCN_SUNWSYMINFO ".SUNW_syminfo"
815 @ MSG_SCN_SUNWVERSION ".SUNW_version"
816 @ MSG_SCN_SUNWVERSYM ".SUNW_versym"
817 @ MSG_SCN_SUNWCAP ".SUNW_cap"
818 @ MSG_SCN_SUNWCAPINFO ".SUNW_capinfo"
819 @ MSG_SCN_SUNWCAPCHAIN ".SUNW_capchain"
820 @ MSG_SCN_SYMTAB   ".syntab"
821 @ MSG_SCN_SYMTAB_SHNDX ".syntab_shndx"
822 @ MSG_SCN_TBSS     ".tbss"
823 @ MSG_SCN_TDATA    ".tdata"
824 @ MSG_SCN_TEXT     ".text"

826 @ MSG_SYM_FINIARRAY "finiarrray"
827 @ MSG_SYM_INITARRAY "initarray"
828 @ MSG_SYM_PREINITARRAY "preinitarray"

830 #
831 # GNU section names
832 #
833 @ MSG_SCN_CTORS    ".ctors"
834 @ MSG_SCN_DTORS    ".dtors"
835 @ MSG_SCN_EHFRAME  ".eh_frame"
836 @ MSG_SCN_EHFRAME_HDR ".eh_frame_hdr"
837 @ MSG_SCN_GCC_X_TBL ".gcc_except_table"
838 @ MSG_SCN_JCR      ".jcr"

840 # Segment names for segments referenced by entrance criteria

842 @ MSG_ENT_BSS      "bss"
843 @ MSG_ENT_DATA     "data"
844 @ MSG_ENT_EXTRA    "extra"
845 @ MSG_ENT_LDATA    "ldata"
846 @ MSG_ENT_LRODATA  "lrodata"
847 @ MSG_ENT_NOTE     "note"
848 @ MSG_ENT_TEXT     "text"

850 # Symbol names

852 @ MSG_SYM_START    "_start"
853 @ MSG_SYM_MAIN     "main"

```

```

855 @ MSG_SYM_FINI_U     "_fini"
856 @ MSG_SYM_INIT_U     "_init"
857 @ MSG_SYM_DYNAMIC     "DYNAMIC"
858 @ MSG_SYM_DYNAMIC_U   "_DYNAMIC"
859 @ MSG_SYM_EDATA       "edata"
860 @ MSG_SYM_EDATA_U     "_edata"
861 @ MSG_SYM_END         "end"
862 @ MSG_SYM_END_U       "_end"
863 @ MSG_SYM_ETEXT       "etext"
864 @ MSG_SYM_ETEXT_U     "_etext"
865 @ MSG_SYM_GOFTBL     "GLOBAL_OFFSET_TABLE_"
866 @ MSG_SYM_GOFTBL_U   "_GLOBAL_OFFSET_TABLE_"
867 @ MSG_SYM_PLKTBBL    "PROCEDURE_LINKAGE_TABLE_"
868 @ MSG_SYM_PLKTBBL_U   "_PROCEDURE_LINKAGE_TABLE_"
869 @ MSG_SYM_TLSGETADDR_U "__tls_get_addr"
870 @ MSG_SYM_TLSGETADDR_UU "__tls_get_addr"

872 @ MSG_SYM_L_END       "END_"
873 @ MSG_SYM_L_END_U     "_END_"
874 @ MSG_SYM_L_START     "START_"
875 @ MSG_SYM_L_START_U   "_START_"

877 # Support functions
878 @ MSG_SUP_VERSION     "ld_version"
879 @ MSG_SUP_INPUT_DONE  "ld_input_done"
880 @ MSG_SUP_START_64    "ld_start64"
881 @ MSG_SUP_ATEXIT_64   "ld_atexit64"
882 @ MSG_SUP_OPEN_64     "ld_open64"
883 @ MSG_SUP_FILE_64    "ld_file64"
884 @ MSG_SUP_INSEC_64   "ld_input_section64"
885 @ MSG_SUP_SEC_64     "ld_section64"
886 @ MSG_SUP_START       "ld_start"
887 @ MSG_SUP_ATEXIT     "ld_atexit"
888 @ MSG_SUP_OPEN        "ld_open"
889 @ MSG_SUP_FILE        "ld_file"
890 @ MSG_SUP_INSEC       "ld_input_section"
891 @ MSG_SUP_SEC         "ld_section"

896 #
897 # Message previously in 'ld'
898 #
899 #
900 @ _START_

902 # System error messages
903 @ MSG_SYS_STAT        "file %s: stat failed: %s"
904 @ MSG_SYS_READ        "file %s: read failed: %s"
905 @ MSG_SYS_NOTREG      "file %s: is not a regular file"
906 @ MSG_SYS_NOTREG      "file %s: is not a regular file"

908 # Argument processing messages
909 @ MSG_ARG_DY_INCOMP   "%s option is incompatible with building a dynamic \
                           executable"
910 @ MSG_MARG_DY_INCOMP  "%s is incompatible with building a dynamic \
                           executable"
911 @ MSG_ARG_ST_INCOMP   "%s option is incompatible with building a static \
                           object (-dn, -r, --relocatable)"
912 @ MSG_MARG_ST_INCOMP  "%s is incompatible with building a static \
                           object (-dn, -r, --relocatable)"
913 @ MSG_ARG_INCOMP      "%s option is incompatible with building a shared object"
914 @ MSG_MARG_ST_ONLYAVL "%s is only available when building a shared object"
915 @ MSG_ARG_INCOMP      "option %s and %s are incompatible"
916 @ MSG_MARG_ST_INCOMP  "%s is incompatible with building a static \
                           object (-dn, -r, --relocatable)"
917 @ MSG_ARG_INCOMP      "%s option is incompatible with building a shared object"
918 @ MSG_MARG_ST_INCOMP  "%s is incompatible with building a static \
                           object (-dn, -r, --relocatable)"
919 @ MSG_ARG_INCOMP      "option %s and %s are incompatible"

```

new/usr/src/cmd/sgs/libld/common/libld.msg

15

```

920 @ MSG_MARG_INCOMP      "%s and %s are incompatible"
921 @ MSG_ARG_MTONCE       "option %s appears more than once, first setting taken"
922 @ MSG_MARG_MTONECE     "%s appears more than once, first setting taken"
923 @ MSG_ARG_ILLEGAL      "option %s has illegal argument '%s'"
924 @ MSG_ARG_YP          "option -YP and -Y%c may not be specified concurrently"
925 @ MSG_ARG_STRIP        "%s is specified with %s; only debugging \
                           information stripped"
926 @ MSG_ARG_NOFILES      "no files on input command line"
928 @ MSG_ARG_NOFLTR       "option %s is only meaningful when building a filter"
929 @ MSG_ARG_NODEFLIB     "the default library search path has been suppressed, \
                           but no runpaths have been specified via %s"
930 @ MSG_ARG_NOENTRY      "entry point symbol '%s' is undefined"
932 @ MSG_ARG_UNSUPPORTED  "option %s is no longer supported; ignored"
933 @ MSG_MARG_ONLY         "option %s can only be used with a %s"
934 @ MSG_ARG_UNKNOWN      "unrecognized option '-%c'"
935 @ MSG_ARG_LONG_UNKNOWN "unrecognized option '%s'"
936 #endif /* ! codereview */
937 @ MSG_ARG_USEHELP      "use the -z help option for usage information"

940 @ MSG_ARG_FLAGS         "flags processing errors"
941 @ MSG_ARG_FILES         "file processing errors. No output written to %s"
942 @ MSG_ARG_SYM_WARN      "symbol referencing errors"
943 @ MSG_ARG_SYM_FATAL    "symbol referencing errors. No output written to %s"
944 @ MSG_ARG_AR_GRP_OLAP   "%s cannot be nested"
945 @ MSG_ARG_AR_GRP_BAD    "%s used without corresponding %s"

948 # Messages used to refer to options where there is more than
949 # one name accepted.

951 @ MSG_MARG_AR_GRPS     "archive rescan groups \
                           (-z rescan-start, -(, --start-group))"
952 @ MSG_MARG_AR_GRP_END   "archive rescan group end option \
                           (-z rescan-end, -, --end-group)"
954 @ MSG_MARG_AR_GRP_START "archive rescan group start option \
                           (-z rescan-start, -(, --start-group))"
955 @ MSG_MARG_AR_GRP_ENTRY "entry point option (-e, --entry)"
956 @ MSG_MARG_FILTER_AUX   "auxiliary filter option (-f, --auxiliary)"
957 @ MSG_MARG_FILTER       "filter option (-F, --filter)"
958 @ MSG_MARG_OUTFILE      "output object option (-o, --output)"
959 @ MSG_MARG_REL          "relocatable object option (-r, --relocatable)"
960 @ MSG_MARG_RPATH        "runpath option (-R, -rpath)"
961 @ MSG_MARG_SO           "shared object option (-G, -shared)"
962 @ MSG_MARG_SONAME       "soname option (-h, --soname)"
963 @ MSG_MARG_STRIP        "strip option (-s, --strip-all)"

967 # Entrance criteria messages

969 @ MSG_ENT_MAP_FMT_TIL_1 "\t\t%s\n\n"
970 @ MSG_ENT_MAP_TITLE_1   "LINK EDITOR MEMORY MAP"

972 #
973 # TRANSLATION_NOTE -- Entry map header
974 #
975 # The next message is a format string for a title. The title is composed of
976 # two lines. In C locale, it would look like:
977 #
978 #      output      input      new
979 #      section     section    displacement    size
980 #
981 # The \t characters are used for alignment. (output section), (input section),
982 # and (new displacement) have to be aligned.
983 #
984 @ MSG_ENT_MAP_FMT_TIL_2 "\n%$s\t$t%$s\t$t%$s\n%$s\t$t%$s\t$t%$s\t$t%$s\n"
985 @ MSG_ENT_MAP_FMT_TIL_3 "\n%$s\t$t%$s\t$t%$s\n%$s\t$t%$s\t$t%$s\t$t%$s\n"

```

new/usr/src/cmd/sgs/libld/common/libld.msg

new/usr/src/cmd/sgs/libld/common/libld.msg

17

```

1052 @ MSG_MAP_BADONAME
1053
1054 @ MSG_MAP_REDEFATT
1055 @ MSG_MAP_PREAMEOF
1056 @ MSG_MAP_ILLCHAR
1057 @ MSG_MAP_MALFORM
1058 @ MSG_MAP_NONLOAD
1059 @ MSG_MAP_NOSTACK1
1060 @ MSG_MAP_MORENCE
1061 @ MSG_MAP_NOTERM
1062 @ MSG_MAP_SECINSEG
1063
1064 @ MSG_MAP_UNEXINHERIT
1065
1066 @ MSG_MAP_UNEXTOK

1068 @ MSG_MAP_SEGEMUPLOAD
1069 @ MSG_MAP_SEGEMPEXE
1070
1071 @ MSG_MAP_SEGEMPATT
1072
1073 @ MSG_MAP_SEGEMPNOATT
1074
1075 @ MSG_MAP_SEGEMPSEC
1076
1077 @ MSG_MAP_SEGEMNOPERM
1078

1080 @ MSG_MAP_CNTADDRORDER
1081
1082 @ MSG_MAP_CNTDISSEG
1083 @ MSG_MAP_DUPNAMENT
1084 @ MSG_MAP_DUPORDSEG
1085 @ MSG_MAP_DUP_OS_ORD
1086 @ MSG_MAP_DUP_IS_ORD
1087
1088 @ MSG_MAP_UNKENT
1089
1090 @ MSG_MAP_UNKSEG
1091 @ MSG_MAP_UNKSYMDEF
1092 @ MSG_MAP_UNKSEGTYP
1093 @ MSG_MAP_UNKSOTYP
1094 @ MSG_MAP_UNKSEGATT
1095 @ MSG_MAP_UNKSEGFLG
1096 @ MSG_MAP_UNKSECTYP

1098 @ MSG_MAP_SEGSIZE
1099
1100 @ MSG_MAP_SEGADDR
1101 @ MSG_MAP_BADCAPVAL
1102 @ MSG_MAP_UNKCAPATTR
1103 @ MSG_MAP_EMPTYCAP

1105 @ MSG_MAP_SYMDEF1
1106
1107 @ MSG_MAP_SYMDEF2

1109 @ MSG_MAP_EXPSCOL
1110 @ MSG_MAP_EXPEQU
1111 @ MSG_MAP_EXPSEGATT
1112
1113 @ MSG_MAP_EXPSGNAM
1114
1115 @ MSG_MAP_EXPSGEGTYPE
1116
1117 @ MSG_MAP_EXPSYM_1

```

"%s: %llu: object name cannot contain path \ separator ('/'): %" "%s: %llu: redefining %s attribute for '%s'" "%s: %llu: premature EOF" "%s: %llu: illegal character '\\x030'" "%s: %llu: malformed entry" "%s: %llu: %s not allowed on non-LOAD segments" "%s: %llu: %s not allowed on STACK segment" "%s: %llu: %s set more than once on same line" "%s: %llu: unterminated quoted string: %s" "%s: %llu: section within segment ordering done on \ a non-existent segment '%s'" "%s: %llu: unnamed version cannot inherit from other versions: %s" "%s: %llu: unexpected occurrence of '%c' token"

"%s: %llu: empty segment must be of type LOAD or NULL" "%s: %llu: a LOAD empty segment definition is only \ allowed when creating a dynamic executable" "%s: %llu: a LOAD empty segment must have an address \ and size" "%s: %llu: a NULL empty segment must not have an \ address or size" "%s: %llu: empty segment can not have sections \ assigned to it" "%s: %llu: empty segment must not have \ p_flags set: 0x%x"

"%s: %llu: segment cannot have an explicit address \ and also be in the SEGMENT ORDER list: %s" "%s: %llu: segment cannot be disabled: %s" "%s: %llu: cannot redefine entrance criteria: %s" "%s: %llu: segment is already in %s list: %s" "%s: %llu: section is already in OS_ORDER list: %s" "%s: %llu: entrance criteria is already in \ IS_ORDER list: %s" "%s: %llu: unknown entrance criteria \ (ASSIGN_SECTION): %s" "%s: %llu: unknown segment: %s" "%s: %llu: unknown symbol definition: %s" "%s: %llu: unknown internal segment type %d" "%s: %llu: unknown shared object type: %s" "%s: %llu: unknown segment attribute: %s" "%s: %llu: unknown segment flag: %c" "%s: %llu: unknown section type: %s"

"%s: %lld: existing segment size symbols cannot \ be reset: %s" "%s: %llu: segment address or length '%s' %s" "%s: %llu: bad capability value: %s" "%s: %llu: unknown capability attribute %s" "%s: %llu: empty capability definition; ignored"

"%s: %llu: symbol '%s' is already defined in file: \ %s" "%s: %llu: symbol '%s': %s"

"%s: %llu: expected a ';'%" "%s: %llu: expected a '=' , ':", '|', or '@'" "%s: %llu: expected one or more segment attributes \ after an '='%" "%s: %llu: expected a segment name at the beginning \ of a line" "%s: %llu: %s segment cannot be used with %s \ directive: %s" "%s: %llu: expected a symbol name after '@'"

new/usr/src/cmd/sgs/libld/common/libld.msc

```

1118 @ MSG_MAP_EXPSYM_2
1119 @ MSG_MAP_EXPSEC
1120 @ MSG_MAP_EXPSO
1121
1122 @ MSG_MAP_MULTFILTEE
1123 @ MSG_MAP_NOFILTER
1124 @ MSG_MAP_BADSF1
1125
1126 @ MSG_MAP_INADDR32SF1
1127
1128 @ MSG_MAP_NOINTPOSE
1129
1130 @ MSG_MAP_NOEXVLSZ
1131
1132 @ MSG_MAP_FLTR_ONLYAVL
1133

1135 @ MSG_MAP_SEGSAME
1136
1137 @ MSG_MAP_EXCLIMIT
1138 @ MSG_MAP_NOBADFRM

1140 @ MSG_MAP_SEGTYP
1141 @ MSG_MAP_SEGVADDR
1142 @ MSG_MAP_SEGPHYS
1143 @ MSG_MAP_SEGLEN
1144 @ MSG_MAP_SEGFLAG
1145 @ MSG_MAP_SEGALIGN
1146 @ MSG_MAP_SEGROUND

1148 @ MSG_MAP_SECTYP
1149 @ MSG_MAP_SECFLAG
1150 @ MSG_MAP_SECNAME

1152 @ MSG_MAP_SYMVAL
1153 @ MSG_MAP_SYMSIZE

1155 @ MSG_MAP_DIFF_SYMVAL
1156 @ MSG_MAP_DIFF_SYMSZ
1157 @ MSG_MAP_DIFF_SYMTYP
1158 @ MSG_MAP_DIFF_SYMNDX
1159 @ MSG_MAP_DIFF_SYMLCL
1160 @ MSG_MAP_DIFF_SYMGLOB
1161 @ MSG_MAP_DIFF_SYMPROT
1162 @ MSG_MAP_DIFF_SYMVER
1163 @ MSG_MAP_DIFF_SYMMUL
1164 @ MSG_MAP_DIFF_SNGLDIR
1165
1166 @ MSG_MAP_DIFF_PROTNDIR
1167

