```
**********************************************************
   50780 Fri Jun 28 20:46:48 2013
new/usr/src/lib/libc/inc/thr_uberdata.h
3849 implement __cxa_atexit/__cxa_finalize
**********************************************************
______unchanged_portion_omitted_
 826 #endif  /* _SYSCALL32 */

 828 #define NBUCKETS        10      /* sizes ranging from 64 to 32768 */


 831 /*
 832  * atexit() data structures.
 833  * See port/gen/atexit.c for details.
 834  */
 835 typedef void (*_exithdlr_func_t) (void*);
 835 typedef void (*_exithdlr_func_t) (void);

 837 typedef struct _exthdlr {
 838         struct _exthdlr         *next;  /* next in handler list */
 839         _exithdlr_func_t        hdlr;   /* handler itself */
 840         void                    *arg;   /* argument to handler */
 841         void                    *dso;   /* DSO associated with handler */
 842 #endif /* ! codereview */
 843 } _exthdlr_t;

 845 typedef struct {
 846         mutex_t         exitfns_lock;
 847         _exthdlr_t      *head;
 848         /*
 849          * exit_frame_monitor is part of a private contract between libc and
 850          * the Sun C++ runtime.
 851          *
 852          * It should be NULL until exit() is called, and thereafter hold the
 853          * frame pointer of the function implementing our exit processing.
 854          */
 855 #endif /* ! codereview */
 856         void            *exit_frame_monitor;
 857         char            exit_pad[64 -   /* pad out to 64 bytes */
 858                 (sizeof (mutex_t) + sizeof (_exthdlr_t *) + sizeof (void *))];
 859 } atexit_root_t;

 861 #ifdef _SYSCALL32
 862 typedef struct {
 863         mutex_t         exitfns_lock;
 864         caddr32_t       head;
 865         caddr32_t       exit_frame_monitor;
 866         char            exit_pad[64 -   /* pad out to 64 bytes */
 867                 (sizeof (mutex_t) + sizeof (caddr32_t) + sizeof (caddr32_t))];
 868 } atexit_root32_t;
 869 #endif  /* _SYSCALL32 */


 872 /*
 873  * This is data that is global to all link maps (uberdata, aka super-global).
 874  * Note: When changing this, please be sure to keep the 32-bit variant of
 875  * this in sync.  (see uberdata32_t below)
 876  */
 877 typedef struct uberdata {
 878         pad_lock_t      _link_lock;
 879         pad_lock_t      _ld_lock;
 880         pad_lock_t      _fork_lock;
 881         pad_lock_t      _atfork_lock;
 882         pad32_lock_t    _callout_lock;
 883         pad32_lock_t    _tdb_hash_lock;
 884         tdb_sync_stats_t tdb_hash_lock_stats;
```



```
 885         siguaction_t    siguaction[NSIG];
 886         bucket_t        bucket[NBUCKETS];
 887         atexit_root_t   atexit_root;
 888         tsd_metadata_t  tsd_metadata;
 889         tls_metadata_t  tls_metadata;
 890         /*
 891          * Every object before this point has size and alignment of 64 bytes.
 892          * Don't add any other type of data before this point.
 893          */
 894         char    primary_map;    /* set when primary link map is initialized */
 895         char    bucket_init;    /* set when bucket[NBUCKETS] is initialized */
 896         char    pad[2];
 897         uberflags_t     uberflags;
 898         queue_head_t    *queue_head;
 899         thr_hash_table_t *thr_hash_table;
 900         uint_t          hash_size;      /* # of entries in thr_hash_table[] */
 901         uint_t          hash_mask;      /* hash_size - 1 */
 902         ulwp_t  *ulwp_one;      /* main thread */
 903         ulwp_t  *all_lwps;      /* circular ul_forw/ul_back list of live lwps */
 904         ulwp_t  *all_zombies;   /* circular ul_forw/ul_back list of zombies */
 905         int     nthreads;       /* total number of live threads/lwps */
 906         int     nzombies;       /* total number of zombie threads */
 907         int     ndaemons;       /* total number of THR_DAEMON threads/lwps */
 908         pid_t   pid;            /* the current process's pid */
 909         void    (*sigacthandler)(int, siginfo_t *, void *);
 910         ulwp_t  *lwp_stacks;
 911         ulwp_t  *lwp_laststack;
 912         int     nfreestack;
 913         int     thread_stack_cache;
 914         ulwp_t  *ulwp_freelist;
 915         ulwp_t  *ulwp_lastfree;
 916         ulwp_t  *ulwp_replace_free;
 917         ulwp_t  *ulwp_replace_last;
 918         atfork_t        *atforklist;    /* circular Q for fork handlers */
 919         robust_t        **robustlocks;  /* table of registered robust locks */
 920         robust_t        *robustlist;    /* list of registered robust locks */
 921         char    *progname;      /* the basename of the program, from argv[0] */
 922         struct uberdata **tdb_bootstrap;
 923         tdb_t   tdb;            /* thread debug interfaces (for libc_db) */
 924 } uberdata_t;

 926 #define link_lock       _link_lock.pad_lock
 927 #define ld_lock         _ld_lock.pad_lock
 928 #define fork_lock       _fork_lock.pad_lock
 929 #define atfork_lock     _atfork_lock.pad_lock
 930 #define callout_lock    _callout_lock.pad_lock
 931 #define tdb_hash_lock   _tdb_hash_lock.pad_lock

 933 #pragma align 64(__uberdata)
 934 extern  uberdata_t      __uberdata;
 935 extern  uberdata_t      **__tdb_bootstrap;      /* known to libc_db and mdb */
 936 extern  int             primary_link_map;

 938 #define ulwp_mutex(ulwp, udp)   \
 939         (&(udp)->thr_hash_table[(ulwp)->ul_ix].hash_lock)
 940 #define ulwp_condvar(ulwp, udp) \
 941         (&(udp)->thr_hash_table[(ulwp)->ul_ix].hash_cond)

 943 /*
 944  * Grab and release the hash table lock for the specified lwp.
 945  */
 946 #define ulwp_lock(ulwp, udp)    lmutex_lock(ulwp_mutex(ulwp, udp))
 947 #define ulwp_unlock(ulwp, udp)  lmutex_unlock(ulwp_mutex(ulwp, udp))

 949 #ifdef _SYSCALL32       /* needed by libc_db */
```

```
typedef struct ulwp32 {
#if defined(__sparc)
        uint32_t        ul_dinstr;      /* scratch space for dtrace */
        uint32_t        ul_padsparc0[15];
        uint32_t        ul_dsave;       /* dtrace: save %g1, %g0, %sp */
        uint32_t        ul_drestore;    /* dtrace: restore %g0, %g0, %g0 */
        uint32_t        ul_dftret;      /* dtrace: return probe fasttrap */
        uint32_t        ul_dreturn;     /* dtrace: return %o0 */
#endif
        caddr32_t       ul_self;        /* pointer to self */
#if defined(__x86)
        uint8_t         ul_dinstr[40];  /* scratch space for dtrace */
#endif
        caddr32_t       ul_uberdata;    /* uber (super-global) data */
        tls32_t         ul_tls;         /* dynamic thread-local storage base */
        caddr32_t       ul_forw;        /* forw, back all_lwps list, */
        caddr32_t       ul_back;        /* protected by link_lock */
        caddr32_t       ul_next;        /* list to keep track of stacks */
        caddr32_t       ul_hash;        /* hash chain linked list */
        caddr32_t       ul_rval;        /* return value from thr_exit() */
        caddr32_t       ul_stk;         /* mapping base of the stack */
        size32_t        ul_mapsiz;      /* mapping size of the stack */
        size32_t        ul_guardsize;   /* normally _lpagesize */
        caddr32_t       ul_stktop;      /* broken thr_stksegment() interface */
        size32_t        ul_stksiz;      /* broken thr_stksegment() interface */
        stack32_t       ul_ustack;      /* current stack boundaries */
        int             ul_ix;          /* hash index */
        lwpid_t         ul_lwpid;       /* thread id, aka the lwp id */
        pri_t           ul_pri;         /* scheduling priority */
        pri_t           ul_epri;        /* real-time ceiling priority */
        char            ul_policy;      /* scheduling policy */
        char            ul_cid;         /* scheduling class id */
        union {
                struct {
                        char    cursig; /* deferred signal number */
                        char    pleasestop; /* lwp requested to stop itself */
                } s;
                short   curplease;      /* for testing both at once */
        } ul_cp;
        char            ul_stop;        /* reason for stopping */
        char            ul_signalled;   /* this lwp was cond_signal()d */
        char            ul_dead;        /* this lwp has called thr_exit */
        char            ul_unwind;      /* posix: unwind C++ stack */
        char            ul_detached;    /* THR_DETACHED at thread_create() */
                                        /* or pthread_detach() was called */
        char            ul_writer;      /* sleeping in rw_wrlock() */
        char            ul_stopping;    /* set by curthread: stopping self */
        char            ul_cancel_prologue;     /* for _cancel_prologue() */
        short           ul_preempt;     /* no_preempt()/preempt() */
        short           ul_savpreempt;  /* pre-existing preempt value */
        char            ul_sigsuspend;  /* thread is in sigsuspend/pollsys */
        char            ul_main;        /* thread is the main thread */
        char            ul_fork;        /* thread is performing a fork */
        char            ul_primarymap;  /* primary link-map is initialized */
        /* per-thread copies of the corresponding global variables */
        uint8_t         ul_max_spinners;        /* thread_max_spinners */
        char            ul_door_noreserve;      /* thread_door_noreserve */
        char            ul_queue_fifo;          /* thread_queue_fifo */
        char            ul_cond_wait_defer;     /* thread_cond_wait_defer */
        char            ul_error_detection;     /* thread_error_detection */
        char            ul_async_safe;          /* thread_async_safe */
        char            ul_rt;                  /* found on an RT queue */
        char            ul_rtqueued;            /* was RT when queued */
        char            ul_misaligned;          /* thread_locks_misaligned */
        char            ul_pad[3];
        int             ul_adaptive_spin;       /* thread_adaptive_spin */
```



```
        int             ul_queue_spin;          /* thread_queue_spin */
        int             ul_critical;    /* non-zero == in a critical region */
        int             ul_sigdefer;    /* non-zero == defer signals */
        int             ul_vfork;       /* thread is the child of vfork() */
        int             ul_cancelable;  /* _cancelon()/_canceloff() */
        char            ul_cancel_pending;  /* pthread_cancel() was called */
        char            ul_cancel_disabled; /* PTHREAD_CANCEL_DISABLE */
        char            ul_cancel_async;    /* PTHREAD_CANCEL_ASYNCHRONOUS */
        char            ul_save_async;  /* saved copy of ul_cancel_async */
        char            ul_mutator;     /* lwp is a mutator (java interface) */
        char            ul_created;     /* created suspended */
        char            ul_replace;     /* replacement; must be free()d */
        uchar_t         ul_nocancel;    /* cancellation can't happen */
        int             ul_errno;       /* per-thread errno */
        caddr32_t       ul_errnop;      /* pointer to errno or self->ul_errno */
        caddr32_t       ul_clnup_hdr;   /* head of cleanup handlers list */
        caddr32_t       ul_schedctl_called; /* ul_schedctl is set up */
        caddr32_t       ul_schedctl;    /* schedctl data */
        int             ul_bindflags;   /* bind_guard() interface to ld.so.1 */
        uint_t          ul_libc_locks;  /* count of cancel_safe_mutex_lock()s */
        caddr32_t       ul_stsd;        /* slow TLS for keys >= TSD_NFAST */
        caddr32_t       ul_ftsd[TSD_NFAST]; /* fast TLS for keys < TSD_NFAST */
        td_evbuf32_t    ul_td_evbuf;    /* event buffer */
        char            ul_td_events_enable;    /* event mechanism enabled */
        char            ul_sync_obj_reg;        /* tdb_sync_obj_register() */
        char            ul_qtype;       /* MX or CV */
        char            ul_cv_wake;     /* != 0: just wake up, don't requeue */
        int             ul_rtld;        /* thread is running inside ld.so.1 */
        int             ul_usropts;     /* flags given to thr_create() */
        caddr32_t       ul_startpc;     /* start func (thr_create()) */
        caddr32_t       ul_startarg;    /* argument for start function */
        caddr32_t       ul_wchan;       /* synch object when sleeping */
        caddr32_t       ul_link;        /* sleep queue link */
        caddr32_t       ul_sleepq;      /* sleep queue thread is waiting on */
        caddr32_t       ul_cvmutex;     /* mutex dropped when waiting on a cv */
        caddr32_t       ul_mxchain;     /* chain of owned ceiling mutexes */
        int             ul_save_state;  /* bind_guard() interface to ld.so.1 */
        uint_t          ul_rdlockcnt;   /* # entries in ul_readlock array */
                                /* 0 means there is but a single entry */
        union {                         /* single entry or pointer to array */
                readlock32_t    single;
                caddr32_t       array;
        } ul_readlock;
        uint_t          ul_heldlockcnt; /* # entries in ul_heldlocks array */
                                /* 0 means there is but a single entry */
        union {                         /* single entry or pointer to array */
                caddr32_t       single;
                caddr32_t       array;
        } ul_heldlocks;
        /* PROBE_SUPPORT begin */
        caddr32_t       ul_tpdp;
        /* PROBE_SUPPORT end */
        caddr32_t       ul_siglink;     /* pointer to previous context */
        uint_t          ul_spin_lock_spin;      /* spin lock statistics */
        uint_t          ul_spin_lock_spin2;
        uint_t          ul_spin_lock_sleep;
        uint_t          ul_spin_lock_wakeup;
        queue_root32_t  ul_queue_root;  /* root of a sleep queue */
        id_t            ul_rtclassid;   /* real-time class id */
        uint_t          ul_pilocks;     /* count of PI locks held */
                /* the following members *must* be last in the structure */
                /* they are discarded when ulwp is replaced on thr_exit() */
        sigset_t        ul_sigmask;     /* thread's current signal mask */
        sigset_t        ul_tmpmask;     /* signal mask for sigsuspend/pollsys */
        siginfo32_t     ul_siginfo;     /* deferred siginfo */
        mutex_t         ul_spinlock;    /* used when suspending/continuing */
```

```
1083         fpuenv32_t      ul_fpuenv;      /* floating point state */
1084         caddr32_t       ul_sp;          /* stack pointer when blocked */
1085 #if defined(sparc)
1086         caddr32_t       ul_unwind_ret;  /* used only by _ex_clnup_handler() */
1087 #endif
1088 } ulwp32_t;

1090 #define REPLACEMENT_SIZE32      ((size_t)&((ulwp32_t *)NULL)->ul_sigmask)

1092 typedef struct uberdata32 {
1093         pad_lock_t      _link_lock;
1094         pad_lock_t      _ld_lock;
1095         pad_lock_t      _fork_lock;
1096         pad_lock_t      _atfork_lock;
1097         pad32_lock_t    _callout_lock;
1098         pad32_lock_t    _tdb_hash_lock;
1099         tdb_sync_stats_t tdb_hash_lock_stats;
1100         siguaction32_t  siguaction[NSIG];
1101         bucket32_t      bucket[NBUCKETS];
1102         atexit_root32_t atexit_root;
1103         tsd_metadata32_t tsd_metadata;
1104         tls_metadata32_t tls_metadata;
1105         char            primary_map;
1106         char            bucket_init;
1107         char            pad[2];
1108         uberflags_t     uberflags;
1109         caddr32_t       queue_head;
1110         caddr32_t       thr_hash_table;
1111         uint_t          hash_size;
1112         uint_t          hash_mask;
1113         caddr32_t       ulwp_one;
1114         caddr32_t       all_lwps;
1115         caddr32_t       all_zombies;
1116         int             nthreads;
1117         int             nzombies;
1118         int             ndaemons;
1119         int             pid;
1120         caddr32_t       sigacthandler;
1121         caddr32_t       lwp_stacks;
1122         caddr32_t       lwp_laststack;
1123         int             nfreestack;
1124         int             thread_stack_cache;
1125         caddr32_t       ulwp_freelist;
1126         caddr32_t       ulwp_lastfree;
1127         caddr32_t       ulwp_replace_free;
1128         caddr32_t       ulwp_replace_last;
1129         caddr32_t       atforklist;
1130         caddr32_t       robustlocks;
1131         caddr32_t       robustlist;
1132         caddr32_t       progname;
1133         caddr32_t       tdb_bootstrap;
1134         tdb32_t         tdb;
1135 } uberdata32_t;

1137 #endif  /* _SYSCALL32 */

1139 /* ul_stop values */
1140 #define TSTP_REGULAR    0x01    /* Stopped by thr_suspend() */
1141 #define TSTP_MUTATOR    0x08    /* stopped by thr_suspend_*mutator*() */
1142 #define TSTP_FORK       0x20    /* stopped by suspend_fork() */

1144 /*
1145  * Implementation-specific attribute types for pthread_mutexattr_init() etc.