1170 @ MSG_MAP_SECORDER
1171

1174 # Mapfile Directives

1176 @ MSG_MAP_EXP_ATTR
1177
1178 @ MSG_MAP_EXP_CAPMASK
1179
1180 @ MSG_MAP_EXP_CAPNAME
1181 @ MSG_MAP_EXP_CAPID
1182 @ MSG_MAP_EXP_CAPHW
1183

    "%s: %llu: expected a symbol name after '{'"
    "%s: %llu: expected a section name after '|'"
    "%s: %llu: expected a shared object definition \
after '-'"
    "%s: %llu: multiple filtee definitions are unsupported"
    "%s: %llu: filtee definition required"
    "%s: %llu: unknown software capabilities: 0x%llx; \
ignored"
    "%s: %llu: software capability ADDR32: is ineffective \
when building 32-bit object: ignored"
    "%s: %llu: interposition symbols can only be defined \
when building a dynamic executable"
    "%s: %llu: value and size attributes are incompatible \
with extern or parent symbols"
    "%s: %llu: symbol filtering is only available when \
building a shared object"

    "segments '%s' and '%s' have the same assigned \
virtual address"
    "exceeds internal limit"
    "number is badly formed"

    "segment type"
    "segment virtual address"
    "segment physical address"
    "segment length"
    "segment flags"
    "segment alignment"
    "segment rounding"

    "section type"
    "section flags"
    "section name"

    "symbol value"
    "symbol size"

    "symbol values differ"
    "symbol sizes differ"
    "symbol types differ"
    "symbol indexes differ"
    "symbol scope conflict against local and non-local"
    "symbol scope conflict against singleton/exported"
    "symbol scope conflict against protected"
    "symbol version conflict"
    "symbol multiple definition"
    "singleton scope and direct declaration are \
incompatible"
    "protected scope and no-direct declaration \
are incompatible"

    "section ordering requested, but no matching section \
found: segment: %s section: %s"

    "%s: %llu: expected attribute name (%s), or \
terminator (';', ','): %s"
    "%s: %llu: expected capability name, integer value, or \
terminator (';', ','): %s"
    "%s: %llu: expected name, or terminator (';', ','): %s"
    "%s: %llu: expected name, or '{' following %s: %s"
    "%s: %llu: expected hardware capability, or \
terminator (';', ','): %s"

```

```

1184 @ MSG_MAP_EXP_CAPSF    "%s: %llu: expected software capability, or \
1185                                terminator ('', ','): %s"
1186 @ MSG_MAP_EXP_EQ        "%s: %llu: expected '=' following %s: %s"
1187 @ MSG_MAP_EXP_EQ_ALL    "%s: %llu: expected '=', '+', or '-' following %s: %s"
1188 @ MSG_MAP_EXP_EQ_PEQ    "%s: %llu: expected '=' following %s: %s"
1189 @ MSG_MAP_EXP_DIR      "%s: %llu: expected mapfile directive (%s): %s"
1190 @ MSG_MAP_SFLG_EXBANG   "%s: %llu: '!' appears without corresponding flag"
1191 @ MSG_MAP_EXP_FILNAM    "%s: %llu: expected file name following %s: %s"
1192 @ MSG_MAP_EXP_FILPATH   "%s: %llu: expected file path following %s: %s"
1193 @ MSG_MAP_EXP_INT       "%s: %llu: expected integer value following %s: %s"
1194 @ MSG_MAP_EXP_LBKTT     "%s: %llu: expected '{' following %s: %s"
1195 @ MSG_MAP_EXP_OBJNAM    "%s: %llu: expected object name following %s: %s"
1196 @ MSG_MAP_SFLG_ONEBANG  "%s: %llu: '!' can only be specified once per flag"
1197 @ MSG_MAP_EXP_SECFLAG   "%s: %llu: expected section flag (%s), '!', or \
1198                                terminator ('', ','): %s"
1199 @ MSG_MAP_EXP_SECNAM    "%s: %llu: expected section name following %s: %s"
1200 @ MSG_MAP_EXP_SEGFLAG   "%s: %llu: expected segment flag (%s), or \
1201                                terminator ('', ','): %s"
1202 @ MSG_MAP_EXP_ECNAM    "%s: %llu: expected entrance criteria (ASSIGN_SECTION) \
1203                                \name, or terminator ('', ','): %s"
1204 @ MSG_MAP_EXP_SEGNAM    "%s: %llu: expected segment name following %s: %s"
1205 @ MSG_MAP_EXP_SEM       "%s: %llu: expected ';' to terminate %s: %s"
1206 @ MSG_MAP_EXP_SEMLBKT   "%s: %llu: expected ';' or '{' following %s: %s"
1207 @ MSG_MAP_EXP_SEMRBKT   "%s: %llu: expected ';' or '}' to terminate %s: %s"
1208 @ MSG_MAP_EXP_SHTYPE    "%s: %llu: expected section type: %s"
1209 @ MSG_MAP_EXP_SYM       "%s: %llu: expected symbol name, symbol scope, \
1210                                \or '**': %s"
1211 @ MSG_MAP_EXP_SYMEND   "%s: %llu: expected inherited version name, or \
1212                                terminator ('', ','): %s"
1213 @ MSG_MAP_EXP_SYMDELIM  "%s: %llu: expected one of ':', ',', or '{': %s"
1214 @ MSG_MAP_EXP_SYMFLAG  "%s: %llu: expected symbol flag (%s), or \
1215                                terminator ('', ','): %s"
1216 @ MSG_MAP_EXP_SYMNAME  "%s: %llu: expected symbol name following %s: %s"
1217 @ MSG_MAP_EXP_SYMSCOPE  "%s: %llu: expected symbol scope (%s): %s"
1218 @ MSG_MAP_EXP_SYMTYPE  "%s: %llu: expected symbol type (%s): %s"
1219 @ MSG_MAP_EXP_VERSION   "%s: %llu: expected version name following %s: %s"
1220 @ MSG_MAP_BADEXTRA     "%s: %llu: unexpected text found following %s directive"
1221 @ MSG_MAP_VALUELIMIT   "%s: %llu: numeric value exceeds word size: %s"
1222 @ MSG_MAP_MALVALUE     "%s: %llu: malformed numeric value: %s"
1223 @ MSG_MAP_BADVALUETAIL "%s: %llu: unexpected characters following numeric \
1224                                \constant: %s"
1225 @ MSG_MAP_WSNEEDED    "%s: %llu: whitespace needed before token: %s"
1226 @ MSG_MAP_BADCHAR     "%s: %llu: unexpected text: %s"
1227 @ MSG_MAP_BADKQUOTE    "%s: %llu: mapfile keywords should not be quoted: %s"
1228 @ MSG_MAP_CDIR_NOTBOL  "%s: %llu: mapfile control directive not at start of \
1229                                \line: %s"
1230 @ MSG_MAP_NOATTR      "%s: %llu: %s specified no attributes (empty {})"
1231 @ MSG_MAP_NOVALUES    "%s: %llu: %s specified without values"
1232 @ MSG_MAP_INTERR      "<internal error>"
1233 @ MSG_MAP_ISORDVER   "%s: %llu: version 0 mapfile ?O flag and version 1 \
1234                                \segment IS_ORDER attribute are mutually exclusive: %s"
1235 @ MSG_MAP_SYMATTR    "symbol attributes";

1237 # Mapfile Control Directives

1239 @ MSG_MAP_CDIR_BADVDIR "%s: %llu: $mapfile_version directive must specify \
1240                                \version 2 or higher: %d"
1241 @ MSG_MAP_CDIR_BADVER  "%s: %llu: unknown mapfile version: %d"
1242 @ MSG_MAP_CDIR_REPVER  "%s: %llu: $mapfile_version must be first directive \
1243                                \in file"
1244 @ MSG_MAP_CDIR_REQARG  "%s: %llu: %s directive requires an argument"
1245 @ MSG_MAP_CDIR REQNOARG "%s: %llu: %s directive does not accept arguments"
1246 @ MSG_MAP_CDIR_BAD     "%s: %llu: unrecognized mapfile control directive"
1247 @ MSG_MAP_CDIR_NOIF    "%s: %llu: %s directive used without opening $if"
1248 @ MSG_MAP_CDIR_ELSE    "%s: %llu: %s directive preceded by $else on line %d"
1249 @ MSG_MAP_CDIR_NOEND   "%s: %llu: EOF encountered without closing $endif \

```

```

1250                                for $if on line %d"
1251 @ MSG_MAP_CDIR_ERROR    "%s: %llu: error: %s"

1254 # Mapfile Conditional Expressions

1256 @ MSG_MAP_CEXP_TOKERR   "%s: %llu: syntax error in conditional expression at: %s"
1257 @ MSG_MAP_CEXP_SEMERR   "%s: %llu: malformed conditional expression"
1258 @ MSG_MAP_CEXP_BADOPUSE "%s: %llu: invalid operator use in conditional \
1259                                \expression"
1260 @ MSG_MAP_CEXP_UNBALPAR "%s: %llu: unbalanced parenthesis in conditional \
1261                                \expression"
1262 @ MSG_MAP_BADCESC      "%s: %llu: unrecognized escape in double quoted \
1263                                \token: \\%c\\n"
1265 # Generic error diagnostic labels

1267 @ MSG_STR_NULL          "(null)"

1269 @ MSG_DBG_DFLT_FMT     "debug: "
1270 @ MSG_DBG_AOUT_FMT      "debug: a.out: "
1271 @ MSG_DBG_NAME_FMT      "debug: %s: "

1273 # -z assert-deflib strings

1275 @ MSG_ARG_ASSDEFLIB_MALFORMED "library name malformed: %s"
1276 @ MSG_ARG_ASSDEFLIB_FOUND   "dynamic library found on default search path \
1277                                \(%s): lib%s.so"
1279 @ _END_

1282 # Software identification. Note, the SGU strings is historic, and has
1283 # little relevance. It is preserved as applications have used this
1284 # string to identify the Solaris link-editor.

1286 @ MSG_SGS_ID            "ld: Software Generation Utilities - \
1287                                \Solaris Link Editors: "

1289 # The following strings represent reserved words, files, pathnames and symbols.
1290 # Reference to this strings is via the MSG_ORIG() macro, and thus no message
1291 # translation is required.

1293 @ MSG_DBG_FOPEN_MODE    "w"
1295 @ MSG_DBG_CLS32_FMT     "32: "
1296 @ MSG_DBG_CLS64_FMT     "64: "
1298 @ MSG_STR_PATHTOK       ";:"
1299 @ MSG_STR_AOUT          "a.out"
1301 @ MSG_STR_LIB_A         "%s/lib%s.a"
1302 @ MSG_STR_LIB_SO        "%s/lib%s.so"
1303 @ MSG_STR_PATH          "%s/%s"
1304 @ MSG_STR_STRLN         "%s\\n"
1305 @ MSG_STR_NL             "\n"
1306 @ MSG_STR_CAPGROUPID   "CAP_GROUP_%d"
1308 @ MSG_STR_LD_DYNAMIC    "dynamic"
1309 @ MSG_STR_SYMBOLIC      "symbolic"
1310 @ MSG_STR_ELIMINATE     "eliminate"
1311 @ MSG_STR_LOCAL          "local"
1312 @ MSG_STR_PROGBITS      "progbits"
1313 @ MSG_STR_SYMTAB         "symsym"
1314 @ MSG_STR_DYNSYM        "dynam"
1315 @ MSG_STR_REL             "rel"

```

```

1316 @ MSG_STR_REL A "rela"
1317 @ MSG_STR_STRTAB "strtab"
1318 @ MSG_STR_HASH "hash"
1319 @ MSG_STR_LIB "lib"
1320 @ MSG_STR_NOTE "note"
1321 @ MSG_STR_NOBITS "nobits"
1322 @ MSG_STR_HWCAP_1 "hwcap_1"
1323 @ MSG_STR_SFCAP_1 "sfcap_1"
1324 @ MSG_STR_SOEXT ".so"

1326 @ MSG_STR_OPTIONS "3:6:abc:d:e:f:h:i:l:m:o:p:rstu:z:B:CD:F:GI:L:M:N:P:Q:R:\nS:VW:Y:?"
1327

1329 # Argument processing strings

1331 @ MSG_ARG_3 "-3"
1332 @ MSG_ARG_6 "-6"
1333 @ MSG_ARG_A "-a"
1334 @ MSG_ARG_B "-b"
1335 @ MSG_ARG_CB "-B"
1336 @ MSG_ARG_BDIRECT "-Bdirect"
1337 @ MSG_ARG_BDYNAMIC "-Bdynamic"
1338 @ MSG_ARG_BELEMINATE "-Beliminate"
1339 @ MSG_ARG_BGROUP "-Bgroup"
1340 @ MSG_ARG_BLOCAL "-Blocal"
1341 @ MSG_ARG_BNODIRECT "-Bnodirect"
1342 @ MSG_ARG_BSMBOLIC "-Bsymbolic"
1343 @ MSG_ARG_BTRANSLATOR "-Btranslator"
1344 @ MSG_ARG_C "-c"
1345 @ MSG_ARG_D "-d"
1346 @ MSG_ARG_DY "-dy"
1347 @ MSG_ARG_CI "-I"
1348 @ MSG_ARG_CN "-N"
1349 @ MSG_ARG_P "-p"
1350 @ MSG_ARG_CP "-P"
1351 @ MSG_ARG_CQ "-Q"
1352 @ MSG_ARG_CY "-Y"
1353 @ MSG_ARG_CYL "-YL"
1354 @ MSG_ARG_CYP "-YP"
1355 @ MSG_ARG_CYU "-YU"
1356 @ MSG_ARG_Z "-Z"
1357 @ MSG_ARG_ZDEFNODEF "-z[defs|nodefs]"
1358 @ MSG_ARG_ZGUIDE "-zguide"
1359 @ MSG_ARG_ZNODEF "-znodefs"
1360 @ MSG_ARG_ZNOINTERP "-znointerp"
1361 @ MSG_ARG_ZRELAXRELOC "-zrelaxreloc"
1362 @ MSG_ARG_ZNORELAXRELOC "-zno relaxreloc"
1363 @ MSG_ARG_ZTEXT "-ztext"
1364 @ MSG_ARG_ZTEXTOFF "-ztextoff"
1365 @ MSG_ARG_ZTEXTWARN "-ztextwarn"
1366 @ MSG_ARG_ZTEXTTALL "-z[text|textwarn|textoff]"
1367 @ MSG_ARG_ZLOADFLTR "-zloadfltr"
1368 @ MSG_ARG_ZCOMBRELOC "-zcombreloc"
1369 @ MSG_ARG_ZSYMBOLCAP "-zsymbolcap"
1370 @ MSG_ARG_ZFATWNOFATW "-z[fatal-warnings|nofatalwarnings]"

1372 @ MSG_ARG_ABSEXEC "absexec"
1373 @ MSG_ARG_ALTEXEC64 "altexec64"
1374 @ MSG_ARG_NOCOMPSTRTAB "nocomprtab"
1375 @ MSG_ARG_GROUPPERM "groupperm"
1376 @ MSG_ARG_NOGROUPPERM "nogroupperm"
1377 @ MSG_ARG_LAZYLOAD "lazyload"
1378 @ MSG_ARG_NOLOAD "noload"
1379 @ MSG_ARG_INTERPOSE "interpose"
1380 @ MSG_ARG_DIRECT "direct"
1381 @ MSG_ARG_NODIRECT "nodirect"

```