1146  */

1148 typedef struct  _cvattr {
```



```
1149         int     pshared;
1150         clockid_t clockid;
1151 } cvattr_t;

1153 typedef struct  _mattr {
1154         int     pshared;
1155         int     protocol;
1156         int     prioceiling;
1157         int     type;
1158         int     robustness;
1159 } mattr_t;

1161 typedef struct  _thrattr {
1162         size_t  stksize;
1163         void    *stkaddr;
1164         int     detachstate;
1165         int     daemonstate;
1166         int     scope;
1167         int     prio;
1168         int     policy;
1169         int     inherit;
1170         size_t  guardsize;
1171 } thrattr_t;

1173 typedef struct  _rwlattr {
1174         int     pshared;
1175 } rwlattr_t;

1177 /* _curthread() is inline for speed */
1178 extern  ulwp_t          *_curthread(void);
1179 #define curthread       (_curthread())

1181 /* this version (also inline) can be tested for NULL */
1182 extern  ulwp_t          *__curthread(void);

1184 /* get the current stack pointer (also inline) */
1185 extern  greg_t          stkptr(void);

1187 /*
1188  * Suppress __attribute__((...)) if we are not compiling with gcc
1189  */
1190 #if !defined(__GNUC__)
1191 #define __attribute__(string)
1192 #endif

1194 /* Fetch the dispatch (kernel) priority of a thread */
1195 #define real_priority(ulwp)     \
1196         ((ulwp)->ul_schedctl? (ulwp)->ul_schedctl->sc_priority : 0)

1198 /*
1199  * Implementation functions.  Not visible outside of the library itself.
1200  */
1201 extern  int     __nanosleep(const timespec_t *, timespec_t *);
1202 extern  void    getgregs(ulwp_t *, gregset_t);
1203 extern  void    setgregs(ulwp_t *, gregset_t);
1204 extern  void    thr_panic(const char *);
1205 #pragma rarely_called(thr_panic)
1206 extern  ulwp_t  *find_lwp(thread_t);
1207 extern  void    finish_init(void);
1208 extern  void    update_sched(ulwp_t *);
1209 extern  void    queue_alloc(void);
1210 extern  void    tsd_exit(void);
1211 extern  void    tsd_free(ulwp_t *);
1212 extern  void    tls_setup(void);
1213 extern  void    tls_exit(void);
1214 extern  void    tls_free(ulwp_t *);
```

```
1215 extern  void    rwl_free(ulwp_t *);
1216 extern  void    heldlock_exit(void);
1217 extern  void    heldlock_free(ulwp_t *);
1218 extern  void    sigacthandler(int, siginfo_t *, void *);
1219 extern  void    signal_init(void);
1220 extern  int     sigequalset(const sigset_t *, const sigset_t *);
1221 extern  void    mutex_setup(void);
1222 extern  void    take_deferred_signal(int);
1223 extern  void    *setup_top_frame(void *, size_t, ulwp_t *);
1224 extern  int     setup_context(ucontext_t *, void *(*func)(ulwp_t *),
1225                         ulwp_t *ulwp, caddr_t stk, size_t stksize);
1226 extern  volatile sc_shared_t *setup_schedctl(void);
1227 extern  void    *lmalloc(size_t);
1228 extern  void    lfree(void *, size_t);
1229 extern  void    *libc_malloc(size_t);
1230 extern  void    *libc_realloc(void *, size_t);
1231 extern  void    libc_free(void *);
1232 extern  char    *libc_strdup(const char *);
1233 extern  void    ultos(uint64_t, int, char *);
1234 extern  void    lock_error(const mutex_t *, const char *, void *, const char *);
1235 extern  void    rwlock_error(const rwlock_t *, const char *, const char *);
1236 extern  void    thread_error(const char *);
1237 extern  void    grab_assert_lock(void);
1238 extern  void    dump_queue_statistics(void);
1239 extern  void    collect_queue_statistics(void);
1240 extern  void    record_spin_locks(ulwp_t *);
1241 extern  void    remember_lock(mutex_t *);
1242 extern  void    forget_lock(mutex_t *);
1243 extern  void    register_lock(mutex_t *);
1244 extern  void    unregister_locks(void);
1245 #if defined(__sparc)
1246 extern  void    _flush_windows(void);
1247 #else
1248 #define _flush_windows()
1249 #endif
1250 extern  void    set_curthread(void *);

1252 /*
1253  * Utility function used when waking up many threads (more than MAXLWPS)
1254  * all at once.  See mutex_wakeup_all(), cond_broadcast(), and rw_unlock().
1255  */
1256 #define MAXLWPS 128     /* max remembered lwpids before overflow */
1257 #define NEWLWPS 2048    /* max remembered lwpids at first overflow */
1258 extern  lwpid_t *alloc_lwpids(lwpid_t *, int *, int *);

1260 /* enter a critical section */
1261 #define enter_critical(self)    (self->ul_critical++)

1263 /* exit a critical section, take deferred actions if necessary */
1264 extern  void    do_exit_critical(void);
1265 #define exit_critical(self)                                     \
1266         (void) (self->ul_critical--,                           \
1267             ((self->ul_curplease && self->ul_critical == 0)?   \
1268             (do_exit_critical(), 0) : 0))

1270 /*
1271  * Like enter_critical()/exit_critical() but just for deferring signals.
1272  * Unlike enter_critical()/exit_critical(), ul_sigdefer may be set while
1273  * calling application functions like constructors and destructors.
1274  * Care must be taken if the application function attempts to set
1275  * the signal mask while a deferred signal is present; the setting
1276  * of the signal mask must also be deferred.
1277  */
1278 #define sigoff(self)    (self->ul_sigdefer++)
1279 #define sigon(self)                                             \
1280         (void) ((--self->ul_sigdefer == 0 &&                   \
```



```
1281             self->ul_curplease && self->ul_critical == 0)?     \
1282             (do_exit_critical(), 0) : 0)

1284 /* these are exported functions */
1285 extern  void    _sigoff(void);
1286 extern  void    _sigon(void);

1288 #define sigorset(s1, s2)                                        \
1289         (((s1)->__sigbits[0] |= (s2)->__sigbits[0]),    \
1290         ((s1)->__sigbits[1] |= (s2)->__sigbits[1]),     \
1291         ((s1)->__sigbits[2] |= (s2)->__sigbits[2]),     \
1292         ((s1)->__sigbits[3] |= (s2)->__sigbits[3]))

1294 #define sigandset(s1, s2)                                       \
1295         (((s1)->__sigbits[0] &= (s2)->__sigbits[0]),    \
1296         ((s1)->__sigbits[1] &= (s2)->__sigbits[1]),     \
1297         ((s1)->__sigbits[2] &= (s2)->__sigbits[2]),     \
1298         ((s1)->__sigbits[3] &= (s2)->__sigbits[3]))

1300 #define sigdiffset(s1, s2)                                      \
1301         (((s1)->__sigbits[0] &= ~(s2)->__sigbits[0]),   \
1302         ((s1)->__sigbits[1] &= ~(s2)->__sigbits[1]),    \
1303         ((s1)->__sigbits[2] &= ~(s2)->__sigbits[2]),    \
1304         ((s1)->__sigbits[3] &= ~(s2)->__sigbits[3]))

1306 #define delete_reserved_signals(s)                              \
1307         (((s)->__sigbits[0] &= MASKSET0),               \
1308         ((s)->__sigbits[1] &= (MASKSET1 & ~SIGMASK(SIGCANCEL))),\
1309         ((s)->__sigbits[2] &= MASKSET2),                \
1310         ((s)->__sigbits[3] &= MASKSET3))

1312 extern  void    block_all_signals(ulwp_t *self);

1314 /*
1315  * When restoring the signal mask after having previously called
1316  * block_all_signals(), if we have a deferred signal present then
1317  * do nothing other than ASSERT() that we are in a critical region.
1318  * The signal mask will be set when we emerge from the critical region
1319  * and call take_deferred_signal().  There is no race condition here
1320  * because the kernel currently has all signals blocked for this thread.
1321  */
1322 #define restore_signals(self)                                           \
1323         ((void) ((self)->ul_cursig?                                    \
1324         (ASSERT((self)->ul_critical + (self)->ul_sigdefer != 0), 0) :   \
1325         __lwp_sigmask(SIG_SETMASK, &(self)->ul_sigmask)))

1327 extern  void    set_cancel_pending_flag(ulwp_t *, int);
1328 extern  void    set_cancel_eintr_flag(ulwp_t *);
1329 extern  void    set_parking_flag(ulwp_t *, int);
1330 extern  int     cancel_active(void);

1332 extern  void    *_thrp_setup(ulwp_t *);
1333 extern  void    _fpinherit(ulwp_t *);
1334 extern  void    _lwp_start(void);
1335 extern  void    _lwp_terminate(void);
1336 extern  void    lmutex_lock(mutex_t *);
1337 extern  void    lmutex_unlock(mutex_t *);
1338 extern  void    lrw_rdlock(rwlock_t *);
1339 extern  void    lrw_wrlock(rwlock_t *);
1340 extern  void    lrw_unlock(rwlock_t *);
1341 extern  void    sig_mutex_lock(mutex_t *);
1342 extern  void    sig_mutex_unlock(mutex_t *);
1343 extern  int     sig_mutex_trylock(mutex_t *);
1344 extern  int     sig_cond_wait(cond_t *, mutex_t *);
1345 extern  int     sig_cond_reltimedwait(cond_t *, mutex_t *, const timespec_t *);
1346 extern  void    cancel_safe_mutex_lock(mutex_t *);
```

```
extern  void    cancel_safe_mutex_unlock(mutex_t *);
extern  int     cancel_safe_mutex_trylock(mutex_t *);
extern  void    _prefork_handler(void);
extern  void    _postfork_parent_handler(void);
extern  void    _postfork_child_handler(void);
extern  void    postfork1_child(void);
extern  void    postfork1_child_aio(void);
extern  void    postfork1_child_sigev_aio(void);
extern  void    postfork1_child_sigev_mq(void);
extern  void    postfork1_child_sigev_timer(void);
extern  void    postfork1_child_tpool(void);
extern  void    fork_lock_enter(void);
extern  void    fork_lock_exit(void);
extern  void    suspend_fork(void);
extern  void    continue_fork(int);
extern  void    do_sigcancel(void);
extern  void    setup_cancelsig(int);
extern  void    init_sigev_thread(void);
extern  void    init_aio(void);
extern  void    init_progname(void);
extern  void    _cancelon(void);
extern  void    _canceloff(void);
extern  void    _canceloff_nocancel(void);
extern  void    _cancel_prologue(void);
extern  void    _cancel_epilogue(void);
extern  void    no_preempt(ulwp_t *);
extern  void    preempt(ulwp_t *);
extern  void    _thrp_unwind(void *);

extern  pid_t   __forkx(int);
extern  pid_t   __forkallx(int);
extern  int     __open(const char *, int, mode_t);
extern  int     __open64(const char *, int, mode_t);
extern  int     __openat(int, const char *, int, mode_t);
extern  int     __openat64(int, const char *, int, mode_t);
extern  int     __close(int);
extern  ssize_t __read(int, void *, size_t);
extern  ssize_t __write(int, const void *, size_t);
extern  int     __fcntl(int, int, ...);
extern  int     __lwp_continue(lwpid_t);
extern  int     __lwp_create(ucontext_t *, uint_t, lwpid_t *);
extern  int     ___lwp_suspend(lwpid_t);
extern  int     lwp_wait(lwpid_t, lwpid_t *);
extern  int     __lwp_wait(lwpid_t, lwpid_t *);
extern  int     __lwp_detach(lwpid_t);
extern  sc_shared_t *__schedctl(void);

/* actual system call traps */
extern  int     __setcontext(const ucontext_t *);
extern  int     __getcontext(ucontext_t *);
extern  int     __clock_gettime(clockid_t, timespec_t *);
extern  void    abstime_to_reltime(clockid_t, const timespec_t *, timespec_t *);
extern  void    hrt2ts(hrtime_t, timespec_t *);

extern  int     __sigaction(int, const struct sigaction *, struct sigaction *);
extern  int     __sigprocmask(int, const sigset_t *, sigset_t *);
extern  int     __lwp_sigmask(int, const sigset_t *);
extern  void    __sighndlr(int, siginfo_t *, ucontext_t *, void (*)());
extern  caddr_t __sighndlrend;
#pragma unknown_control_flow(__sighndlr)

/* belongs in <pthread.h> */
#define PTHREAD_CREATE_DAEMON_NP        0x100   /* = THR_DAEMON */
#define PTHREAD_CREATE_NONDAEMON_NP     0
extern  int     pthread_attr_setdaemonstate_np(pthread_attr_t *, int);
extern  int     pthread_attr_getdaemonstate_np(const pthread_attr_t *, int *);
```



```
extern  int     mutex_held(mutex_t *);
extern  int     mutex_lock_internal(mutex_t *, timespec_t *, int);
extern  int     mutex_unlock_internal(mutex_t *, int);

/* not cancellation points: */
extern  int     __cond_wait(cond_t *, mutex_t *);
extern  int     __cond_timedwait(cond_t *, mutex_t *, const timespec_t *);
extern  int     __cond_reltimedwait(cond_t *, mutex_t *, const timespec_t *);

extern  int     rw_read_held(rwlock_t *);
extern  int     rw_write_held(rwlock_t *);

extern  int     _thrp_create(void *, size_t, void *(*)(void *), void *, long,
                        thread_t *, size_t);
extern  int     _thrp_suspend(thread_t, uchar_t);
extern  int     _thrp_continue(thread_t, uchar_t);

extern  void    _thrp_terminate(void *);
extern  void    _thrp_exit(void);

extern  const pcclass_t *get_info_by_class(id_t);
extern  const pcclass_t *get_info_by_policy(int);
extern  const thrattr_t *def_thrattr(void);
extern  id_t    setparam(idtype_t, id_t, int, int);
extern  id_t    setprio(idtype_t, id_t, int, int *);
extern  id_t    getparam(idtype_t, id_t, int *, struct sched_param *);

/*
 * System call wrappers (direct interfaces to the kernel)
 */
extern  int     ___lwp_mutex_register(mutex_t *, mutex_t **);
extern  int     ___lwp_mutex_trylock(mutex_t *, ulwp_t *);
extern  int     ___lwp_mutex_timedlock(mutex_t *, timespec_t *, ulwp_t *);
extern  int     ___lwp_mutex_unlock(mutex_t *);
extern  int     ___lwp_mutex_wakeup(mutex_t *, int);
extern  int     ___lwp_cond_wait(cond_t *, mutex_t *, timespec_t *, int);
extern  int     ___lwp_sema_timedwait(lwp_sema_t *, timespec_t *, int);
extern  int     __lwp_rwlock_rdlock(rwlock_t *, timespec_t *);
extern  int     __lwp_rwlock_wrlock(rwlock_t *, timespec_t *);
extern  int     __lwp_rwlock_tryrdlock(rwlock_t *);
extern  int     __lwp_rwlock_trywrlock(rwlock_t *);
extern  int     __lwp_rwlock_unlock(rwlock_t *);
extern  int     __lwp_park(timespec_t *, lwpid_t);
extern  int     __lwp_unpark(lwpid_t);
extern  int     __lwp_unpark_all(lwpid_t *, int);
#if defined(__x86)
extern  int     ___lwp_private(int, int, void *);
#endif  /* __x86 */

/*
 * inlines
 */
extern  int             set_lock_byte(volatile uint8_t *);
extern  uint32_t        atomic_swap_32(volatile uint32_t *, uint32_t);
extern  uint32_t        atomic_cas_32(volatile uint32_t *, uint32_t, uint32_t);
extern  void            atomic_inc_32(volatile uint32_t *);
extern  void            atomic_dec_32(volatile uint32_t *);
extern  void            atomic_and_32(volatile uint32_t *, uint32_t);
extern  void            atomic_or_32(volatile uint32_t *, uint32_t);
#if defined(__sparc)
extern  ulong_t         caller(void);
extern  ulong_t         getfp(void);
#endif  /* __sparc */

#include "thr_inlines.h"
```

```
1480 #endif  /* _THR_UBERDATA_H */
```

```
**********************************************************
   10870 Fri Jun 28 20:46:50 2013
new/usr/src/lib/libc/port/gen/atexit.c
3849 implement __cxa_atexit/__cxa_finalize
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1988 AT&T */
/*        All Rights Reserved   */

#pragma weak _atexit = atexit

#include "lint.h"
#include "thr_uberdata.h"
#include "libc_int.h"
#include "atexit.h"
#include "stdiom.h"

/*
 * Note that memory is managed by lmalloc()/lfree().