```

1382 @ MSG_ARG_IGNORE "ignore"
1383 @ MSG_ARG_RECORD "record"
1384 @ MSG_ARG_INITFIRST "initfirst"
1385 @ MSG_ARG_INITARRAY "initarray="
1386 @ MSG_ARG_FINIARRAY "finiarray="
1387 @ MSG_ARG_PREINITARRAY "preinitarray="
1388 @ MSG_ARG_RTLDINFO "rtldinfo="
1389 @ MSG_ARG_DTRACE "dtrace="
1390 @ MSG_ARG_TRANSLATOR "translator"
1391 @ MSG_ARG_NOOPEN "nodopen"
1392 @ MSG_ARG_NOW "now"
1393 @ MSG_ARG_ORIGIN "origin"
1394 @ MSG_ARG_DEFS "defs"
1395 @ MSG_ARG_NODEFS "nodefs"
1396 @ MSG_ARG_NODUMP "nodump"
1397 @ MSG_ARG_NOVERSION "noversion"
1398 @ MSG_ARG_TEXT "text"
1399 @ MSG_ARG_TEXTOFF "textoff"
1400 @ MSG_ARG_TEXTWARN "textwarn"
1401 @ MSG_ARG_MULDEFS "muldefs"
1402 @ MSG_ARG_NODELETE "nodelete"
1403 @ MSG_ARG_NOINTERP "nointerp"
1404 @ MSG_ARG_NOPARTIAL "nopartial"
1405 @ MSG_ARG_NORELOC "noreloc"
1406 @ MSG_ARG_REDLOCSYM "redlocsym"
1407 @ MSG_ARG_VERBOSE "verbose"
1408 @ MSG_ARG_WEAKEXT "weakextract"
1409 @ MSG_ARG_LOADFLTR "loadfltr"
1410 @ MSG_ARG_ALLEXTRT "allextact"
1411 @ MSG_ARG_DFLEXTRT "defaultextract"
1412 @ MSG_ARG_COMBRELOC "combreloc"
1413 @ MSG_ARG_NOCOMBRELOC "nocombreloc"
1414 @ MSG_ARG_NODEFAULTLIB "nodefaultlib"
1415 @ MSG_ARG_ENDFILTEE "endfiltee"
1416 @ MSG_ARG_LD32 "ld32"
1417 @ MSG_ARG_LD64 "ld64"
1418 @ MSG_ARG_RESCAN "rescan"
1419 @ MSG_ARG_RESCAN_NOW "rescan-now"
1420 @ MSG_ARG_RESCAN_START "rescan-start"
1421 @ MSG_ARG_RESCAN_END "rescan-end"
1422 @ MSG_ARG_GUIDE "guidance"
1423 @ MSG_ARG_NOLDYNSYM "noldynsym"
1424 @ MSG_ARG_RELAXRELOC "relaxreloc"
1425 @ MSG_ARG_NORELAXRELOC "norelaxreloc"
1426 @ MSG_ARG_NOSIGHANDLER "nosighandler"
1427 @ MSG_ARG_GLOBAUDIT "globalaudit"
1428 @ MSG_ARG_TARGET "target="
1429 @ MSG_ARG_WRAP "wrap="
1430 @ MSG_ARG_FATWARN "fatal-warnings"
1431 @ MSG_ARG_NOFATWARN "nofatal-warnings"
1432 @ MSG_ARG_HELP "help"
1433 @ MSG_ARG_GROUP "group"
1434 @ MSG_ARG_REDUCE "reduce"
1435 @ MSG_ARG_STATIC "static"
1436 @ MSG_ARG_SYMBOLCAP "symbolcap"
1437 @ MSG_ARG_DEFERRED "deferred"
1438 @ MSG_ARG_NODEFERRED "nodeferred"
1439 @ MSG_ARG_ASSDEFLIB "assert-deflib"

1441 @ MSG_ARG_LCOM "L,"
1442 @ MSG_ARG_PCOM "P,"
1443 @ MSG_ARG_UCOM "U,"

1445 @ MSG_ARG_T_RPATH "rpath"
1446 @ MSG_ARG_T_SHARED "shared"
1447 @ MSG_ARG_T SONAME "soname"

```

```

1448 @ MSG_ARG_T_WL      "l,-"
1450 @ MSG_ARG_T_AUXFLTR  "-auxiliary"
1451 @ MSG_ARG_T_MULDEFS  "-allow-multiple-definition"
1452 @ MSG_ARG_T_INTERP   "-dynamic-linker"
1453 @ MSG_ARG_T_ENDGROUP  "-end-group"
1454 @ MSG_ARG_T_ENTRY    "-entry"
1455 @ MSG_ARG_T_STDFLTR  "-filter"
1456 @ MSG_ARG_T_FATWARN  "-fatal-warnings"
1457 @ MSG_ARG_T_NOFATWARN "-no-fatal-warnings"
1458 @ MSG_ARG_T_HELP     "-help"
1459 @ MSG_ARG_T_LIBRARY   "-library"
1460 @ MSG_ARG_T_LIBPATH   "-library-path"
1461 @ MSG_ARG_T_NOUNDEF   "-no-undefined"
1462 @ MSG_ARG_T_NOWHOLEARC "-no-whole-archive"
1463 @ MSG_ARG_T_OUTPUT    "-output"
1464 @ MSG_ARG_T_RELOCATABLE "-relocatable"
1465 @ MSG_ARG_T_STARTGROUP "-start-group"
1466 @ MSG_ARG_T_STRIP     "-strip-all"
1467 @ MSG_ARG_T_UNDEF     "-undefined"
1468 @ MSG_ARG_T_VERSION   "-version"
1469 @ MSG_ARG_T_WHOLEARC  "-whole-archive"
1470 @ MSG_ARG_T_WRAP      "-wrap"
1471 @ MSG_ARG_T_OPAR      "("
1472 @ MSG_ARG_T_CPAR      ")"

```

```

1474 # -z guidance=item strings
1475 @ MSG_ARG_GUIDE_DELIM  ",: \t"
1476 @ MSG_ARG_GUIDE_NO_ALL  "noall"
1477 @ MSG_ARG_GUIDE_NO_DEFS  "nodefs"
1478 @ MSG_ARG_GUIDE_NO_DIRECT "-nodirect"
1479 @ MSG_ARG_GUIDE_NO_LAZYLOAD "-nolazyload"
1480 @ MSG_ARG_GUIDE_NO_MAPFILE "-nomapfile"
1481 @ MSG_ARG_GUIDE_NO_TEXT  "notext"
1482 @ MSG_ARG_GUIDE_NO_UNUSED "-nounused"

```

```
1484 # Environment variable strings
```

```

1486 @ MSG_LD_RUN_PATH     "LD_RUN_PATH"
1487 @ MSG_LD_LIBPATH_32    "LD_LIBRARY_PATH_32"
1488 @ MSG_LD_LIBPATH_64    "LD_LIBRARY_PATH_64"
1489 @ MSG_LD_LIBPATH      "LD_LIBRARY_PATH"

1491 @ MSG_LD_NOVERSION_32  "LD_NOVERSION_32"
1492 @ MSG_LD_NOVERSION_64  "LD_NOVERSION_64"
1493 @ MSG_LD_NOVERSION    "LD_NOVERSION"

1495 @ MSG_SGS_SUPPORT_32   "SGS_SUPPORT_32"
1496 @ MSG_SGS_SUPPORT_64   "SGS_SUPPORT_64"
1497 @ MSG_SGS_SUPPORT      "SGS_SUPPORT"

```

```
1500 # Symbol names
```

```
1502 @ MSG_SYM_LIBVER_U    "_lib_version"
```

```
1505 # Mapfile tokens
```

```

1507 @ MSG_MAP_LOAD        "load"
1508 @ MSG_MAP_NOTE         "note"
1509 @ MSG_MAP_NULL         "null"
1510 @ MSG_MAP_STACK        "stack"
1511 @ MSG_MAP_ADDVERS      "addvers"
1512 @ MSG_MAP_FUNCTION      "function"
1513 @ MSG_MAP_DATA          "data"

```

```

1514 @ MSG_MAP_COMMON      "common"
1515 @ MSG_MAP_PARENT       "parent"
1516 @ MSG_MAP_EXTERN        "extern"
1517 @ MSG_MAP_DIRECT        "direct"
1518 @ MSG_MAP_NODIRECT      "nodirect"
1519 @ MSG_MAP_FILTER        "filter"
1520 @ MSG_MAP_AUXILIARY     "auxiliary"
1521 @ MSG_MAP_OVERRIDE      "override"
1522 @ MSG_MAP_INTERPOSE      "interpose"
1523 @ MSG_MAP_DYNSORT       "dyncsort"
1524 @ MSG_MAP_NODYNSORT     "nodynsoft"

1526 @ MSG_MAPKW_ALIGN       "ALIGN"
1527 @ MSG_MAPKW_ALLOC       "ALLOC"
1528 @ MSG_MAPKW_ALLOW       "ALLOW"
1529 @ MSG_MAPKW_AMD64_LARGE  "AMD64_LARGE"
1530 @ MSG_MAPKW_ASSIGN_SECTION "ASSIGN_SECTION"
1531 @ MSG_MAPKW_AUX          "AUXILIARY"
1532 @ MSG_MAPKW_CAPABILITY   "CAPABILITY"
1533 @ MSG_MAPKW_COMMON       "COMMON"
1534 @ MSG_MAPKW_DATA          "DATA"
1535 @ MSG_MAPKW_DEFAULT      "DEFAULT"
1536 @ MSG_MAPKW_DEPEND VERSIONS "DEPEND VERSIONS"
1537 @ MSG_MAPKW_DIRECT       "DIRECT"
1538 @ MSG_MAPKW_DISABLE      "DISABLE"
1539 @ MSG_MAPKW_DYN SORT     "DYN SORT"
1540 @ MSG_MAPKW_ELIMINATE    "ELIMINATE"
1541 @ MSG_MAPKW_EXECUTE      "EXECUTE"
1542 @ MSG_MAPKW_EXPORTED     "EXPORTED"
1543 @ MSG_MAPKW_EXTERN        "EXTERN"
1544 @ MSG_MAPKW_FILTER        "FILTER"
1545 @ MSG_MAPKW_FILE_BASENAME "FILE_BASENAME"
1546 @ MSG_MAPKW_FILE_PATH     "FILE_PATH"
1547 @ MSG_MAPKW_FILE_OBJNAME  "FILE_OBJNAME"
1548 @ MSG_MAPKW_FUNCTION      "FUNCTION"
1549 @ MSG_MAPKW_FLAGS          "FLAGS"
1550 @ MSG_MAPKW_GLOBAL        "GLOBAL"
1551 @ MSG_MAPKW_INTERPOSE    "INTERPOSE"
1552 @ MSG_MAPKW_HIDDEN        "HIDDEN"
1553 @ MSG_MAPKW_HDR_NOALLOC   "HDR_NOALLOC"
1554 @ MSG_MAPKW_HW             "HW"
1555 @ MSG_MAPKW_HW_1           "HW_1"
1556 @ MSG_MAPKW_HW_2           "HW_2"
1557 @ MSG_MAPKW_IS_NAME        "IS_NAME"
1558 @ MSG_MAPKW_IS_ORDER       "IS_ORDER"
1559 @ MSG_MAPKW_LOAD_SEGMENT   "LOAD_SEGMENT"
1560 @ MSG_MAPKW_LOCAL          "LOCAL"
1561 @ MSG_MAPKW_MACHINE        "MACHINE"
1562 @ MSG_MAPKW_MAX_SIZE       "MAX_SIZE"
1563 @ MSG_MAPKW_NOHDR          "NOHDR"
1564 @ MSG_MAPKW_NODIRECT       "NODIRECT"
1565 @ MSG_MAPKW_NODYNSORT      "NODYNSORT"
1566 @ MSG_MAPKW_NOTE_SEGMENT   "NOTE_SEGMENT"
1567 @ MSG_MAPKW_NULL_SEGMENT   "NULL_SEGMENT"
1568 @ MSG_MAPKW_OS_ORDER        "OS_ORDER"
1569 @ MSG_MAPKW_PADDR          "PADDR"
1570 @ MSG_MAPKW_PARENT          "PARENT"
1571 @ MSG_MAPKW_PHDR_ADD_NULL  "PHDR_ADD_NULL"
1572 @ MSG_MAPKW_PLATFROM       "PLATFROM"
1573 @ MSG_MAPKW_PROTECTED      "PROTECTED"
1574 @ MSG_MAPKW_READ            "READ"
1575 @ MSG_MAPKW_ROUND           "ROUND"
1576 @ MSG_MAPKW_REQUIRE         "REQUIRE"
1577 @ MSG_MAPKW_SEGMENT_ORDER   "SEGMENT_ORDER"
1578 @ MSG_MAPKW_SF              "SF"
1579 @ MSG_MAPKW_SF_1             "SF_1"

```

```
1580 @ MSG_MAPKW_SINGLETON      "SINGLETON"
1581 @ MSG_MAPKW_SIZE          "SIZE"
1582 @ MSG_MAPKW_SIZE_SYMBOL    "SIZE_SYMBOL"
1583 @ MSG_MAPKW_STACK          "STACK"
1584 @ MSG_MAPKW_SYMBOL_SCOPE    "SYMBOL_SCOPE"
1585 @ MSG_MAPKW_SYMBOL_VERSION  "SYMBOL_VERSION"
1586 @ MSG_MAPKW_SYMBOLIC        "SYMBOLIC"
1587 @ MSG_MAPKW_TYPE            "TYPE"
1588 @ MSG_MAPKW_VADDR           "VADDR"
1589 @ MSG_MAPKW_VALUE           "VALUE"
1590 @ MSG_MAPKW_WRITE           "WRITE"

1593 @ MSG_STR_DTRACE          "PT_SUNWDTRACE"
```

new/usr/src/cmd/sgs/libld/common/util.c

```

20166 Sun Oct 27 14:16:05 2013
new/usr/src/cmd/sgs/libld/common/util.c
4270 ld(1) argument error reporting is still pretty bad
4227 ld -library-path is translated to -l-path, not -L
***** unchanged_portion_omitted_ *****
303 /*
304 * Determine whether this string, possibly with an associated option, should
305 * be translated to an option character. If so, update the optind and optarg
306 * and optopt as described for short options in getopt(3c).
307 *
308 * entry:
309 *      lml - Link map list for debug messages
310 *      ndx - Starting optind for current item
311 *      argc, argv - Command line arguments
312 *      arg - Option to be examined
313 *      c, opt - Option character (c) and corresponding long name (opt)
314 *      optsz - 0 if option does not accept a value. If option does
315 *              accept a value, strlen(opt), giving the offset to the
316 *              value if the option and value are combined in one string.
317 *      cbfunc - NULL, or pointer to function to call if a translation is
318 *              successful.
319 */
320 static int
321 str2chr(Lm_list *lml, int ndx, int argc, char **argv, char *arg, int c,
322 const char *opt, size_t optsz, int cbfunc(int))
323 {
324     if (optsz == 0) {
325         /*
326         * Compare a single option (ie. there's no associated option
327         * argument).
328         */
329         if (strcmp(arg, opt) == 0) {
330             DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));
331             optind += 1;
332             optopt = c;
333 #endif /* ! codereview */
334             return (c);
335         }
336     } else if ((strcmp(arg, opt) == 0) ||
337     ((arg[optsz] == '=') & strncmp(arg, opt, optsz) == 0)) {
338
339     } else if (strncmp(arg, opt, optsz) == 0) {
340         /*
341         * Otherwise, compare the option name, which may be
342         * concatenated with the option argument.
343         */
344         DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));
345
346         if (arg[optsz] == '\0') {
347             /*
348             * Optarg is the next argument (white space separated).
349             * Make sure an optarg is available, and if not return
350             * a failure to prevent any fall-through to the generic
351             * getopt() processing.
352             */
353             /*
354             * Since we'll be completely failing this option we
355             * don't want to update optopt with the translation,
356             * but also need to set it to _something_. Setting it
357             * to the '-' of the argument causes us to behave
358             * correctly.

```

new/usr/src/cmd/sgs/libld/common/util.c

```
356 #endif /* ! codereview */
357     */
358     if ((++optind + 1) > argc) {
359         optopt = argv[0];
360     }
361     return ('?');
362 }
363 optarg = argv[optind];
364 optind++;
365 } else {
366     /*
367      * Optarg concatenated to option (no white space).
368      * GNU option/option argument pairs can be represented
369      * with a "=" separator. If this is the case, remove
370      * the separator.
371      */
372     optarg = &arg[optsz];
373     optind++;
374     if (*optarg == '=') {
375         if ((*(++optarg) == '\0')) {
376             optopt = argv[0];
377         }
378     }
379 }
380
381     if (cbfunc != NULL)
382         c = (*cbfunc)(c);
383     optopt = c;
384
385 }
386 return (0);
387 }