 *
 * Among other reasons, this is occasioned by the insistence of our
 * brothers sh(1) and csh(1) that they can do malloc, etc., better than
 * libc can.  Those programs define their own malloc routines, and
 * initialize the underlying mechanism in main().  This means that calls
 * to malloc occuring before main will crash.  The loader calls atexit(3C)
 * before calling main, so we'd better avoid malloc() when it does.
 *
 * Another reason for using lmalloc()/lfree() is that the atexit()
 * list must transcend all link maps.  See the Linker and Libraries
 * Guide for information on alternate link maps.
 *
 * See "thr_uberdata.h" for the definitions of structures used here.
 */

static int in_range(void *, Lc_addr_range_t[], uint_t count);
static int in_range(_exithdlr_func_t, Lc_addr_range_t[], uint_t count);

extern  caddr_t _getfp(void);

/*
 * exitfns_lock is declared to be a recursive mutex so that we
```



```
 * can hold it while calling out to the registered functions.
 * If they call back to us, we are self-consistent and everything
 * works, even the case of calling exit() from functions called
 * by _exithandle() (recursive exit()).  All that is required is
 * that the registered functions actually return (no longjmp()s).
 *
 * Because exitfns_lock is declared to be a recursive mutex, we
 * cannot use it with lmutex_lock()/lmutex_unlock() and we must
 * use mutex_lock()/mutex_unlock().  This means that atexit()
 * and exit() are not async-signal-safe.  We make them fork1-safe
 * via the atexit_locks()/atexit_unlocks() functions, called from
 * libc_prepare_atfork()/libc_child_atfork()/libc_parent_atfork()
 */

/*
 * atexit_locks() and atexit_unlocks() are called on every link map.
 * Do not use curthread->ul_uberdata->atexit_root for these.
 */
void
atexit_locks()
{
        (void) mutex_lock(&__uberdata.atexit_root.exitfns_lock);
}
```

_____*unchanged_portion_omitted_*

```
#endif /* ! codereview */
/*
 * atexit() is called before the primordial thread is fully set up.
 * Be careful about dereferencing self->ul_uberdata->atexit_root.
 */
int
__cxa_atexit(void (*hdlr)(void *), void *arg, void *dso)
atexit(void (*func)(void))
{
        ulwp_t *self;
        atexit_root_t *arp;
        _exthdlr_t *p;

        if ((p = lmalloc(sizeof (_exthdlr_t))) == NULL)
                return (-1);

        if ((self = __curthread()) == NULL)
                arp = &__uberdata.atexit_root;
        else {
                arp = &self->ul_uberdata->atexit_root;
                (void) mutex_lock(&arp->exitfns_lock);
        }
        p->hdlr = hdlr;
        p->arg = arg;
        p->dso = dso;
        p->hdlr = func;
        p->next = arp->head;
        arp->head = p;

#endif /* ! codereview */
        if (self != NULL)
                (void) mutex_unlock(&arp->exitfns_lock);
        return (0);
}

int
atexit(void (*func)(void))
{
        return (__cxa_atexit((_exithdlr_func_t)func, NULL, NULL));
}
```

```
/*
 * Note that we may be entered recursively, as we'll call __cxa_finalize(0) at
 * exit, one of our handlers is ld.so.1'atexit_fini, and libraries may call
 * __cxa_finalize(__dso_handle) from their _fini.
 */
#endif /* ! codereview */
void
__cxa_finalize(void *dso)
_exithandle(void)
{
        atexit_root_t *arp = &curthread->ul_uberdata->atexit_root;
        _exthdlr_t *p, *o;
        _exthdlr_t *p;
        int cancel_state;

        /* disable cancellation while running atexit handlers */
        (void) pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, &cancel_state);
        (void) mutex_lock(&arp->exitfns_lock);

        o = NULL;
        arp->exit_frame_monitor = _getfp() + STACK_BIAS;
        p = arp->head;
        while (p != NULL) {
                if ((dso == NULL) || (p->dso == dso)) {
                        if (o != NULL)
                                o->next = p->next;
                        else
#endif /* ! codereview */
                                arp->head = p->next;

                        p->hdlr(p->arg);
                p->hdlr();
                        lfree(p, sizeof (_exthdlr_t));
                        o = NULL;
#endif /* ! codereview */
                        p = arp->head;
                } else {
                        o = p;
                        p = p->next;
#endif /* ! codereview */
                }
        }

#endif /* ! codereview */
        (void) mutex_unlock(&arp->exitfns_lock);
        (void) pthread_setcancelstate(cancel_state, NULL);
}

void
_exithandle(void)
{
        atexit_root_t *arp = &curthread->ul_uberdata->atexit_root;

        arp->exit_frame_monitor = _getfp() + STACK_BIAS;
        __cxa_finalize(NULL);
}

#endif /* ! codereview */
/*
 * _get_exit_frame_monitor is called by the C++ runtimes.
 */
void *
_get_exit_frame_monitor(void)
{
        atexit_root_t *arp = &curthread->ul_uberdata->atexit_root;
```



```
        return (&arp->exit_frame_monitor);
}

/*
 * The following is a routine which the loader (ld.so.1) calls when it
 * processes a dlclose call on an object.  It resets all signal handlers
 * which fall within the union of the ranges specified by the elements
 * of the array range to SIG_DFL.
 */
static void
_preexec_sig_unload(Lc_addr_range_t range[], uint_t count)
{
        uberdata_t *udp = curthread->ul_uberdata;
        int sig;
        rwlock_t *rwlp;
        struct sigaction *sap;
        struct sigaction oact;
        void (*handler)();

        for (sig = 1; sig < NSIG; sig++) {
                sap = (struct sigaction *)&udp->siguaction[sig].sig_uaction;
again:
                handler = sap->sa_handler;
                if (handler != SIG_DFL && handler != SIG_IGN &&
                    in_range((void *)handler, range, count)) {
                    in_range(handler, range, count)) {
                        rwlp = &udp->siguaction[sig].sig_lock;
                        lrw_wrlock(rwlp);
                        if (handler != sap->sa_handler) {
                                lrw_unlock(rwlp);
                                goto again;
                        }
                        sap->sa_handler = SIG_DFL;
                        sap->sa_flags = SA_SIGINFO;
                        (void) sigemptyset(&sap->sa_mask);
                        if (__sigaction(sig, NULL, &oact) == 0 &&
                            oact.sa_handler != SIG_DFL &&
                            oact.sa_handler != SIG_IGN)
                                (void) __sigaction(sig, sap, NULL);
                        lrw_unlock(rwlp);
                }
        }
}

/*
 * The following is a routine which the loader (ld.so.1) calls when it
 * processes a dlclose call on an object.  It cancels all atfork() entries
 * whose prefork, parent postfork, or child postfork functions fall within
 * the union of the ranges specified by the elements of the array range.