---



unchanged portion omitted


```

```
*****
88046 Sun Oct 27 14:16:05 2013
new/usr/src/cmd/sgs/packages/common/SUNWORLD-README
4270 ld(1) argument error reporting is still pretty bad
4227 ld --library-path is translated to -l-path, not -L
*****
1 # Copyright (c) 1996, 2010, Oracle and/or its affiliates. All rights reserved.
3 #
4 # CDDL HEADER START
5 #
6 # The contents of this file are subject to the terms of the
7 # Common Development and Distribution License (the "License").
8 # You may not use this file except in compliance with the License.
9 #
10 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
11 # or http://www.opensolaris.org/os/licensing.
12 # See the License for the specific language governing permissions
13 # and limitations under the License.
14 #
15 # When distributing Covered Code, include this CDDL HEADER in each
16 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
17 # If applicable, add the following below this CDDL HEADER, with the
18 # fields enclosed by brackets "[]" replaced with your own identifying
19 # information: Portions Copyright [yyyy] [name of copyright owner]
20 #
21 # CDDL HEADER END
22 #
23 # Note: The contents of this file are used to determine the versioning
24 # information for the SGS toolset. The number of CRs listed in
25 # this file must grow monotonically, or the SGS version will
26 # move backwards, causing a great deal of confusion. As such,
27 # CRs must never be removed from this file. See
28 # libconv/common/bld_vernote.ksh, and bug#4519569 for more
29 # details on SGS versioning.
30 #
31 -----
32 SUNWORLD - link-editors development package.
33 -----
```

35 The SUNWORLD package is an internal development package containing the
36 link-editors and some related tools. All components live in the OSNET
37 source base, but not all components are delivered as part of the normal
38 OSNET consolidation. The intent of this package is to provide access
39 to new features/bugfixes before they become generally available.

41 General link-editor information can be found:

43 <http://linkers.central/>
44 <http://linkers.sfbay/> (also known as linkers.eng)

46 Comments and Questions:

48 Contact Rod Evans, Ali Bahrami, and/or Seizo Sakurai.

50 Warnings:

52 The postremove script for this package employs /usr/sbin/static/mv,
53 and thus, besides the common core dependencies, this package also
54 has a dependency on the SUNWSTL package.

56 Patches:

58 If the patch has been made official, you'll find it in:

60 <http://sunsolve.east/cgi/show.pl?target=patches/os-patches>

62 If it hasn't been released, the patch will be in:
64 /net/sunsoftpatch/patches/temporary

66 Note, any patches logged here refer to the temporary ("T") name, as we
67 never know when they're made official, and although we try to keep all
68 patch information up-to-date the real status of any patch can be
69 determined from:
71 <http://sunsoftpatch.eng>

73 If it has been obsoleted, the patch will be in:
74 /net/on\${RELEASE}-patch/on\${RELEASE}/patches/\${MACH}/obsolete

78 History:

80 Note, starting after Solaris 10, letter codes in parenthesis may
81 be found following the bug synopsis. Their meanings are as follows:

83 (D) A documentation change accompanies the implementation change.
84 (P) A packaging change accompanies the implementation change.

86 In all cases, see the implementation bug report for details.

88 The following bug fixes exist in the OSNET consolidation workspace
89 from which this package is created:

91 -----
92 Solaris 8
93 -----
94 Bugid Risk Synopsis
95 -----
96 4225937 i386 linker emits sparc specific warning messages
97 4215164 shf_order flag handling broken by fix for 4194028.
98 4215587 using ld and the -r option on solaris 7 with compiler option -xarch=v9
99 causes link errors.
100 4234657 103627-08 breaks purify 4.2 (plt padding should not be enabled for
101 32-bit)
102 4235241 dbx no longer gets dlclose notification.
103 -----
104 All the above changes are incorporated in the following patches:
105 Solaris/SunOS 5.7_sparc patch 106950-05 (never released)
106 Solaris/SunOS 5.7_x86 patch 106951-05 (never released)
107 Solaris/SunOS 5.6_sparc patch 107733-02 (never released)
108 Solaris/SunOS 5.6_x86 patch 107734-02
109 -----
110 4248290 ineted dumps core upon bootup - failure in dlclose() logic.
111 4238071 dlopen() leaks while descriptors under low memory conditions
112 -----
113 All the above changes are incorporated in the following patches:
114 Solaris/SunOS 5.7_sparc patch 106950-06
115 Solaris/SunOS 5.7_x86 patch 106951-06
116 Solaris/SunOS 5.6_sparc patch 107733-03 (never released)
117 Solaris/SunOS 5.6_x86 patch 107734-03
118 -----
119 4267980 INITFIRST flag of the shard object could be ignored.
120 -----
121 All the above changes plus:
122 4238973 fix for 4121152 affects linking of Ada objects
123 4158744 patch 103627-02 causes core when RPATH has blank entry and
124 dlopen/dlclose is used
125 are incorporated in the following patches:
126 Solaris/SunOS 5.5.1_sparc patch 103627-12 (never released)

```

127      Solaris/SunOS 5.5.1_x86      patch 103628-11
128 -----
129 4256518 miscalculated calloc() during dlclose/tsorting can result in segv
130 4254171 DT_SPARC_REGISTER has invalid value associated with it.
131 -----
132 All the above changes are incorporated in the following patches:
133      Solaris/SunOS 5.7_sparc      patch 106950-07
134      Solaris/SunOS 5.7_x86       patch 106951-07
135      Solaris/SunOS 5.6_sparc      patch 107733-04 (never released)
136      Solaris/SunOS 5.6_x86       patch 107734-04
137 -----
138 4293159 ld needs to combine sections with and without SHF_ORDERED flag(comdat)
139 4292238 linking a library which has a static char ptr invokes mprotect() call
140 -----
141 All the above changes except for:
142 4256518 miscalculated calloc() during dlclose/tsorting can result in segv
143 4254171 DT_SPARC_REGISTER has invalid value associated with it.
144 plus:
145 4238973 fix for 4121152 affects linking of Ada objects
146 4158744 patch 103627-02 causes core when RPATH has blank entry and
147      dlopen/dlclose is used
148 are incorporated in the following patches:
149      Solaris/SunOS 5.5.1_sparc      patch 103627-13
150      Solaris/SunOS 5.5.1_x86       patch 103628-12
151 -----
152 All the above changes are incorporated in the following patches:
153      Solaris/SunOS 5.7_sparc      patch 106950-08
154      Solaris/SunOS 5.7_x86       patch 106951-08
155      Solaris/SunOS 5.6_sparc      patch 107733-05
156      Solaris/SunOS 5.6_x86       patch 107734-05
157 -----
158 4295613 COMMON symbol resolution can be incorrect
159 -----
160 All the above changes plus:
161 4238973 fix for 4121152 affects linking of Ada objects
162 4158744 patch 103627-02 causes core when RPATH has blank entry and
163      dlopen/dlclose is used
164 are incorporated in the following patches:
165      Solaris/SunOS 5.5.1_sparc      patch 103627-14
166      Solaris/SunOS 5.5.1_x86       patch 103628-13
167 -----
168 All the above changes plus:
169 4351197 nfs performance problem by 103627-13
170 are incorporated in the following patches:
171      Solaris/SunOS 5.5.1_sparc      patch 103627-15
172      Solaris/SunOS 5.5.1_x86       patch 103628-14
173 -----
174 All the above changes are incorporated in the following patches:
175      Solaris/SunOS 5.7_sparc      patch 106950-09
176      Solaris/SunOS 5.7_x86       patch 106951-09
177      Solaris/SunOS 5.6_sparc      patch 107733-06
178      Solaris/SunOS 5.6_x86       patch 107734-06
179 -----
180 4158971 increase the default segment alignment for i386 to 64k
181 4064994 Add an $ISALIST token to those understood by the dynamic linker
182 xxxxxxxx ia64 common code putback
183 4239308 LD_DEBUG busted for sparc machines
184 4239008 Support MAP_ANON
185 4238494 link-auditing extensions required
186 4232239 R_SPARC_LOX10 truncates field
187 4231722 R_SPARC_UA* relocations are busted
188 4235514 R_SPARC_OLO10 relocation fails
189 4244025 sgmsvc update
190 4239281 need to support SECREL relocations for ia64
191 4253751 ia64 linker must support PT_IA_64_UNWIND tables
192 4259254 dlmopen mistakenly closes fd 0 (stdin) under certain error conditions

```

```

193 4260872 libelf hangs when libthread present
194 4224569 linker core dumping when profiling specified
195 4270937 need mechanism to suppress ld.so.1's use of a default search path
196 1050476 ld.so to permit configuration of search path
197 4273654 filtee processing using $ISALIST could be optimized
198 4271860 get MERCED cruft out of elf.h
199 4248991 Dynamic loader (via PLT) corrupts register G4
200 4275754 cannot mmap file: Resource temporarily unavailable
201 4277689 The linker can not handle relocation against MOVE tabl
202 4270766 atexit processing required on dlclose().
203 4279229 Add a "release" token to those understood by the dynamic linker
204 4215433 ld can bus error when insufficient disc space exists for output file
205 4285571 Pssst, want some free disk space? ld's miscalculating.
206 4286236 ar gives confusing "bad format" error with a null .stab section
207 4286838 ld.so.1 can't handle a no-bits segment
208 4287364 ld.so.1 runtime configuration cleanup
209 4289573 disable linking of ia64 binaries for Solaris8
210 4293966 crle(1)'s default directories should be supplied
211 -----
213 -----
214 Solaris 8 600 (1st Q-update - s28u1)
215 -----
216 Bugid Risk Synopsis
217 -----
218 4309212 dlsym can't find symbol
219 4311226 rejection of preloading in secure apps is inconsistent
220 4312449 dlclose: invalid deletion of dependency can occur using RTLD_GLOBAL
221 -----
222 All the above changes are incorporated in the following patches:
223      Solaris/SunOS 5.8_sparc      patch 109147-01
224      Solaris/SunOS 5.8_x86       patch 109148-01
225      Solaris/SunOS 5.7_sparc      patch 106950-10
226      Solaris/SunOS 5.7_x86       patch 106951-10
227      Solaris/SunOS 5.6_sparc      patch 107733-07
228      Solaris/SunOS 5.6_x86       patch 107734-07
229 -----
231 -----
232 Solaris 8 900 (2nd Q-update - s28u2)
233 -----
234 Bugid Risk Synopsis
235 -----
236 4324775 non-PIC code & -zcombreloc don't mix very well...
237 4327653 run-time linker should preload tables it will process (madvise)
238 4324324 shared object code can be referenced before .init has fired
239 4321634 .init firing of multiple INITFIRST objects can fail
240 -----
241 All the above changes are incorporated in the following patches:
242      Solaris/SunOS 5.8_sparc      patch 109147-03
243      Solaris/SunOS 5.8_x86       patch 109148-03
244      Solaris/SunOS 5.7_sparc      patch 106950-11
245      Solaris/SunOS 5.7_x86       patch 106951-11
246      Solaris/SunOS 5.6_sparc      patch 107733-08
247      Solaris/SunOS 5.6_x86       patch 107734-08
248 -----
249 4338812 crle(1) omits entries in the directory cache
250 4341496 RFE: provide a static version of /usr/bin/crle
251 4340878 rtld should treat $ORIGIN like LD_LIBRARY_PATH in security issues
252 -----
253 All the above changes are incorporated in the following patches:
254      Solaris/SunOS 5.8_sparc      patch 109147-04
255      Solaris/SunOS 5.8_x86       patch 109148-04
256      Solaris/SunOS 5.7_sparc      patch 106950-12
257      Solaris/SunOS 5.7_x86       patch 106951-12
258 -----

```

259 4349563 auxiliary filter error handling regression introduced in 4165487
 260 4355795 ldd -r now gives "displacement relocated" warnings
 261 -----
 262 All the above changes are incorporated in the following patches:
 263 Solaris/SunOS 5.7_sparc patch 106950-13
 264 Solaris/SunOS 5.7_x86 patch 106951-13
 265 Solaris/SunOS 5.6_sparc patch 107733-09
 266 Solaris/SunOS 5.6_x86 patch 107734-09
 267 -----
 268 4210412 versioning a static executable causes ld to core dump
 269 4219652 Linker gives misleading error about not finding main (xarch=v9)
 270 4103449 ld command needs a command line flag to force 64-bits
 271 4187211 problem with RDISP32 linking in copy-relocated objects
 272 4287274 dladdr, dlinfo do not provide the full path name of a shared object
 273 4297563 dlclose still does not remove all objects.
 274 4250694 rtld_db needs a new auxvec entry
 275 4235315 new features for rtld_db (DT_CHECKSUM, dynamic linked .o files
 276 4303609 64bit libelf.so.1 does not properly implement elf_hash()
 277 4310901 su.static fails when OSNet build with lazy-loading
 278 4310324 elf_errno() causes Bus Error(coredump) in 64-bit multithreaded programs
 279 4306415 ld core dump
 280 4316531 BCP: possible failure with dlclose/_preeexec_exit_handlers
 281 4313765 LD_BREADTH should be shot
 282 4318162 crle uses automatic strings in putenv.
 283 4255943 Description of -t option incomplete.
 284 4322528 sgs message test infrastructure needs improvement
 285 4239213 Want an API to obtain linker's search path
 286 4324134 use of extern mapfile directives can contribute unused symbols
 287 4322581 ELF data structures could be layed out more efficiently...
 288 4040628 Unnecessary section header symbols should be removed from .dynsym
 289 4300018 rtld: bindlock should be freed before calling call_fini()
 290 4336102 dlclose with non-deletable objects can mishandle dependencies
 291 4329785 mixing of SHT_SUNW_COMDAT & SHF_ORDERED causes ld to seg fault
 292 4334617 COPY relocations should be produces for references to .bss symbols
 293 4248250 relocation of local ABS symbols incorrect
 294 4335801 For complimentary alignments eliminate ld: warning: symbol 'll'
 295 has differing a
 296 4336980 ld.so.1 relative path processing revisited
 297 4243097 lderror(3DL) is not affected by setlocale(3C).
 298 4344528 dump should remove -D and -l usage message
 299 xxxxxxxx enable LD_ALTEXEC to access alternate link-editor
 300 -----
 301 All the above changes are incorporated in the following patches:
 302 Solaris/SunOS 5.8_sparc patch 109147-06
 303 Solaris/SunOS 5.8_x86 patch 109148-06
 304 -----
 306 -----
 307 Solaris 8 101 (3rd Q-update - s28u3)
 308 -----
 309 Bugid Risk Synopsis
 310 ======
 311 4346144 link-auditing: plt_tracing fails if LA_SYMB_NOPLTENTER given after
 312 being bound
 313 4346001 The ld should support mapfile syntax to generate PT_SUNWSTACK segment
 314 4349137 rtld_db: A third fallback method for locating the linkmap
 315 4343417 dladdr interface information inadequate
 316 4343801 RFE: crle(1): provide option for updating configuration files
 317 4346615 ld.so.1 attempting to open a directory gives: No such device
 318 4352233 crle should not honor umask
 319 4352330 LD_PRELOAD cannot use absolute path for privileged program
 320 4357805 RFE: man page for ld(1) does not document all -z or -B options in
 321 Solaris 8 9/00
 322 4358751 ld.so.1: LD_XXX environ variables and LD_FLAGS should be synchronized.
 323 4358862 link editors should reference "64" symlinks instead of sparcv9 (ia64).
 324 4356879 PLTs could use faster code sequences in some cases

325 4367118 new fast baplt's fail when traversed twice in threaded application
 326 4366905 Need a way to determine path to a shared library
 327 4351197 nfs performance problem by 103627-13
 328 4367405 LD_LIBRARY_PATH_64 not being used
 329 4354500 SHF_ORDERED ordered sections does not properly sort sections
 330 4369068 ld(1)'s weak symbol processing is inefficient (slow and doesn't scale).
 331 -----
 332 All the above changes are incorporated in the following patches:
 333 Solaris/SunOS 5.8_sparc patch 109147-07
 334 Solaris/SunOS 5.8_x86 patch 109148-07
 335 Solaris/SunOS 5.7_sparc patch 106950-14
 336 Solaris/SunOS 5.7_x86 patch 106951-14
 337 -----
 339 -----
 340 Solaris 8 701 (5th Q-update - s28u5)
 341 -----
 342 Bugid Risk Synopsis
 343 ======
 344 4368846 ld(1) fails to version some interfaces given in a mapfile
 345 4077245 dump core dump on null pointer.
 346 4372554 elfdump should demangle symbols (like nm, dump)
 347 4371114 dlclose may unmap a promiscuous object while it's still in use.
 348 4204447 elfdump should understand SHN_AFTER/SHN_BEGIN macro
 349 4377941 initialization of interposers may not occur
 350 4381116 ld/ld.so.1 could aid in detecting unused dependencies
 351 4381783 dlopen/dlclose of a libCrunch+libthread can dump core
 352 4385402 linker & run-time linker must support GABI ELF updates
 353 4394698 ld.so.1 does not process DF_SYMBOLIC - not gABI conforming
 354 4394212 the link editor quietly ignores missing support libraries
 355 4390308 ld.so.1 should provide more flexibility LD_PRELOAD'ing 32-bit/64-bit
 356 objects
 357 4401232 crle(1) could provide better flexibility for alternatives
 358 4401815 fix misc nits in debugging output...
 359 4402861 cleanup /usr/demo/link_audit & /usr/tmp/librtld_db demo source code...
 360 4393044 elfdump should allow raw dumping of sections
 361 4413168 SHF_ORDERED bit causes linker to generate a separate section
 362 -----
 363 All the above changes are incorporated in the following patches:
 364 Solaris/SunOS 5.8_sparc patch 109147-08
 365 Solaris/SunOS 5.8_x86 patch 109148-08
 366 -----
 367 4452202 Typos in <sys/link.h>
 368 4452200 dump doesn't support RUNPATH
 369 -----
 370 All the above changes are incorporated in the following patches:
 371 Solaris/SunOS 5.8_sparc patch 109147-09
 372 Solaris/SunOS 5.8_x86 patch 109148-09
 373 -----
 375 -----
 376 Solaris 8 1001 (6th Q-update - s28u6)
 377 -----
 378 Bugid Risk Synopsis
 379 ======
 380 4421842 fixups in SHT_GROUP processing required...
 381 4450433 problem with liblddbg output on -Dsection,detail when
 382 processing SHF_LINK_ORDER
 383 -----
 384 All the above changes are incorporated in the following patches:
 385 Solaris/SunOS 5.8_sparc patch 109147-10
 386 Solaris/SunOS 5.8_x86 patch 109148-10
 387 Solaris/SunOS 5.7_sparc patch 106950-15
 388 Solaris/SunOS 5.7_x86 patch 106951-15
 389 -----
 390 4463473 pldd showing wrong output

```

391 -----
392 All the above changes are incorporated in the following patches:
393   Solaris/SunOS 5.8_sparc      patch 109147-11
394   Solaris/SunOS 5.8_x86       patch 109148-11
395 ----

397 -----
398 Solaris 8 202 (7th Q-update - s28u7)
399 -----
400 Bugid Risk Synopsis
401 -----
402 4488954 ld.so.1 reuses same buffer to send ummapping range to
403   _preexec_exit_handlers()
404 -----
405 All the above changes are incorporated in the following patches:
406   Solaris/SunOS 5.8_sparc      patch 109147-12
407   Solaris/SunOS 5.8_x86       patch 109148-12
408 ----

410 -----
411 Solaris 9
412 -----
413 Bugid Risk Synopsis
414 -----
415 4505289 incorrect handling of _START_ and _END_
416 4506164 mcs does not recognize #linkbefore or #linkafter qualifiers
417 4447560 strip is creating unexecutable files...
418 4513842 library names not in ld.so string pool cause corefile bugs
419 -----
420 All the above changes are incorporated in the following patches:
421   Solaris/SunOS 5.8_sparc      patch 109147-13
422   Solaris/SunOS 5.8_x86       patch 109148-13
423   Solaris/SunOS 5.7_sparc      patch 106950-16
424   Solaris/SunOS 5.7_x86       patch 106951-16
425 ----

426 4291384 ld -M with a mapfile does not properly align Fortran REAL*8 data
427 4413322 SunOS 5.9 librtld_db doesn't show dlopened ".o" files anymore?
428 4429371 librtld_db busted on ia32 with SC6.x compilers...
429 4418274 elfdump dumps core on invalid input
430 4432224 libelf xlate routines are out of date
431 4433643 Memory leak using dlopen()/dlclose() in Solaris 8
432 4446564 ldd/lddstub - core dump conditions
433 4446115 translating SUNW_move sections is broken
434 4450225 The rdb command can fall into an infinite loop
435 4448531 Linker Causes Segmentation Fault
436 4453241 Regression in 4291384 can result in empty symbol table.
437 4453398 invalid runpath token can cause ld to spin.
438 4460230 ld (for OS 5.8 and 5.9) loses error message
439 4462245 ld.so.1 core dumps when executed directly...
440 4455802 need more flexibility in establishing a support library for ld
441 4467068 dyn_plt_entsize not properly initialized in ld.so.1
442 4468779 elf_plt_trace_write() broken on i386 (link-auditing)
443 4465871 -zld32 and -zld64 does not work the way it should
444 4461890 bad shared object created with -zredlocsym
445 4469400 ld.so.1: is_so_loaded isn't as efficient as we thought...
446 4469566 lazy loading fallback can reference un-relocated objects
447 4470493 libelf incorrectly translates NOTE sections across architectures...
448 4469684 rtld leaks dl_handles and permits on dlopen/dlclose
449 4475174 ld.so.1 prematurely reports the failure to load a object...
450 4475514 ld.so.1 can core dump in memory allocation fails (no swap)
451 4481851 Setting ld.so.1 environment variables globally would be useful
452 4482035 setting LD_PROFILE & LD_AUDIT causes ping command to issue warnings
453   on 5.8
454 4377735 segment reservations cause sbrk() to fail
455 4491434 ld.so.1 can leak file-descriptors when loading same named objects
456 4289232 some of warning/error/debugging messages from libld.so can be revised

```

```

457 4462748 Linker Portion of TLS Support
458 4496718 run-time linkers mutex_locks not working with ld libc interface
459 4497270 The -zredlocsym option should not eliminate partially initialized local
460   symbols
461 4496963 dumping an object with crle(1) that uses $ORIGIN can lose its
462   dependencies
463 4499413 Sun linker orders of magnitude slower than gnu linker
464 4461760 lazy loading libxm and libxt can fail.
465 4469031 The partial initialized (local) symbols for intel platform is not
466   working.
467 4492883 Add link-editor option to multi-pass archives to resolve unsatisfied
468   symbols
469 4503731 linker-related commands misspell "argument"
470 4503768 whocalls(1) should output messages to stderr, not stdout
471 4503748 whocalls(1) usage message and manpage could be improved
472 4503625 nm should be taught about TLS symbols - that they aren't allowed that is
473 4300120 segment address validation is too simplistic to handle segment
474   reservations
475 4404547 krtld/reloc.h could have better error message, has typos
476 4270931 R_SPARC_HIX22 relocation is not handled properly
477 4485320 ld needs to support more the 32768 PLTs
478 4516434 sotruss can not watch libc_psr.so.1
479 4213100 sotruss could use more flexible pattern matching
480 4503457 ld seg fault with comdat
481 4510264 sections with SHF_TLS can come in different orders...
482 4518079 link-editor support library unable to modify section header flags
483 4515913 ld.so.1 can incorrectly decrement external reference counts on dlclose()
484 4519569 ld -V does not return a interesting value...
485 4524512 ld.so.1 should allow alternate termination signals
486 4524767 elfdump dies on bogus sh_name fields...
487 4524735 ld getopt processing of '-' changed
488 4521931 subroutine in a shared object as LOCL instead of GLOB
489 -----
490 All the above changes are incorporated in the following patches:
491   Solaris/SunOS 5.8_sparc      patch 109147-14
492   Solaris/SunOS 5.8_x86       patch 109148-14
493   Solaris/SunOS 5.7_sparc      patch 106950-17
494   Solaris/SunOS 5.7_x86       patch 106951-17
495 -----
496 4532729 tentative definition of TLS variable causes linker to dump core
497 4526745 fixup ld error message about duplicate dependencies/needed names
498 4522999 Solaris linker one order of magnitude slower than GNU linker
499 4518966 dldump undoes existing relocations with no thought of alignment or size.
500 4587441 Certain libraries have race conditions when setting error codes
501 4523798 linker option to align bss to large pagesize alignments.
502 4524008 ld can improperly set st_size of symbols named "_init" or "_fini"
503 4619282 ld cannot link a program with the option -sb
504 4620846 Perl Configure probing broken by ld changes
505 4621122 multiple ld '-zinitarray=' on a commandline fails
506 -----
507   Solaris/SunOS 5.8_sparc      patch 109147-15
508   Solaris/SunOS 5.8_x86       patch 109148-15
509   Solaris/SunOS 5.7_sparc      patch 106950-18
510   Solaris/SunOS 5.7_x86       patch 106951-18
511   Solaris/SunOS 5.6_sparc      patch 107733-10
512   Solaris/SunOS 5.6_x86       patch 107734-10
513 -----
514 All the above changes plus:
515   4616944 ar seg faults when order of object file is reversed.
516 are incorporated in the following patches:
517   Solaris/SunOS 5.8_sparc      patch 109147-16
518   Solaris/SunOS 5.8_x86       patch 109148-16
519 -----
520 All the above changes plus:
521   4872634 Large LD_PRELOAD values can cause SEGV of process
522 are incorporated in the following patches:

```

```

523      Solaris/SunOS 5.6_sparc      patch T107733-11
524      Solaris/SunOS 5.6_x86       patch T107734-11
525 -----
527 -----
528 Solaris 9 1202 (2nd Q-update - s9u2)
529 -----
530 Bugid Risk Synopsis
531 =====
532 4546416 add help messages to ld.so mdbmodule
533 4526752 we should build and ship ld.so's mdb module
534 4624658 update 386 TLS relocation values
535 4622472 LA_SYMB_DLSYM not set for la_bind() invocations
536 4638070 ldd/ld.so.1 could aid in detecting unreferenced dependencies
537 PSARC/2002/096 Detecting unreferenced dependencies with ldd(1)
538 4633860 Optimization for unused static global variables
539 PSARC/2002/113 ld -zignore - section elimination
540 4642829 ld.so.1 mprotect()'s text segment for weak relocations (it shouldn't)
541 4621479 'make' in $SRC/cmd/sgs/tools tries to install things in the proto area
542 4529912 purge ia64 source from sgs
543 4651709 dlopen(RTLD_NOLOAD) can disable lazy loading
544 4655066 crle: -u with nonexistent config file doesn't work
545 4654406 string tables created by the link-editor could be smaller...
546 PSARC/2002/160 ld -znocompstrtab - disable string-table compression
547 4651493 RTLD_NOW can result in binding to an object prior to its init being run.
548 4662575 linker displacement relocation checking introduces significant
549 linker overhead
550 4533195 ld interposes on malloc()/free() preventing support library from freeing
551 memory
552 4630224 crle get's confused about memory layout of objects...
553 4664855 crle on application failed with ld.so.1 encountering mmap() returning
554 ENOMEM err
555 4669582 latest dynamic linker causes libthread _init to get skipped
556 4671493 ld.so.1 inconsistently assigns PATHNAME() on primary objects
557 4668517 compile with map.bssalign doesn't copy _lob to bss
558 -----
559 All the above changes are incorporated in the following patches:
560 Solaris/SunOS 5.9_sparc      patch T112963-01
561 Solaris/SunOS 5.8_sparc      patch T109147-17
562 Solaris/SunOS 5.8_x86       patch T109148-17
563 -----
564 4701749 On Solaris 8 + 109147-16 ld crashes when building a dynamic library.
565 4707808 The ldd command is broken in the latest 2.8 linker patch.
566 -----
567 All the above changes are incorporated in the following patches:
568 Solaris/SunOS 5.9_sparc      patch T112963-02
569 Solaris/SunOS 5.8_sparc      patch T109147-18
570 Solaris/SunOS 5.8_x86       patch T109148-18
571 -----
572 4696204 enable extended section indexes in relocatable objects
573 PSARC/2001/332 ELF gABI updates - round II
574 PSARC/2002/369 libelf interfaces to support ELF Extended Sections
575 4706503 linkers need to cope with EF_SPARCV9_PSO/EF_SPARCV9_RMO
576 4716929 updating of local register symbols in dynamic symtab busted...
577 4710814 add "official" support for the "symbolic" keyword in linker map-file
578 PSARC/2002/439 linker mapfile visibility declarations
579 -----
580 All the above changes are incorporated in the following patches:
581 Solaris/SunOS 5.9_sparc      patch T112963-03
582 Solaris/SunOS 5.8_sparc      patch T109147-19
583 Solaris/SunOS 5.8_x86       patch T109148-19
584 Solaris/SunOS 5.7_sparc      patch T106950-19
585 Solaris/SunOS 5.7_x86       patch T106951-19
586 -----
588 -----

```

```

589 Solaris 9 403 (3rd Q-update - s9u3)
590 -----
591 Bugid Risk Synopsis
592 =====
593 4731174 strip(1) does not fixup SHT_GROUP data
594 4733697 -zignore with gcc may exclude C++ exception sections
595 4733317 R_SPARC_*_HIX22 calculations are wrong with 32bit LD building
596 ELF64 binaries
597 4735165 fatal linker error when compiling C++ programs with -xlinkopt
598 4736951 The mcs broken when the target file is an archive file
599 -----
600 All the above changes are incorporated in the following patches:
601 Solaris/SunOS 5.8_sparc      patch T109147-20
602 Solaris/SunOS 5.8_x86       patch T109148-20
603 Solaris/SunOS 5.7_sparc      patch T106950-20
604 Solaris/SunOS 5.7_x86       patch T106951-20
605 -----
606 4739660 Threads deadlock in schedlock and dynamic linker lock.
607 4653148 ld.so.1/libc should unregister its dlclose() exit handler via a fini.
608 4743413 ld.so.1 doesn't terminate argv with NULL pointer when invoked directly
609 4746231 linker core-dumps when SECTION relocations are made against discarded
610 sections
611 4730433 ld.so.1 wastes time repeatedly opening dependencies
612 4744337 missing RD_CONSISTENT event with dlopen(LD_ID_NEWM, ...)
613 4670835 rd_load_objiter can ignore callback's return value
614 4745932 strip utility doesn't strip out Dwarf2 debug section
615 4754751 "strip" command doesn't remove comdat stab sections.
616 4755674 Patch 109147-18 results in coredump.
617 -----
618 All the above changes are incorporated in the following patches:
619 Solaris/SunOS 5.9_sparc      patch T112963-04
620 Solaris/SunOS 5.7_sparc      patch T106950-21
621 Solaris/SunOS 5.7_x86       patch T106951-21
622 -----
623 4772927 strip core dumps on an archive library
624 4774727 direct-bindings can fail against copy-reloc symbols
625 -----
626 All the above changes are incorporated in the following patches:
627 Solaris/SunOS 5.9_sparc      patch T112963-05
628 Solaris/SunOS 5.9_x86       patch T113986-01
629 Solaris/SunOS 5.8_sparc      patch T109147-21
630 Solaris/SunOS 5.8_x86       patch T109148-21
631 Solaris/SunOS 5.7_sparc      patch T106950-22
632 Solaris/SunOS 5.7_x86       patch T106951-22
633 -----
635 -----
636 Solaris 9 803 (4th Q-update - s9u4)
637 -----
638 Bugid Risk Synopsis
639 =====
640 4730110 ld.so.1 list implementation could scale better
641 4728822 restrict the objects dlsym() searches.
642 PSARC/2002/478 New dlopen(3dl) flag - RTLD_FIRST
643 4714146 crle: 64-bit secure pathname is incorrect.
644 4504895 dlclose() does not remove all objects
645 4698800 Wrong comments in /usr/lib/ld/sparcv9/map.*.
646 4745129 dldump is inconsistent with .dynamic processing errors.
647 4753066 LD_SIGNAL isn't very useful in a threaded environment
648 PSARC/2002/569 New dlinfo(3dl) flag - RTLD_DI_SIGNAL
649 4765536 crle: symbolic links can confuse alternative object configuration info
650 4766815 ld -r of object the TLS data fails
651 4770484 elfdump can not handle stripped archive file
652 4770494 The ld command gives improper error message handling broken archive
653 4775738 overwriting output relocation table when 'ld -zignore' is used
654 4778247 elfdump -e of core files fails

```

```

655 4779976 elfdump dies on bad relocation entries
656 4785759 invalid SHT_GROUP entries can cause linker to seg fault
657 4783869 disclose: filter closure exhibits hang/failure - introduced with 4504895
658 4778418 ld.so.1: there be nits out there
659 4792461 Thread-Local Storage - x86 instruction sequence updates
660  PSARC/2002/746 Thread-Local Storage - x86 instruction sequence updates
661 4461340 sgs: ugly build output while suppressing ia64 (64-bit) build on Intel
662 4790194 dlopen(..., RTLD_GROUP) has an odd interaction with interposition
663 4804328 auditing of threaded applications results in deadlock
664 4806476 building relocatable objects with SHF_EXCLUDE loses relocation
665  information
666 -----
667 All the above changes are incorporated in the following patches:
668 Solaris/SunOS 5.9_sparc patch T112963-06
669 Solaris/SunOS 5.9_x86 patch T113986-02
670 Solaris/SunOS 5.8_sparc patch T109147-22
671 Solaris/SunOS 5.8_x86 patch T109148-22
672 -----
673 4731183 compiler creates .tbsbss section instead of .tbss as documented
674 4816378 TLS: a tls test case dumps core with C and C++ compilers
675 4817314 TLS_GD relocations against local symbols do not reference symbol...
676 4811951 non-default symbol visibility overridden by definition in shared object
677 4802194 relocation error of mozilla built by K2 compiler
678 4715815 ld should allow linking with no output file (or /dev/null)
679 4793721 Need a way to null all code in ISV objects enabling ld performance
680 tuning
681 -----
682 All the above changes plus:
683 4796237 RFE: link-editor became extremely slow with patch 109147-20 and
684 static libraries
685 are incorporated in the following patches:
686 Solaris/SunOS 5.9_sparc patch T112963-07
687 Solaris/SunOS 5.9_x86 patch T113986-03
688 Solaris/SunOS 5.8_sparc patch T109147-23
689 Solaris/SunOS 5.8_x86 patch T109148-23
690 -----
692 -----
693 Solaris 9 1203 (5th Q-update - s9u5)
694 -----
695 Bugid Risk Synopsis
696 -----
697 4830584 mmap for the padding region doesn't get freed after dlclose
698 4831650 ld.so.1 can walk off the end of its call_init() array...
699 4831544 ldd using .so modules compiled with FD7 compiler caused a core dump
700 4834784 Accessing members in a TLS structure causes a core dump in Oracle
701 4824026 segv when -z combreloc is used with -xlinkopt
702 4825296 typo in elfdump
703 -----
704 All the above changes are incorporated in the following patches:
705 Solaris/SunOS 5.9_sparc patch T112963-08
706 Solaris/SunOS 5.9_x86 patch T113986-04
707 Solaris/SunOS 5.8_sparc patch T109147-24
708 Solaris/SunOS 5.8_x86 patch T109148-24
709 -----
710 4470917 Solaris Process Model Unification (link-editor components only)
711  PSARC/2002/117 Solaris Process Model Unification
712 4744411 Bloomberg wants a faster linker.
713 4811969 64-bit links can be much slower than 32-bit.
714 4825065 ld(1) should ignore consecutive empty sections.
715 4838226 unrelocated shared objects may be erroneously collected for init firing
716 4830889 TLS: testcase core dumps with -xarch=v9 and -g
717 4845764 filter removal can leave dangling filtee pointer
718 4811093 apctrace -F libc date core dumps
719 4826315 Link editors need to be pre- and post- Unified Process Model aware
720 4868300 interposing on direct bindings can fail

```