 */
static void
_preexec_atfork_unload(Lc_addr_range_t range[], uint_t count)
{
        ulwp_t *self = curthread;
        uberdata_t *udp = self->ul_uberdata;
        atfork_t *atfork_q;
        atfork_t *atfp;
        atfork_t *next;
        void (*func)(void);
        int start_again;

        (void) mutex_lock(&udp->atfork_lock);
        if ((atfork_q = udp->atforklist) != NULL) {
                atfp = atfork_q;
                do {
                        next = atfp->forw;
```

```
257                         start_again = 0;

259                         if (((func = atfp->prepare) != NULL &&
260                             in_range((void *)func, range, count)) ||
167                             in_range(func, range, count)) ||
261                             ((func = atfp->parent) != NULL &&
262                             in_range((void *)func, range, count)) ||
169                             in_range(func, range, count)) ||
263                             ((func = atfp->child) != NULL &&
264                             in_range((void *)func, range, count))) {
171                             in_range(func, range, count))) {
265                                 if (self->ul_fork) {
266                                         /*
267                                          * dlclose() called from a fork handler.
268                                          * Deleting the entry would wreak havoc.
269                                          * Just null out the function pointers
270                                          * and leave the entry in place.
271                                          */
272                                         atfp->prepare = NULL;
273                                         atfp->parent = NULL;
274                                         atfp->child = NULL;
275                                         continue;
276                                 }
277                                 if (atfp == atfork_q) {
278                                         /* deleting the list head member */
279                                         udp->atforklist = atfork_q = next;
280                                         start_again = 1;
281                                 }
282                                 atfp->forw->back = atfp->back;
283                                 atfp->back->forw = atfp->forw;
284                                 lfree(atfp, sizeof (atfork_t));
285                                 if (atfp == atfork_q) {
286                                         /* we deleted the whole list */
287                                         udp->atforklist = NULL;
288                                         break;
289                                 }
290                         }
291                 } while ((atfp = next) != atfork_q || start_again);
292         }
293         (void) mutex_unlock(&udp->atfork_lock);
294 }

296 /*
297  * The following is a routine which the loader (ld.so.1) calls when it
298  * processes a dlclose call on an object.  It sets the destructor
299  * function pointer to NULL for all keys whose destructors fall within
300  * the union of the ranges specified by the elements of the array range.
301  * We don't assign TSD_UNALLOCATED (the equivalent of pthread_key_destroy())
302  * because the thread may use the key's TSD further on in fini processing.
303  */
304 static void
305 _preexec_tsd_unload(Lc_addr_range_t range[], uint_t count)
306 {
307         tsd_metadata_t *tsdm = &curthread->ul_uberdata->tsd_metadata;
308         void (*func)(void *);
309         int key;

311         lmutex_lock(&tsdm->tsdm_lock);
312         for (key = 1; key < tsdm->tsdm_nused; key++) {
313                 if ((func = tsdm->tsdm_destro[key]) != NULL &&
314                     func != TSD_UNALLOCATED &&
315                     in_range((void *)func, range, count))
222                     in_range((_exithdlr_func_t)func, range, count))
316                         tsdm->tsdm_destro[key] = NULL;
317         }
318         lmutex_unlock(&tsdm->tsdm_lock);
```



```
319 }

321 /*
322  * The following is a routine which the loader (ld.so.1) calls when it
323  * processes dlclose calls on objects with atexit registrations.  It
324  * executes the exit handlers that fall within the union of the ranges
325  * specified by the elements of the array range in the REVERSE ORDER of
326  * their registration.  Do not change this characteristic; it is REQUIRED
327  * BEHAVIOR.
328  */
329 int
330 _preexec_exit_handlers(Lc_addr_range_t range[], uint_t count)
331 {
332         atexit_root_t *arp = &curthread->ul_uberdata->atexit_root;
333         _exthdlr_t *o;          /* previous node */
334         _exthdlr_t *p;          /* this node */
335         int cancel_state;

337         /* disable cancellation while running atexit handlers */
338         (void) pthread_setcancelstate(PTHREAD_CANCEL_DISABLE, &cancel_state);
339         (void) mutex_lock(&arp->exitfns_lock);
340         o = NULL;
341         p = arp->head;
342         while (p != NULL) {
343                 /*
344                  * We call even CXA handlers of functions present in the
345                  * library being unloaded.  The specification isn't
346                  * particularly clear on this, and this seems the most sane.
347                  * This is the behaviour of FreeBSD 9.1 (GNU libc leaves the
348                  * handler on the exit list, and crashes at exit time).
349                  *
350                  * This won't cause handlers to be called twice, because
351                  * anything called from a __cxa_finalize call from the
352                  * language runtime will have been removed from the list.
353                  */
354                 if (in_range((void *)p->hdlr, range, count)) {
250                 if (in_range(p->hdlr, range, count)) {
355                         /* We need to execute this one */
356                         if (o != NULL)
357                                 o->next = p->next;
358                         else
359                                 arp->head = p->next;
360                         p->hdlr(p->arg);
256                         p->hdlr();
361                         lfree(p, sizeof (_exthdlr_t));
362                         o = NULL;
363                         p = arp->head;
364                 } else {
365                         o = p;
366                         p = p->next;
367                 }
368         }
369         (void) mutex_unlock(&arp->exitfns_lock);
370         (void) pthread_setcancelstate(cancel_state, NULL);

372         _preexec_tsd_unload(range, count);
373         _preexec_atfork_unload(range, count);
374         _preexec_sig_unload(range, count);

376         return (0);
377 }

379 static int
380 in_range(void *addr, Lc_addr_range_t ranges[], uint_t count)
276 in_range(_exithdlr_func_t addr, Lc_addr_range_t ranges[], uint_t count)
381 {
```

```
382         uint_t idx;

384         for (idx = 0; idx < count; idx++) {
385                 if (addr >= ranges[idx].lb &&
386                     addr < ranges[idx].ub) {
281                 if ((void *)addr >= ranges[idx].lb &&
282                     (void *)addr < ranges[idx].ub) {
387                         return (1);
388                 }
389         }

391         return (0);
392 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   54742 Fri Jun 28 20:46:50 2013
new/usr/src/lib/libc/port/mapfile-vers
3849 implement __cxa_atexit/__cxa_finalize
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.

#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#

$mapfile_version 2

#
# All function names added to this or any other libc mapfile
# must be placed under the 'protected:' designation.
# The 'global:' designation is used *only* for data
# items and for the members of the malloc() family.
#

#
# README README README README README README: how to update this file
#   1) each version of Solaris/OpenSolaris gets a version number.
#      (Actually since Solaris is actually a series of OpenSolaris releases
#       we'll just use OpenSolaris for this exercise.)
#       OpenSolaris 2008.11 gets 1.23
#       OpenSolaris 2009.04 gets 1.24
#       etc.
#   2) each project integration uses a unique version number.
```



```
#       PSARC/2008/123 gets 1.24.1
#       PSARC/2008/456 gets 1.24.2
#       etc.
#