```

721 4872634 Large LD_PRELOAD values can cause SEGV of process
722 -----
723 All the above changes are incorporated in the following patches:
724 Solaris/SunOS 5.9_sparc patch T112963-09
725 Solaris/SunOS 5.9_x86 patch T113986-05
726 Solaris/SunOS 5.8_sparc patch T109147-25
727 Solaris/SunOS 5.8_x86 patch T109148-25
728 -----
730 -----
731 Solaris 9 404 (6th Q-update - s9u6)
732 -----
733 Bugid Risk Synopsis
734 =====
735 4870260 The elfdump command should produce more warning message on invalid move
736 entries.
737 4865418 empty PT_TLS program headers cause problems in TLS enabled applications
738 4825151 compiler core dumped with a -mt -xF%all test
739 4845829 The runtime linker fails to dlopen() long path name.
740 4900684 shared libraries with more than 32768 plt's fail for sparc ELF64
741 4906062 Makefiles under usr/src/cmd/sgs needs to be updated
742 -----
743 All the above changes are incorporated in the following patches:
744 Solaris/SunOS 5.9_sparc patch T112963-10
745 Solaris/SunOS 5.9_x86 patch T113986-06
746 Solaris/SunOS 5.8_sparc patch T109147-26
747 Solaris/SunOS 5.8_x86 patch T109148-26
748 Solaris/SunOS 5.7_sparc patch T106950-24
749 Solaris/SunOS 5.7_x86 patch T106951-24
750 -----
751 4900320 rtld library mapping could be faster
752 4911775 implement GOTDATA proposal in ld
753 PSARC/2003/477 SPARC GOTDATA instruction sequences
754 4904565 Functionality to ignore relocations against external symbols
755 4764817 add section types SHT_DEBUG and SHT_DEBUGSTR
756 PSARC/2003/510 New ELF DEBUG and ANNOTATE sections
757 4850703 enable per-symbol direct bindings
758 4716275 Help required in the link analysis of runtime interfaces
759 PSARC/2003/519 Link-editors: Direct Binding Updates
760 4904573 elfdump may hang when processing archive files
761 4918310 direct binding from an executable can't be interposed on
762 4918938 ld.so.1 has become SPARC32PLUS - breaks 4.x binary compatibility
763 4911796 S1S8 C++: ld dump core when compiled and linked with xlinkopt=1.
764 4889914 ld crashes with SEGV using -M mapfile under certain conditions
765 4911936 exception are not catch from shared library with -zignore
766 -----
767 All the above changes are incorporated in the following patches:
768 Solaris/SunOS 5.9_sparc patch T112963-11
769 Solaris/SunOS 5.9_x86 patch T113986-07
770 Solaris/SunOS 5.8_sparc patch T109147-27
771 Solaris/SunOS 5.8_x86 patch T109148-27
772 Solaris/SunOS 5.7_sparc patch T106950-25
773 Solaris/SunOS 5.7_x86 patch T106951-25
774 -----
775 4946992 ld crashes due to huge number of sections (>65,000)
776 4951840 mcs -c goes into a loop on executable program
777 4939869 Need additional relocation types for abs34 code model
778 PSARC/2003/684 abs34 ELF relocations
779 -----
780 All the above changes are incorporated in the following patches:
781 Solaris/SunOS 5.9_sparc patch T112963-12
782 Solaris/SunOS 5.9_x86 patch T113986-08
783 Solaris/SunOS 5.8_sparc patch T109147-28
784 Solaris/SunOS 5.8_x86 patch T109148-28
785 -----

```

```

787 -----
788 Solaris 9 904 (7th Q-update - s9u7)
789 -----
790 Bugid Risk Synopsis
791 =====
792 4912214 Having multiple of libc.so.1 in a link map causes malloc() to fail
793 4926878 ld.so.1 should pass MAP_ALIGN flag to give kernel more flexibility
794 4930997 sgs bld_vernole.ksh script needs to be hardend...
795 4796286 ld.so.1: scenario for trouble?
796 4930985 clean up cruft under usr/src/cmd/sgs/tools
797 4933300 remove references to Ultra-1 in librtld_db demo
798 4936305 string table compression is much too slow...
799 4939626 SUNWorld internal package must be updated...
800 4939565 per-symbol filtering required
801 4948119 ld(1) -z loadfltr fails with per-symbol filtering
802 4948427 ld.so.1 gives fatal error when multiple RTLDINFO objects are loaded
803 4940894 ld core dumps using "-xldscope=symbolic"
804 4955373 per-symbol filtering refinements
805 4878827 crle(1M) - display post-UPM search paths, and compensate for pre-UPM.
806 4955802 /usr/ccs/bin/ld dumps core in process_reld()
807 4964415 elfdump issues wrong relocation error message
808 4966455 LD_NOAUXFLTR fails when object is both a standard and auxiliary filter
809 4973865 the link-editor does not scale properly when linking objects with
810 lots of syms
811 4975598 SHT_SUNW_ANNNOTE section relocation not resolved
812 4974828 nss_files nss_compat_r_mt tests randomly segfaulting
813 -----
814 All the above changes are incorporated in the following patches:
815   Solaris/SunOS 5.9_sparc      patch T112963-13
816   Solaris/SunOS 5.9_x86       patch T113986-09
817 -----
818 4860508 link-editors should create/promote/verify hardware capabilities
819 5002160 crle: reservation for dumped objects gets confused by mmaped object
820 4967869 linking stripped library causes segv in linker
821 5006657 link-editor doesn't always handle nondirect binding syminfo information
822 4915901 no way to see ELF information
823 5021773 ld.so.1 has trouble with objects having more than 2 segments.
824 -----
825 All the above changes are incorporated in the following patches:
826   Solaris/SunOS 5.9_sparc      patch T112963-14
827   Solaris/SunOS 5.9_x86       patch T113986-10
828   Solaris/SunOS 5.8_sparc      patch T109147-29
829   Solaris/SunOS 5.8_x86       patch T109148-29
830 -----
831 All the above changes plus:
832   6850124 dlopen reports "No such file or directory" in spite of ENOMEM
833   when mmap fails in anon_map()
834 are incorporated in the following patches:
835   Solaris/SunOS 5.9_sparc      patch TXXXXXX-XX
836   Solaris/SunOS 5.9_x86       patch TXXXXXX-XX
837 -----
839 -----
840 Solaris 10
841 -----
842 Bugid Risk Synopsis
843 =====
844 5044797 ld.so.1: secure directory testing is being skipped during filtee
845 processing
846 4963676 Remove remaining static libraries
847 5021541 unnecessary PT_SUNWBSS segment may be created
848 5031495 elfdump complains about bad symbol entries in core files
849 5012172 Need error when creating shared object with .o compiled
850   -xarch=v9 -xcode=abs44
851 4994738 rd_plt_resolution() resolves ebx-relative PLT entries incorrectly
852 5023493 ld -m output with patch 109147-25 missing .o information

```

```

853 -----
854 All the above changes are incorporated in the following patches:
855   Solaris/SunOS 5.9_sparc      patch T112963-15
856   Solaris/SunOS 5.9_x86       patch T113986-11
857   Solaris/SunOS 5.8_sparc      patch T109147-30
858   Solaris/SunOS 5.8_x86       patch T109148-30
859 -----
860 5071614 109147-29 & -30 break the build of on28-patch on Solaris 8 2/04
861 5029830 crle: provide for optional alternative dependencies.
862 5034652 ld.so.1 should save, and print, more error messages
863 5036561 ld.so.1 outputs non-fatal fatal message about auxiliary filter libraries
864 5042713 4866170 broke ld.so's ::setenv
865 5047082 ld can core dump on bad gcc objects
866 5047612 ld.so.1: secure pathname verification is flawed with filter use
867 5047235 elfdump can core dump printing PT_INTERP section
868 4798376 nits in demo code
869 5041446 gelf_update_*() functions inconsistently return NULL or 0
870 5032364 M_ID_TLSBSS and M_ID_UNKNOWN have the same value
871 4707030 Empty LD_PRELOAD_64 doesn't override LD_PRELOAD
872 4968618 symbolic linkage causes core dump
873 5062313 dladdr() can cause deadlock in MT apps.
874 5056867 $ISALIST/$HWCAP expansion should be more flexible.
875 4918303 @0.so.1 should not use compiler-supplied crt*.o files
876 5058415 whocalls cannot take more than 10 arguments
877 5067518 The fix for 4918303 breaks the build if a new work space is used.
878 -----
879 All the above changes are incorporated in the following patches:
880   Solaris/SunOS 5.9_sparc      patch T112963-16
881   Solaris/SunOS 5.9_x86       patch T113986-12
882   Solaris/SunOS 5.8_sparc      patch T109147-31
883   Solaris/SunOS 5.8_x86       patch T109148-31
884 -----
885 5013759 *file* should report hardware/software capabilities (link-editor
886 components only)
887 5063580 libldstab: file /tmp/posto...: .stab[.index].sbfocus] found with no
888 matching stri
889 5076838 elfdump(1) is built with a CTF section (the wrong one)
890 5080344 Hardware capabilities are not enforced for a.out
891 5079061 RTLD_DEFAULT can be expensive
892 5086352 PSARC/2004/747 New dlsym(3c) Handle - RTLD_PROBE
893 5064973 allow normal relocs against TLS symbols for some sections
894 5085792 LD_XXXX_64 should override LD_XXXX
895 5096272 every executable or library has a .SUNW_dof section
896 5094135 Bloomberg wants a faster ldd.
897 5086352 libld.so.3 should be built with a .SUNW_ctf ELF section, ready for CR
898 5098205 elfdump gives wrong section name for the global offset table
899 5092414 Linker patch 109147-29 makes Broadvision One-To-One server v4.1
900 installation fail
901 5080256 dump(1) doesn't list ELF hardware capabilities
902 5097347 recursive read lock in gelf_getsym()
903 -----
904 All the above changes are incorporated in the following patches:
905   Solaris/SunOS 5.9_sparc      patch T112963-17
906   Solaris/SunOS 5.9_x86       patch T113986-13
907   Solaris/SunOS 5.8_sparc      patch T109147-32
908   Solaris/SunOS 5.8_x86       patch T109148-32
909 -----
910 5106206 ld.so.1 fail to run a Solaris9 program that has libc linked with
911   -z lazyload
912 5102601 ON should deliver a 64-bit operating system for Opteron systems
913   (link-editor components only)
914 6173852 enable link_auditing technology for amd64
915 6174599 linker does not create .eh_frame_hdr sections for eh_frame sections
916   with SHF_LINK_ORDER
917 6175609 amd64 run-time linker has a corrupted note section
918 6175843 amd64 rdb_demo files not installed

```

```

919 6182293 ld.so.1 can repeatedly relocate object .plt (RTLD_NOW).
920 6183645 ld core dumps when automounter fails
921 6178667 ldd list unexpected (file not found) in x86 environment.
922 6181928 Need new reloc types R_AMD64_GOTOFF64 and R_AMD64_GOTPC32
923 6182884 AMD64: ld coreldumps when building a shared library
924 6173559 The ld may set incorrect value for sh_addralign under some conditions.
925 5105601 ld.so.1 gets a little too enthusiastic with interposition
926 6189384 ld.so.1 should accommodate a files dev/inode change (libc loopback mnt)
927 6177838 AMD64: linker cannot resolve PLT for 32-bit a.out(s) on amd64-S2 kernel
928 6190863 sparc disassembly code should be removed from rdb_demo
929 6191488 unwind eh_frame_hdr needs corrected encoding value
930 6192490 moe(1) returns /lib/libc.so.1 for optimal expansion of libc HWCAP
931 6192164 libraries
932 6192164 AMD64: introduce dlamdd64getunwind interface
933 6195030 PSARC/2004/747 libc::dlamdd64getunwind()
934 6195030 libdl has bad version name
935 6195521 64-bit moe(1) missed the train
936 6198358 AMD64: bad eh_frame_hdr data when C and C++ mixed in a.out
937 6204123 ld.so.1: symbol lookup fails even after lazy loading fallback
938 6207495 UNIX98/UNIX03 vsx namespace violation DYNL.hdr/misc/dlfcn/T.dlfcn
939 14 Failed
940 6217285 ctfmerge crashed during full onnv build
941 -----
943 -----
944 Solaris 10 106 (1st Q-update - s10u1)
945 -----
946 Bugid Risk Synopsis
947 =====
948 6209350 Do not include signature section from dynamic dependency library into
949 relocatable object
950 6212797 The binary compiled on SunOS4.x doesn't run on Solaris8 with Patch
951 109147-31
952 -----
953 All the above changes are incorporated in the following patches:
954 Solaris/SunOS 5.9_sparc patch T112963-18
955 Solaris/SunOS 5.9_x86 patch T113986-14
956 Solaris/SunOS 5.8_sparc patch T109147-33
957 Solaris/SunOS 5.8_x86 patch T109148-33
958 -----
959 6219538 112963-17: linker patch causes binary to dump core
960 -----
961 All the above changes are incorporated in the following patches:
962 Solaris/SunOS 5.10_sparc patch T117461-01
963 Solaris/SunOS 5.10_x86 patch T118345-01
964 Solaris/SunOS 5.9_sparc patch T112963-19
965 Solaris/SunOS 5.9_x86 patch T113986-15
966 Solaris/SunOS 5.8_sparc patch T109147-34
967 Solaris/SunOS 5.8_x86 patch T109148-34
968 -----
969 6257177 incremental builds of usr/src/cmd/sgs can fail...
970 6219651 AMD64: Linker does not issue error for out of range R_AMD64_PC32
971 -----
972 All the above changes are incorporated in the following patches:
973 Solaris/SunOS 5.10_sparc patch T117461-02
974 Solaris/SunOS 5.10_x86 patch T118345-02
975 Solaris/SunOS 5.9_sparc patch T112963-20
976 Solaris/SunOS 5.9_x86 patch T113986-16
977 Solaris/SunOS 5.8_sparc patch T109147-35
978 Solaris/SunOS 5.8_x86 patch T109148-35
979 NOTE: The fix for 6219651 is only applicable for 5.10_x86 platform.
980 -----
981 5080443 lazy loading failure doesn't clean up after itself (D)
982 6226206 ld.so.1 failure when processing single segment hwcap filtee
983 6228472 ld.so.1: link-map control list stacking can loose objects
984 6235000 random packages not getting installed in svn_09 and svn_10 -

```

```

985 rtld/common/malloc.c Assertion
986 6219317 Large page support is needed for mapping executables, libraries and
987 files (link-editor components only)
988 6244897 ld.so.1 can't run apps from commandline
989 6251798 moe(1) returns an internal assertion failure message in some
990 circumstances
991 6251722 ld fails silently with exit 1 status when -z ignore passed
992 6254364 ld won't build libgenunix.so with absolute relocations
993 6215444 ld.so.1 caches "not there" lazy libraries, foils svc.startd(1M)'s logic
994 6222525 dlsym(3C) trusts caller(), which may return wrong results with tail call
995 optimization
996 6241995 warnings in sgs should be fixed (link-editor components only)
997 6258834 direct binding availability should be verified at runtime
998 6260361 lari shouldn't count a.out non-zero undefined entries as interesting
999 6260780 ldd doesn't recognize LD_NOAUXFLTR
1000 6266261 Add ld(1) -Bnodirect support (D)
1001 6261990 invalid e_flags error could be a little more friendly
1002 6261803 lari(1) should find more events uninteresting (D)
1003 6267352 libd_malloc provides inadequate alignment
1004 6268693 SHN_SUNW_IGNORE symbols should be allowed to be multiply defined
1005 6262789 Infosys wants a faster linker
1006 -----
1007 All the above changes are incorporated in the following patches:
1008 Solaris/SunOS 5.10_sparc patch T117461-03
1009 Solaris/SunOS 5.10_x86 patch T118345-03
1010 Solaris/SunOS 5.9_sparc patch T112963-21
1011 Solaris/SunOS 5.9_x86 patch T113986-17
1012 Solaris/SunOS 5.8_sparc patch T109147-36
1013 Solaris/SunOS 5.8_x86 patch T109148-36
1014 -----
1015 6283601 The usr/src/cmd/sgs/packages/common/copyright contains old information
1016 legally problematic
1017 6276905 dlinfo gives inconsistent results (relative vs absolute linkname) (D)
1018 PSARC/2005/357 dlinfo(3c) RTLD_DI_ARGSINFO
1019 6284941 excessive link times with many groups/sections
1020 6280467 dlclose() unmaps shared library before library's _fini() has finished
1021 6291547 ld.so mishandles LD_AUDIT causing security problems.
1022 -----
1023 All the above changes are incorporated in the following patches:
1024 Solaris/SunOS 5.10_sparc patch T117461-04
1025 Solaris/SunOS 5.10_x86 patch T118345-04
1026 Solaris/SunOS 5.9_sparc patch T112963-22
1027 Solaris/SunOS 5.9_x86 patch T113986-18
1028 Solaris/SunOS 5.8_sparc patch T109147-37
1029 Solaris/SunOS 5.8_x86 patch T109148-37
1030 -----
1031 6295971 UNIX98/UNIX03 *vsx* DYNL.hdr/misc/dlfcn/T.dlfcn 14 fails, auxv.h syntax
1032 error
1033 6299525 .init order failure when processing cycles
1034 6273855 gcc and sgs/crle don't get along
1035 6273864 gcc and sgs/libld don't get along
1036 6273875 gcc and sgs/rtdld don't get along
1037 6272563 gcc and amd64/krtld/doreloc.c don't get along
1038 6290157 gcc and sgs/librtld_db/rdb_demo don't get along
1039 6301218 Matlab dumps core on startup when running on 112963-22 (D)
1040 -----
1041 All the above changes are incorporated in the following patches:
1042 Solaris/SunOS 5.10_sparc patch T117461-06
1043 Solaris/SunOS 5.10_x86 patch T118345-08
1044 Solaris/SunOS 5.9_sparc patch T112963-23
1045 Solaris/SunOS 5.9_x86 patch T113986-19
1046 Solaris/SunOS 5.8_sparc patch T109147-38
1047 Solaris/SunOS 5.8_x86 patch T109148-38
1048 -----
1049 6314115 Checkpoint refuses to start, crashes on start, after application of
1050 linker patch 112963-22

```

1051 -----
1052 All the above changes are incorporated in the following patches:
1053 Solaris/SunOS 5.9_sparc patch T112963-24
1054 Solaris/SunOS 5.9_x86 patch T113986-20
1055 Solaris/SunOS 5.8_sparc patch T109147-39
1056 Solaris/SunOS 5.8_x86 patch T109148-39
1057 -----
1058 6318306 a dlsym() from a filter should be redirected to an associated filtee
1059 6318401 mis-aligned TLS variable
1060 6324019 ld.so.1: malloc alignment is insufficient for new compilers
1061 6324589 psh coredumps on x86 machines on snv_23
1062 6236594 AMD64: Linker needs to handle the new .lbss section (D)
1063 PSARC 2005/514 AMD64 - large section support
1064 6314743 Linker: incorrect resolution for R_AMD64_GOTPC32
1065 6311865 Linker: x86 medium model; invalid ELF program header
1066 -----
1067 All the above changes are incorporated in the following patches:
1068 Solaris/SunOS 5.10_sparc patch T117461-07
1069 Solaris/SunOS 5.10_x86 patch T118345-12
1070 -----
1071 6309061 link_audit should use __asm__ with gcc
1072 6310736 gcc and sgs/libld don't get along on SPARC
1073 6329796 Memory leak with iconv_open/iconv_close with patch 109147-33
1074 6332983 s9 linker patches 112963-24/113986-20 causing cluster machines not
 to boot
1075 -----
1076 All the above changes are incorporated in the following patches:
1077 Solaris/SunOS 5.10_sparc patch T117461-08
1078 Solaris/SunOS 5.10_x86 patch T121208-02
1079 Solaris/SunOS 5.9_sparc patch T112963-25
1080 Solaris/SunOS 5.9_x86 patch T113986-21
1081 Solaris/SunOS 5.8_sparc patch T109147-40
1082 Solaris/SunOS 5.8_x86 patch T109148-40
1083 -----
1084 6445311 The sparc S8/S9/S10 linker patches which include the fix for the
 CR6222525 are hit by the CR6439613.
1085 -----
1086 All the above changes are incorporated in the following patches:
1087 Solaris/SunOS 5.9_sparc patch T112963-26
1088 Solaris/SunOS 5.8_sparc patch T109147-41
1089 -----
1090 6487273 ld.so.1 may open arbitrary locale files when relative path is built
 from locale environment vars
1091 6487284 ld.so.1: buffer overflow in doprf() function
1092 -----
1093 Solaris 10 807 (4th Q-update - s10u4)
1094 -----
1095 Bugid Risk Synopsis
1096 -----
1097 ======
1098 6487273 ld.so.1 may open arbitrary locale files when relative path is built
 from locale environment vars
1099 6487284 ld.so.1: buffer overflow in doprf() function
1100 -----
1101 All the above changes are incorporated in the following patches:
1102 Solaris/SunOS 5.10_sparc patch T124922-01
1103 Solaris/SunOS 5.10_x86 patch T124923-01
1104 Solaris/SunOS 5.9_sparc patch T112963-27
1105 Solaris/SunOS 5.9_x86 patch T113986-22
1106 Solaris/SunOS 5.8_sparc patch T109147-42
1107 Solaris/SunOS 5.8_x86 patch T109148-41
1108 -----
1109 6477132 ld.so.1: memory leak when running set*id application
1110 -----
1111 All the above changes are incorporated in the following patches:
1112 Solaris/SunOS 5.10_sparc patch T124922-02
1113 Solaris/SunOS 5.10_x86 patch T124923-02
1114 Solaris/SunOS 5.9_sparc patch T112963-30
1115 Solaris/SunOS 5.9_x86 patch T113986-24
1116 -----

1117 -----
1118 6340814 ld.so.1 core dump with HWCAP relocatable object + updated statistics
1119 6307274 crle bug with LD_LIBRARY_PATH
1120 6317969 elfheader limited to 65535 segments (link-editor components only)
1121 6350027 ld.so.1 aborts with assertion failed on amd64
1122 6362044 ld(1) inconsistencies with LD_DEBUG=Dunused and -zignore
1123 6362047 ld.so.1 dumps core when combining HWCAP and LD_PROFILE
1124 6304206 runtime linker may respect LANG and LC_MESSAGE more than LC_ALL
1125 6363495 Catchup required with Intel relocations
1126 6326497 ld.so not properly processing LD_LIBRARY_PATH ending in :
1127 6307146 mcs dumps core when appending null string to comment section
1128 6371877 LD_PROFILE_64 with gprof does not produce correct results on amd64
1129 6372082 ld -r erroneously creates .got section on i386
1130 6201866 amd64: linker symbol elimination is broken
1131 6372620 printstack() segfaults when called from static function (D)
1132 6380470 32-bit ld(1) incorrectly builds 64-bit relocatable objects
1133 6391407 Insufficient alignment of 32-bit object in archive makes ld segfault
1134 (libelf component only) (D)
1135 6316708 LD_DEBUG should provide a means of identifying/isolating individual
 link-map lists (P)
1136 6280209 elfdump cores on memory model 0x3
1137 6197234 elfdump and dump don't handle 64-bit symbols correctly
1138 6398893 Extended section processing needs some work
1139 6379726 ldd dumps core in elf_fix_name
1140 6327926 ld does not set etext symbol correctly for AMD64 medium model (D)
1141 6390410 64-bit LD_PROFILE can fail: relocation error when binding profile plt
1142 6382945 AMD64-GCC: dbx: internal error: dwarf reference attribute out of bounds
1143 6262333 init section of .so dlopened from audit interface not being called
1144 6409613 elf_outsync() should fsync()
1145 6426048 C++ exceptions broken in Nevada for amd64
1146 6429418 ld.so.1: need work-around for Nvidia drivers use of static TLS
1147 6429504 crle(1) shows wrong defaults for non-existent 64-bit config file
1148 6431835 data corruption on x64 in 64-bit mode while LD_PROFILE is in effect
1149 6423051 static TLS support within the link-editors needs a major face lift (D)
1150 6388946 attempting to dlopen a .o file mislabeled as .so fails
1151 6446740 allow mapfile symbol definitions to create backing storage (D)
1152 4986360 linker crash on exec of .so (as opposed to a.out) -- error preferred
1153 instead
1154 6229145 ld: initarray/finiarray processing occurs after got size is determined
1155 6324924 the linker should warn if there's a .init section but not _init
1156 6424132 elfdump inserts extra whitespace in bitmap value display
1157 6449485 ld(1) creates misaligned TLS in binary compiled with -xpg
1158 6424550 Write to unallocated (wua) errors when libraries are built with
 -z lazyload
1159 6464235 executing the 64-bit ld(1) should be easy (D)
1160 6465623 need a way of building unix without an interpreter
1161 6467925 ld: section deletion (-z ignore) requires improvement
1162 6357230 specfiles should be nuked (link-editor components only)
1163 -----
1164 All the above changes are incorporated in the following patches:
1165 Solaris/SunOS 5.10_sparc patch T124922-03
1166 Solaris/SunOS 5.10_x86 patch T124923-03
1167 -----
1168 These patches also include the framework changes for the following bug fixes.
1169 However, the associated feature has not been enabled in Solaris 10 or earlier
1170 releases:
1171 6174390 crle configuration files are inconsistent across platforms (D, P)
1172 6432984 ld(1) output file removal - change default behavior (D)
1173 PSARC/2006/353 ld(1) output file removal - change default behavior
1174 -----
1175 6477132 ld.so.1: memory leak when running set*id application
1176 -----
1177 6487273 ld.so.1 may open arbitrary locale files when relative path is built
 from locale environment vars
1178 6487284 ld.so.1: buffer overflow in doprf() function
1179 -----
1180 Solaris 10 508 (5th Q-update - s10u5)
1181 -----
1182 Bugid Risk Synopsis

```

1183 -----
1184 6561987 data vac_conflict faults on liphread libthread libs in s10.
1185 -----
1186 All the above changes are incorporated in the following patches:
1187 Solaris/SunOS 5.10_sparc      patch T127111-01
1188 Solaris/SunOS 5.10_x86       patch T127112-01
1189 -----
1190 6501793 GOTOP relocation transition (optimization) fails with offsets > 2^32
1191 6532924 AMD64: Solaris 5.11 55b: SEGV after whocatches
1192 6551627 OGL: SIGSEGV when trying to use OpenGL pipeline with splash screen,
1193 Solaris/Nvidia only
1194 -----
1195 All the above changes are incorporated in the following patches:
1196 Solaris/SunOS 5.10_sparc      patch T127111-04
1197 Solaris/SunOS 5.10_x86       patch T127112-04
1198 -----
1199 6479848 Enhancements to the linker support interface needed. (D)
1200 PSARC/2006/595 link-editor support library interface - ld_open()
1201 6521608 assertion failure in runtime linker related to auditing
1202 6494228 pclose() error when an audit library calls popen() and the main target
1203 is being run under ldd (D)
1204 6568745 segfault when using LD_DEBUG with bit_audit library when instrumenting
1205 mozilla (D)
1206 PSARC/2007/413 Add -zglobalaudit option to ld
1207 6602294 ps_pbrandname breaks apps linked directly against librtld_db
1208 -----
1209 All the above changes are incorporated in the following patches:
1210 Solaris/SunOS 5.10_sparc      patch T127111-07
1211 Solaris/SunOS 5.10_x86       patch T127112-07
1212 -----
1214 -----
1215 Solaris 10 908 (6th Q-update - s10u6)
1216 -----
1217 Bugid Risk Synopsis
1218 -----
1219 6672544 elf_rtbndr must support non-ABI aligned stacks on amd64
1220 6668050 First trip through PLT does not preserve args in xmm registers
1221 -----
1222 All the above changes are incorporated in the following patch:
1223 Solaris/SunOS 5.10_x86       patch T137138-01
1224 -----
1226 -----
1227 Solaris 10 409 (7th Q-update - s10u7)
1228 -----
1229 Bugid Risk Synopsis
1230 -----
1231 6629404 ld with -z ignore doesn't scale
1232 6606203 link editor ought to allow creation of >2gb sized objects (P)
1233 -----
1234 All the above changes are incorporated in the following patches:
1235 Solaris/SunOS 5.10_sparc      patch T139574-01
1236 Solaris/SunOS 5.10_x86       patch T139575-01
1237 -----
1238 6746674 setuid applications do not find libraries any more because trusted
1239 directories behavior changed (D)
1240 -----
1241 All the above changes are incorporated in the following patches:
1242 Solaris/SunOS 5.10_sparc      patch T139574-02
1243 Solaris/SunOS 5.10_x86       patch T139575-02
1244 -----
1245 6703683 Can't build VirtualBox on Build 88 or 89
1246 6737579 process_req_lib() in liblbd consumes file descriptors
1247 6685125 ld/elfdump do not handle ZERO terminator .eh_frame amd64 unwind entry
1248 -----

```

```

1249 All the above changes are incorporated in the following patches:
1250 Solaris/SunOS 5.10_sparc      patch T139574-03
1251 Solaris/SunOS 5.10_x86       patch T139575-03
1252 -----
1254 -----
1255 Solaris 10 1009 (8th Q-update - s10u8)
1256 -----
1257 Bugid Risk Synopsis
1258 -----
1259 6782597 32-bit ld.so.1 needs to accept objects with large inode number
1260 6805502 The addition of "inline" keywords to sgs code broke the lint
1261 verification in S10
1262 6807864 ld.so.1 is susceptible to a fatal dlsym()/setlocale() race
1263 -----
1264 All the above changes are incorporated in the following patches:
1265 Solaris/SunOS 5.10_sparc      patch T141692-01
1266 Solaris/SunOS 5.10_x86       patch T141693-01
1267 NOTE: The fix for 6805502 is only applicable to s10.
1268 -----
1269 6826410 ld needs to sort sections using 32-bit sort keys
1270 -----
1271 All the above changes are incorporated in the following patches:
1272 Solaris/SunOS 5.10_sparc      patch T141771-01
1273 Solaris/SunOS 5.10_x86       patch T141772-01
1274 NOTE: The fix for 6826410 is also available for s9 in the following patches:
1275 Solaris/SunOS 5.9_sparc      patch T112963-33
1276 Solaris/SunOS 5.9_x86       patch T113986-27
1277 -----
1278 6568447 bcp is broken by 6551627
1279 6599700 librtld_db needs better plugin support
1280 6713830 mdb dumped core reading a gcore
1281 6756048 rd_loadobj_iter() should always invoke brand plugin callback
1282 6786744 32-bit dbx failed with unknown rtld_db.so error on snv_104
1283 -----
1284 All the above changes are incorporated in the following patches:
1285 Solaris/SunOS 5.10_sparc      patch T141444-06
1286 Solaris/SunOS 5.10_x86       patch T141445-06
1287 -----
1289 -----
1290 Solaris 10 1005 (9th Q-update - s10u9)
1291 -----
1292 Bugid Risk Synopsis
1293 -----
1294 6850124 dlopen reports "No such file or directory" in spite of ENOMEM
1295 when mmap fails in anon_map()
1296 6826513 ldd gets confused by a crle(1) LD_PRELOAD setting
1297 6684577 ld should propagate SHF_LINK_ORDER flag to ET_REL objects
1298 6524709 executables using /usr/lib/libc.so.1 as the ELF interpreter dump core
1299 (link-editor components only)
1300 -----
1301 All the above changes are incorporated in the following patches:
1302 Solaris/SunOS 5.10_sparc      patch T143895-01
1303 Solaris/SunOS 5.10_x86       patch T143896-01
1304 -----
1306 -----
1307 Solaris 10 XXXX (10th Q-update - s10u10)
1308 -----
1309 Bugid Risk Synopsis
1310 -----
1311 6478684 isainfo/cpuid reports pause instruction not supported on amd64
1312 PSARC/2010/089 Removal of AV_386_PAUSE and AV_386_MON
1313 -----
1314 All the above changes are incorporated in the following patches:

```

```

1315      Solaris/SunOS 5.10_sparc      patch TXXXXXX-XX
1316      Solaris/SunOS 5.10_x86       patch TXXXXXX-XX
1317 -----
1319 -----
1320 Solaris Nevada (OpenSolaris 2008.05, snv_86)
1321 -----
1322 Bugid Risk Synopsis
1323 =====
1324 6409350 BrandZ project integration into Solaris (link-editor components only)
1325 6459189 UNIX03: *VSC* c99 compiler overwrites non-writable file
1326 6423746 add an option to relax the resolution of COMDAT relocations (D)
1327 4934427 runtime linker should load up static symbol names visible to
1328     dldaddr() (D)
1329     PSARC/2006/526 SHT_SUNW_LDYNNSYM - default local symbol addition
1330 6448719 sys/elf.h could be updated with additional machine and ABI types
1331 6336605 link-editors need to support R_*_SIZE relocations
1332     PSARC/2006/558 R_*_SIZE relocation support
1333 6475375 symbol search optimization to reduce rescans
1334 6475497 elfdump(1) is misreporting sh_link
1335 6482058 lari(1) could be faster, and handle per-symbol filters better
1336 6482974 defining virtual address of text segment can result in an invalid data
1337     segment
1338 6476734 crle(1m) "-l" as described fails system, crle cores trying to fix
1339     /a/var/ld/ld.config in failsafe
1340 6487499 link_audit "make clobber" creates and populates proto area
1341 6488141 ld(1) should detect attempt to reference 0-length .bss section
1342 6496718 restricted visibility symbol references should trigger archive
1343     extraction
1344 6515970 HWCAP processing doesn't clean up fmap structure - browser fails to
1345     run java applet
1346 6494214 Refinements to symbolic binding, symbol declarations and
1347     interposition (D)
1348     PSARC/2006/714 ld(1) mapfile: symbol interpose definition
1349 6475344 DTrace needs ELF function and data symbols sorted by address (D)
1350     PSARC/2007/026 ELF symbol sort sections
1351 6518480 ld -melf_i386 doesn't complain (D)
1352 6519951 bfu is just another word for exit today (RPATH -> RUNPATH conversion
1353     bites us) (D)
1354 6521504 ld: hardware capabilities processing from relocatables objects needs
1355     hardening.
1356 6518322 Some ELF utilities need updating for .SUNW_ldynsym section (D)
1357     PSARC/2007/074 -L option for nm(1) to display SHT_SUNW_LDYNNSYM symbols
1358 6523787 dlopen() handle gets mistakenly orphaned - results in access to freed
1359     memory
1360 6531189 SEGV in dladdr()
1361 6527318 dlopen(name, RTLD_NOLOAD) returns handle for unloaded library
1362 6518359 extern mapfiles references to _init/_fini can create INIT/FINI
1363     addresses of 0
1364 6533587 ld.so.1: init/fini processing needs to compensate for interposer
1365     expectations
1366 6516118 Reserved space needed in ELF dynamic section and string table (D)
1367     PSARC/2007/127 Reserved space for editing ELF dynamic sections
1368 6535688 elfdump could be more robust in the face of Purify (D)
1369 6516665 The link-editors should be more resilient against gcc's symbol
1370     versioning
1371 6541004 hwcap filter processing can leak memory
1372 5108874 elfdump SEGVs on bad object file
1373 6547441 Uninitialized variable causes ld.so.1 to crash on object cleanup
1374 6341667 elfdump should check alignments of ELF header elements
1375 6387860 elfdump cores, when processing linux built ELF file
1376 6198202 mcs -d dumps core
1377 6246083 elfdump should allow section index specification
1378     (numeric -N equivalent) (D)
1379     PSARC/2007/247 Add -I option to elfdump
1380 6556563 elfdump section overlap checking is too slow for large files

```