# Mnemonic conditional input identifiers:
#
# - amd64, i386, sparc32, sparcv9: Correspond to ISA subdirectories used to
#       hold per-platform code. Note however that we use 'sparc32' instead of
#       'sparc'. Since '_sparc' is predefined to apply to, all sparc platforms,
#       naming the 32-bit version 'sparc' would be too likely to cause errors.
#
# -     lf64: Defined on platforms that offer the 32-bit largefile APIs
#
$if _ELF32
$add lf64
$endif
$if _sparc && _ELF32
$add sparc32
$endif
$if _sparc && _ELF64
$add sparcv9
$endif
$if _x86 && _ELF32
$add i386
$endif
$if _x86 && _ELF64
$add amd64
$endif

SYMBOL_VERSION ILLUMOS_0.5 {    # common C++ ABI exit handlers
    protected:
        __cxa_atexit;
        __cxa_finalize;
} ILLUMOS_0.4;

#endif /* ! codereview */
SYMBOL_VERSION ILLUMOS_0.4 {    # Illumos additions
    protected:
        pipe2;
        dup3;
        mkostemp;
        mkostemps;

$if lf64
        mkostemp64;
        mkostemps64;
$endif
} ILLUMOS_0.3;

SYMBOL_VERSION ILLUMOS_0.3 {    # Illumos additions
    protected:
        assfail3;
} ILLUMOS_0.2;

SYMBOL_VERSION ILLUMOS_0.2 {    # Illumos additions
    protected:
        posix_spawn_pipe_np;
} ILLUMOS_0.1;

SYMBOL_VERSION ILLUMOS_0.1 {    # Illumos additions
    protected:
        timegm;
} SUNW_1.23;
```

```
128 SYMBOL_VERSION SUNW_1.23 {      # SunOS 5.11 (Solaris 11)
129     global:
130         _nl_domain_bindings;
131         _nl_msg_cat_cntr;

133 $if _ELF32
134         dl_iterate_phdr { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
135 $elif sparcv9
136         dl_iterate_phdr { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
137 $elif amd64
138         dl_iterate_phdr { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
139 $else
140 $error unknown platform
141 $endif

143     protected:

145 $if sparc32
146         __align_cpy_1;
147 $endif

149         addrtosymstr;
150         aio_cancel;
151         aiocancel;
152         aio_error;
153         aio_fsync;
154         aio_read;
155         aioread;
156         aio_return;
157         aio_suspend;
158         aiowait;
159         aio_waitn;
160         aio_write;
161         aiowrite;
162         asprintf;
163         assfail;
164         backtrace;
165         backtrace_symbols;
166         backtrace_symbols_fd;
167         canonicalize_file_name;
168         clearenv;
169         clock_getres;
170         clock_gettime;
171         clock_nanosleep;
172         clock_settime;
173         daemon;
174         dirfd;
175         door_bind;
176         door_call;
177         door_create;
178         door_cred;
179         door_getparam;
180         door_info;
181         door_return;
182         door_revoke;
183         door_server_create;
184         door_setparam;
185         door_ucred;
186         door_unbind;
187         door_xcreate;
188         err;
189         errx;
190         faccessat;
191         fchmodat;
192         fcloseall;
193         fdatasync;
```



```
194         ffsl;
195         ffsll;
196         fgetattr;
197         fls;
198         flsl;
199         flsll;
200         forkallx;
201         forkx;
202         fsetattr;
203         getattrat;
204         getdelim;
205         getline;
206         get_nprocs;
207         get_nprocs_conf;
208         getprogname;
209         htonl;
210         htonll;
211         htons;
212         linkat;
213         lio_listio;
214         memmem;
215         mkdirat;
216         mkdtemp;
217         mkfifoat;
218         mknodat;
219         mkstemps;
220         mmapobj;
221         mq_close;
222         mq_getattr;
223         mq_notify;
224         mq_open;
225         mq_receive;
226         mq_reltimedreceive_np;
227         mq_reltimedsend_np;
228         mq_send;
229         mq_setattr;
230         mq_timedreceive;
231         mq_timedsend;
232         mq_unlink;
233         nanosleep;
234         ntohl;
235         ntohll;
236         ntohs;
237         posix_fadvise;
238         posix_fallocate;
239         posix_madvise;
240         posix_memalign;
241         posix_spawn_file_actions_addclosefrom_np;
242         posix_spawnattr_getsigignore_np;
243         posix_spawnattr_setsigignore_np;
244         ppoll;
245         priv_basicset;
246         pthread_key_create_once_np;
247         pthread_mutexattr_getrobust;
248         pthread_mutexattr_setrobust;
249         pthread_mutex_consistent;
250         readlinkat;
251         sched_getparam;
252         sched_get_priority_max;
253         sched_get_priority_min;
254         sched_getscheduler;
255         sched_rr_get_interval;
256         sched_setparam;
257         sched_setscheduler;
258         sched_yield;
259         sem_close;
```

```
260         sem_destroy;
261         sem_getvalue;
262         sem_init;
263         sem_open;
264         sem_post;
265         sem_reltimedwait_np;
266         sem_timedwait;
267         sem_trywait;
268         sem_unlink;
269         sem_wait;
270         setattrat;
271         setprogname;
272         _sharefs;
273         shm_open;
274         shm_unlink;
275         sigqueue;
276         sigtimedwait;
277         sigwaitinfo;
278         smt_pause;
279         stpcpy;
280         stpncpy;
281         strcasestr;
282         strchrnul;
283         strndup;
284         strnlen;
285         strnstr;
286         strsep;
287         symlinkat;
288         thr_keycreate_once;
289         timer_create;
290         timer_delete;
291         timer_getoverrun;
292         timer_gettime;
293         timer_settime;
294         u8_strcmp;
295         u8_validate;
296         uconv_u16tou32;
297         uconv_u16tou8;
298         uconv_u32tou16;
299         uconv_u32tou8;
300         uconv_u8tou16;
301         uconv_u8tou32;
302         vasprintf;
303         verr;
304         verrx;
305         vforkx;
306         vwarn;
307         vwarnx;
308         warn;
309         warnx;
310         wcpcpy;
311         wcpncpy;
312         wcscasecmp;
313         wcsdup;
314         wcsncasecmp;
315         wcsnlen;

317 $if lf64
318         aio_cancel64;
319         aio_error64;
320         aio_fsync64;
321         aio_read64;
322         aioread64;
323         aio_return64;
324         aio_suspend64;
325         aio_waitn64;
```



```
326         aio_write64;
327         aiowrite64;
328         lio_listio64;
329         mkstemps64;
330         posix_fadvise64;
331         posix_fallocate64;
332 $endif
333 } SUNW_1.22.6;

335 SYMBOL_VERSION SUNW_1.22.6 {    # s10u9 - SunOS 5.10 (Solaris 10) patch addition
336     protected:
337         futimens;
338         utimensat;
339 } SUNW_1.22.5;

341 SYMBOL_VERSION SUNW_1.22.5 {    # s10u8 - SunOS 5.10 (Solaris 10) patch addition
342     protected:
343         getpagesizes2;
344 } SUNW_1.22.4;

346 SYMBOL_VERSION SUNW_1.22.4 {    # s10u7 - SunOS 5.10 (Solaris 10) patch addition
347     protected:
348         SUNW_1.22.4;
349 } SUNW_1.22.3;

351 SYMBOL_VERSION SUNW_1.22.3 {    # SunOS 5.10 (Solaris 10) patch additions
352     protected:
353         mutex_consistent;
354         u8_textprep_str;
355         uucopy;
356         uucopystr;
357 } SUNW_1.22.2;

359 SYMBOL_VERSION SUNW_1.22.2 {    # SunOS 5.10 (Solaris 10) patch additions
360     protected:
361         is_system_labeled;
362         ucred_getlabel;
363         _ucred_getlabel;
364 } SUNW_1.22.1;

366 SYMBOL_VERSION SUNW_1.22.1 {    # SunOS 5.10 (Solaris 10) patch additions
367     protected:
368         atomic_add_8;
369         atomic_add_8_nv;
370         atomic_add_char         { FLAGS = NODYNSORT };
371         atomic_add_char_nv      { FLAGS = NODYNSORT };
372         atomic_add_int          { FLAGS = NODYNSORT };
373         atomic_add_int_nv       { FLAGS = NODYNSORT };
374         atomic_add_ptr          { FLAGS = NODYNSORT };
375         atomic_add_ptr_nv       { FLAGS = NODYNSORT };
376         atomic_add_short        { FLAGS = NODYNSORT };
377         atomic_add_short_nv     { FLAGS = NODYNSORT };
378         atomic_and_16;
379         atomic_and_16_nv;
380         atomic_and_32_nv;
381         atomic_and_64;
382         atomic_and_64_nv;
383         atomic_and_8;
384         atomic_and_8_nv;