```

1381 5006034 need ?E mapfile feature extension (D)
1382 6565476 rtld symbol version check prevents GNU ld binary from running
1383 6567670 ld(1) symbol size/section size verification uncovers Haskell
1384     compiler inconsistency
1385 6530249 elfdump should handle ELF files with no section header table (D)
1386     PSARC/2007/395 Add -P option to elfdump
1387 6573641 ld.so.1 does not maintain parent relationship to a dlopen() caller.
1388 6577462 Additional improvements needed to handling of gcc's symbol versioning
1389 6583742 ELF string conversion library needs to lose static writable buffers
1390 6589819 ld generated reference to __tls_get_addr() fails when resolving to a
1391     shared object reference
1392 6595139 various applications should export yy* global variables for libl
1393     PSARC/2007/474 new ldd(1) -w option
1394 6597841 gelf_getdyn() reads too many dynamic entries
1395 6603313 dlclose() can fail to unload objects after fix for 6573641
1396 6234471 need a way to edit ELF objects (D)
1397     PSARC/2007/509 elfedit
1398 5035454 mixing -Kpic and -KPIC may cause SIGSEGV with -xarch=v9
1399 6473571 strip and mcs get confused and corrupt files when passed
1400     non-ELF arguments
1401 6253589 mcs has problems handling multiple SHT_NOTE sections
1402 6610591 do_reloc() should not require unused arguments
1403 6602451 new symbol visibilities required: EXPORTED, SINGLETON and ELIMINATE (D)
1404     PSARC/2007/559 new symbol visibilities - EXPORTED, SINGLETON, and
1405     ELIMINATE
1406 6570616 elfdump should display incorrectly aligned note section
1407 6614968 elfedit needs string table module (D)
1408 6620533 HWCAP filtering can leave uninitialized data behind - results in
1409     "rejected: Invalid argument"
1410 6617855 nodirect tag can be ignored when other syminfo tags are available
1411     (link-editor components only)
1412 6621066 Reduce need for new elfdump options with every section type (D)
1413     PSARC/2007/620 elfdump -T, and simplified matching
1414 6627765 soffice failure after integration of 6603313 - dangling GROUP pointer.
1415 6319025 SUNWbttool packaging issues in Nevada and S10ul.
1416 6626135 elfedit capabilities str->value mapping should come from
1417     usr/src/common/elfcap
1418 6642769 ld(1) -z combreloc should become default behavior (D)
1419     PSARC/2008/006 make ld(1) -z combreloc become default behavior
1420 6634436 XFFLAG should be updated. (link-editor components only)
1421 6492726 Merge SHF_MERGE|SHF_STRINGS input sections (D)
1422 4947191 OSNet should use direct bindings (link-editor components only)
1423 6654381 lazy loading fall-back needs optimizing
1424 6658385 ld core dumps when building Xorg on nv_82
1425 6516808 ld.so.1's token expansion provides no escape for platforms that don't
1426     report HWCAP
1427 6668534 Direct bindings can compromise function address comparisons from
1428     executables
1429 6667661 Direct bindings can compromise executables with insufficient copy
1430     relocation information
1431 6357282 ldd should recognize PARENT and EXTERN symbols (D)
1432     PSARC/2008/148 new ldd(1) -p option
1433 6672394 ldd(1) unused dependency processing is tricked by relocations errors
1434 -----
1436 -----
1437 Solaris Nevada (OpenSolaris 2008.11, snv_101)
1438 -----
1439 Bugid Risk Synopsis
1440 =====
1441 6671255 link-editor should support cross linking (D)
1442     PSARC/2008/179 cross link-editor
1443 6674666 elfedit dyn:posflag1 needs option to locate element via NEEDED item
1444 6675591 elfwrap - wrap data in an ELF file (D,P)
1445     PSARC/2008/198 elfwrap - wrap data in an ELF file
1446 6678244 elfdump dynamic section sanity checking needs refinement

```

1447 6679212 sgs use of SCCS id for versioning is obstacle to mercurial migration
 1448 6681761 lies, darn lies, and linker README files
 1449 6509323 Need to disable the Multiple Files loading - same name, different
 1450 directories (or its stat() use)
 1451 6686889 ldd(1) regression - bad pointer created with 6509323 integration
 1452 6695681 ldd(1) crashes when run from a chrooted environment
 1453 6516212 usr/src/cmd/sgs/libelf warlock targets should be fixed or abandoned
 1454 6678310 using LD_AUDIT, ldd(1) calls shared library's .init before library is
 1455 fully relocated (link-editor components only)
 1456 6699594 The ld command has a problem handling 'protected' mapfile keyword.
 1457 6699131 elfdump should display core file notes (D)
 1458 6702260 single threading .init/.fini sections breaks staroffice
 1459 6703919 boot hangs intermittently on x86 with onnv daily.0430 and on
 1460 6701798 ld can enter infinite loop processing bad mapfile
 1461 6706401 direct binding copy relocation fallback is insufficient for ild
 1462 generated objects
 1463 6705846 multithreaded C++ application seems to get deadlocked in the dynamic
 1464 linker code
 1465 6686343 ldd(1) - unused search path diagnosis should be enabled
 1466 6712292 ld.so.1 should fall back to an interposer for failed direct bindings
 1467 6716350 usr/src/cmd/sgs should be linted by nightly builds
 1468 6720509 usr/src/cmd/sgs/sgsdemangler should be removed
 1469 6617475 gas creates erroneous FILE symbols [was: ld.so.1 is reported as
 1470 false positive by wsdiff]
 1471 6724311 ldldump() mishandles R_AMD64_JUMP_SLOT relocations
 1472 6724774 elfdump -n doesn't print siginfo structure
 1473 6728555 Fix for amd64 aw (6617475) breaks pure gcc builds
 1474 6734598 ld(1) archive processing failure due to mismatched file descriptors (D)
 1475 6735939 ld(1) discarded symbol relocations errors (Studio and GNU).
 1476 6354160 Solaris linker includes more than one copy of code in binary when
 1477 linking gnu object code
 1478 6744003 ld(1) could provide better argument processing diagnostics (D)
 1479 PSARC 2008/583 add gld options to ld(1)
 1480 6749055 ld should generate GNU style VERSYM indexes for VERNEED records (D)
 1481 PSARC/2008/603 ELF objects to adopt GNU-style Versym indexes
 1482 6752728 link-editor can enter UNDEF symbols in symbol sort sections
 1483 6756472 AOUT search path pruning (D)
 1484 -----
 1486 -----
 1487 Solaris Nevada (OpenSolaris 2009.06, snv_111)
 1488 -----
 1489 Bugid Risk Synopsis
 1490 ======
 1492 6754965 introduce the SF1_SUNW_ADDR32 bit in software capabilities (D)
 1493 (link-editor components only)
 1494 PSARC/2008/622 32-bit Address Restriction Software Capabilities Flag
 1495 6756953 customer requests that DT_CONFIG strings be honored for secure apps (D)
 1496 6765299 ld --version-script option not compatible with GNU ld (D)
 1497 6748160 problem with -zrscan (D)
 1498 PSARC/2008/651 New ld archive rescan options
 1499 6763342 sloppy relocations need to get sloppier
 1500 6736890 PT_SUNWBSS should be disabled (D)
 1501 PSARC/2008/715 PT_SUNWBSS removal
 1502 6772661 ldd/lddstub/ld.so.1 dump core in current nightly while processing
 1503 libsoftcrypto_hwcap.so.1
 1504 6765931 mcs generates unlink(NULL) system calls
 1505 6775062 remove /usr/lib/libldstab.so (D)
 1506 6782977 ld segfaults after support lib version error sends bad args to vprintf()
 1507 6773695 ld -z nopartial can break non-pic objects
 1508 6778453 RTLD_GROUP prevents use of application defined malloc
 1509 6789925 64-bit applications with SF1_SUNW_ADDR32 require non-default starting
 1510 address
 1511 6792906 ld -z nopartial fix breaks TLS
 1512 6686372 ld.so.1 should use mmapobj(2)

1513 6726108 dlopen() performance could be improved.
 1514 6792836 ld is slow when processing GNU linkonce sections
 1515 6797468 ld.so.1: orphaned handles aren't processed correctly
 1516 6798676 ld.so.1: enters infinite loop with realloc/defragmentation logic
 1517 6237063 request extension to dl* family to provide segment bounds
 1518 information (D)
 1519 PSARC/2009/054 dlinfo(3c) - segment mapping retrieval
 1520 6800388 shstrtab can be sized incorrectly when -z ignore is used
 1521 6805009 ld.so.1: link map control list tear down leaves dangling pointer -
 1522 pfinstall does it again.
 1523 6807050 GNU linkonce sections can create duplicate and incompatible
 1524 eh_frame FDE entries
 1525 -----
 1527 -----
 1528 Solaris Nevada
 1529 -----
 1530 Bugid Risk Synopsis
 1531 ======
 1532 6813909 generalize eh_frame support to non-amd64 platforms
 1533 6801536 ld: mapfile processing oddities unveiled through mmapobj(2) observations
 1534 6802452 libelf shouldn't use MS_SYNC
 1535 6818012 nm tries to modify readonly segment and dumps core
 1536 6821646 xVM dom0 doesn't boot on daily.0324 and beyond
 1537 6822828 librld_db can return RD_ERR before RD_NOMAPS, which compromises dbx
 1538 expectations.
 1539 6821619 Solaris linkers need systematic approach to ELF OSABI (D)
 1540 PSARC/2009/196 ELF objects to set OSABI / elfdump -O option
 1541 6827468 6801536 breaks 'ld -s' if there are weak/strong symbol pairs
 1542 6715578 AOUT (BCP) symbol lookup can be compromised with lazy loading.
 1543 6752883 ld.so.1 error message should be buffered (not sent to stderr).
 1544 6577982 ld.so.1 calls getpid() before it should when any LD_* are set
 1545 6831285 linker LD_DEBUG support needs improvements (D)
 1546 6806791 filter builds could be optimized (link-editor components only)
 1547 6823371 calloc() uses suboptimal memset() causing 15% regression in SpecCPU2006
 1548 gcc code (link-editor components only)
 1549 6831308 ld.so.1: symbol rescanning does a little too much work
 1550 6837777 ld ordered section code uses too much memory and works too hard
 1551 6841199 Undo 10 year old workaround and use 64-bit ld on 32-bit objects
 1552 6784790 ld should examine archives to determine output object class/machine (D)
 1553 PSARC/2009/305 ld -32 option
 1554 6849998 remove undocumented mapfile \$SPECVERS and \$NEED options
 1555 6851224 elf_getshnum() and elf_getshstrndx() incompatible with 2002 ELF gABI
 1556 agreement (D)
 1557 PSARC/2009/363 replace elf_getphnum, elf_getshnum, and elf_getshstrndx
 1558 6853809 ld.so.1: rescans fallback optimization is invalid
 1559 6854158 ld.so.1: interposition can be skipped because of incorrect
 1560 caller/destination validation
 1561 6862967 rd_loadobj_iter() failing for core files
 1562 6856173 streams core dumps when compiled in 64bit with a very large static
 1563 array size
 1564 6834197 ld pukes when given an empty plate
 1565 6516644 per-symbol filtering shouldn't be allowed in executables
 1566 6878605 ld should accept '%' syntax when matching input SHT_PROGBITS sections
 1567 6850768 ld option to autogenerate wrappers/interposers similar to GNU ld
 1568 --wrap (D)
 1569 PSARC/2009/493 ld -z wrap option
 1570 6888489 Null environment variables are not overriding crle(1) replaceable
 1571 environment variables.
 1572 6885456 Need to implement GNU-ld behavior in construction of .init/.fini
 1573 sections
 1574 6900241 ld should track SHT_GROUP sections by symbol name, not section name
 1575 6901773 Special handling of STT_SECTION group signature symbol for GNU objects
 1576 6901895 Failing asserts in ld update_osym() trying to build gcc 4.5 development
 1577 head
 1578 6909523 core dump when run "LD_DEBUG=help ls" in non-English locale

1579 6903688 mdb(1) can't resolve certain symbols in solaris10-branded processes
 1580 from the global zone
 1581 6923449 elfdump misinterprets _init/_fini symbols in dynamic section test
 1582 6914728 Add dl_iterate_phdr() function to ld.so.1 (D)
 1583 PSARC/2010/015 dl_iterate_phdr
 1584 6916788 ld version 2 mapfile syntax (D)
 1585 PSARC/2009/688 Human readable and extensible ld mapfile syntax
 1586 6929607 ld generates incorrect VERDEF entries for ET_REL output objects
 1587 6924224 linker should ignore SUNW_dof when calculating the elf checksum
 1588 6918143 symbol capabilities (D)
 1589 PSARC/2010/022 Linker-editors: Symbol Capabilities
 1590 6910387 .tdata and .tbss separation invalidates TLS program header information
 1591 6934123 elfdump -d coreumps on PA-RISC elf
 1592 6931044 ld should not allow SHT_PROGBITS .eh_frame sections on amd64 (D)
 1593 6931056 pvs -r output can include empty versions in output
 1594 6938628 ld.so.1 should produce diagnostics for all dl*() entry points
 1595 6938111 nm 'No symbol table data' message goes to stdout
 1596 6941727 ld relocation cache memory use is excessive
 1597 6932220 ld -z allextract skips objects that lack global symbols
 1598 6943772 Testing for a symbols existence with RTLD_PROBE is compromised by
 1599 RTLD BIND_NOW
 1600 PSARC/2010/XXX Deferred symbol references
 1601 6943432 dlsym(RTLD_PROBE) should only bind to symbol definitions
 1602 6668759 an external method for determining whether an ELF dependency is optional
 1603 6954032 Support library with ld_open and -z allextract in svn_139 do not mix
 1604 6949596 wrong section alignment generated in joint compilation with shared
 1605 library
 1606 6961755 ld.so.1's -e arguments should take precedence over environment
 1607 variables. (D)
 1608 6748925 moe returns wrong hwcap library in some circumstances
 1609 6916796 OSnet mapfiles should use version 2 link-editor syntax
 1610 6964517 OSnet mapfiles should use version 2 link-editor syntax (2nd pass)
 1611 6948720 SHT_INIT_ARRAY etc. section names don't follow ELF gABI (D)
 1612 6962343 sgsmsg should use mkstemp() for temporary file creation
 1613 6965723 libsoftcrypto symbol capabilities rely on compiler generated
 1614 capabilities - gcc failure (link-editor components only)
 1615 6952219 ld support for archives larger than 2 GB (D, P)
 1616 PSARC/2010/224 Support for archives larger than 2 GB
 1617 6956152 disclose() from an auditor can be fatal. Preinit/activity events should
 1618 be more flexible. (D)
 1619 6971440 moe can core dump while processing libc.
 1620 6972234 sgs demo's could use some cleanup
 1621 6935867 .dynamic could be readonly in sharable objects
 1622 6975290 ld mishandles GOT relocation against local ABS symbol
 1623 6972860 ld should provide user guidance to improve objects (D)
 1624 PSARC/2010/312 Link-editor guidance
 1625 -----
 1627 -----
 1628 Illumos
 1629 -----
 1630 Bugid Risk Synopsis
 1631 ======

1633 308 ld may misalign sections only preceded by empty sections
 1634 1301 ld crashes with '-z ignore' due to a null data descriptor
 1635 1626 libld may accidentally return success while failing
 1636 2413 %ymm* need to be preserved on way through PLT
 1637 3210 ld should tolerate SHT_PROGBITS for .eh_frame sections on amd64
 1638 3228 Want -zassert-deflib for ld
 1639 3230 ld.so.1 should check default paths for DT_DEPAUDIT
 1640 3260 linker is insufficiently careful with strtok
 1641 3261 linker should ignore unknown hardware capabilities
 1642 3265 link-editor builds bogus .eh_frame_hdr on ia32
 1643 3453 GNU comdat redirection does exactly the wrong thing
 1644 3439 discarded sections shouldn't end up on output lists

1645 3436 relocatable objects also need sloppy relocation
 1646 3451 archive libraries with no symbols shouldn't require a string table
 1647 3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
 1648 3709 need sloppy relocation for GNU .debug_macro
 1649 3722 link-editor is over restrictive of R_AMD64_32 addends
 1650 3926 multiple extern map file definitions corrupt symbol table entry
 1651 3999 libld extended section handling is broken
 1652 4003 dldump() can't deal with extended sections
 1653 4227 ld --library-path is translated to -l-path, not -L
 1654 4270 ld(1) argument error reporting is still pretty bad
 1655 #endif /* ! codereview */