385         atomic_and_uchar        { FLAGS = NODYNSORT };
386         atomic_and_uchar_nv     { FLAGS = NODYNSORT };
387         atomic_and_uint_nv      { FLAGS = NODYNSORT };
388         atomic_and_ulong        { FLAGS = NODYNSORT };
389         atomic_and_ulong_nv     { FLAGS = NODYNSORT };
390         atomic_and_ushort       { FLAGS = NODYNSORT };
391         atomic_and_ushort_nv    { FLAGS = NODYNSORT };
```

```
392         atomic_cas_16;
393         atomic_cas_32;
394         atomic_cas_64;
395         atomic_cas_8;
396         atomic_cas_ptr          { FLAGS = NODYNSORT };
397         atomic_cas_uchar        { FLAGS = NODYNSORT };
398         atomic_cas_uint         { FLAGS = NODYNSORT };
399         atomic_cas_ulong        { FLAGS = NODYNSORT };
400         atomic_cas_ushort       { FLAGS = NODYNSORT };
401         atomic_clear_long_excl  { FLAGS = NODYNSORT };
402         atomic_dec_16;
403         atomic_dec_16_nv;
404         atomic_dec_32;
405         atomic_dec_32_nv;
406         atomic_dec_64;
407         atomic_dec_64_nv;
408         atomic_dec_8;
409         atomic_dec_8_nv;
410         atomic_dec_uchar        { FLAGS = NODYNSORT };
411         atomic_dec_uchar_nv     { FLAGS = NODYNSORT };
412         atomic_dec_uint         { FLAGS = NODYNSORT };
413         atomic_dec_uint_nv      { FLAGS = NODYNSORT };
414         atomic_dec_ulong        { FLAGS = NODYNSORT };
415         atomic_dec_ulong_nv     { FLAGS = NODYNSORT };
416         atomic_dec_ushort       { FLAGS = NODYNSORT };
417         atomic_dec_ushort_nv    { FLAGS = NODYNSORT };
418         atomic_inc_16;
419         atomic_inc_16_nv;
420         atomic_inc_32;
421         atomic_inc_32_nv;
422         atomic_inc_64;
423         atomic_inc_64_nv;
424         atomic_inc_8;
425         atomic_inc_8_nv;
426         atomic_inc_uchar        { FLAGS = NODYNSORT };
427         atomic_inc_uchar_nv     { FLAGS = NODYNSORT };
428         atomic_inc_uint         { FLAGS = NODYNSORT };
429         atomic_inc_uint_nv      { FLAGS = NODYNSORT };
430         atomic_inc_ulong        { FLAGS = NODYNSORT };
431         atomic_inc_ulong_nv     { FLAGS = NODYNSORT };
432         atomic_inc_ushort       { FLAGS = NODYNSORT };
433         atomic_inc_ushort_nv    { FLAGS = NODYNSORT };
434         atomic_or_16;
435         atomic_or_16_nv;
436         atomic_or_32_nv;
437         atomic_or_64;
438         atomic_or_64_nv;
439         atomic_or_8;
440         atomic_or_8_nv;
441         atomic_or_uchar         { FLAGS = NODYNSORT };
442         atomic_or_uchar_nv      { FLAGS = NODYNSORT };
443         atomic_or_uint_nv       { FLAGS = NODYNSORT };
444         atomic_or_ulong         { FLAGS = NODYNSORT };
445         atomic_or_ulong_nv      { FLAGS = NODYNSORT };
446         atomic_or_ushort        { FLAGS = NODYNSORT };
447         atomic_or_ushort_nv     { FLAGS = NODYNSORT };
448         atomic_set_long_excl    { FLAGS = NODYNSORT };
449         atomic_swap_16;
450         atomic_swap_32;
451         atomic_swap_64;
452         atomic_swap_8;
453         atomic_swap_ptr         { FLAGS = NODYNSORT };
454         atomic_swap_uchar       { FLAGS = NODYNSORT };
455         atomic_swap_uint        { FLAGS = NODYNSORT };
456         atomic_swap_ulong       { FLAGS = NODYNSORT };
457         atomic_swap_ushort      { FLAGS = NODYNSORT };
```



```
458         membar_consumer;
459         membar_enter;
460         membar_exit;
461         membar_producer;

463 $if _ELF32
464         enable_extended_FILE_stdio;
465 $endif

467 $if i386
468         # Note: atomic_[and,dec,inc,or]_64_nv are also defined above. Here,
469         # we add the NODYNSORT attribute to them. On this platform, they are
470         # aliases for the non-_nv versions. If that is changed, these lines
471         # should be removed.
472         atomic_and_64_nv        { FLAGS = NODYNSORT };
473         atomic_dec_64_nv        { FLAGS = NODYNSORT };
474         atomic_inc_64_nv        { FLAGS = NODYNSORT };
475         atomic_or_64_nv         { FLAGS = NODYNSORT };
476 $endif
477 $if _sparc
478         # Note: atomic_OP_WIDTH_nv symbols are also defined above. Here,
479         # we add the NODYNSORT attribute to them. On this platform, they are
480         # aliases for the non-_nv versions. If that is changed, these lines
481         # should be removed.
482         atomic_add_8_nv         { FLAGS = NODYNSORT };
483         atomic_and_8_nv         { FLAGS = NODYNSORT };
484         atomic_and_16_nv        { FLAGS = NODYNSORT };
485         atomic_and_32_nv        { FLAGS = NODYNSORT };
486         atomic_and_64_nv        { FLAGS = NODYNSORT };
487         atomic_dec_8_nv         { FLAGS = NODYNSORT };
488         atomic_dec_16_nv        { FLAGS = NODYNSORT };
489         atomic_dec_32_nv        { FLAGS = NODYNSORT };
490         atomic_dec_64_nv        { FLAGS = NODYNSORT };
491         atomic_inc_8_nv         { FLAGS = NODYNSORT };
492         atomic_inc_16_nv        { FLAGS = NODYNSORT };
493         atomic_inc_32_nv        { FLAGS = NODYNSORT };
494         atomic_inc_64_nv        { FLAGS = NODYNSORT };
495         atomic_or_8_nv          { FLAGS = NODYNSORT };
496         atomic_or_16_nv         { FLAGS = NODYNSORT };
497         atomic_or_32_nv         { FLAGS = NODYNSORT };
498         atomic_or_64_nv         { FLAGS = NODYNSORT };
499 $endif
500 } SUNW_1.22;

502 SYMBOL_VERSION SUNW_1.22 {      # SunOS 5.10 (Solaris 10)
503     global:
504 $if _ELF32
505         dladdr          { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
506         dladdr1         { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
507         dlclose         { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
508         dldump          { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
509         dlerror         { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
510         dlinfo          { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
511         dlmopen         { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
512         dlopen          { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
513         dlsym           { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
514 $elif sparcv9
515         dladdr          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
516         dladdr1         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
517         dlclose         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
518         dldump          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
519         dlerror         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
520         dlinfo          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
521         dlmopen         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
522         dlopen          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
523         dlsym           { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
```

```
$elif amd64
        dladdr          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dladdr1         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlamd64getunwind { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlclose         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dldump          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlerror         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlinfo          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlmopen         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlopen          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        dlsym           { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
$else
$error unknown platform
$endif

    protected:
        alphasort;
        _alphasort;
        atomic_add_16;
        atomic_add_16_nv;
        atomic_add_32;
        atomic_add_32_nv;
        atomic_add_64;
        atomic_add_64_nv;
        atomic_add_long         { FLAGS = NODYNSORT };
        atomic_add_long_nv      { FLAGS = NODYNSORT };
        atomic_and_32;
        atomic_and_uint         { FLAGS = NODYNSORT };
        atomic_or_32;
        atomic_or_uint          { FLAGS = NODYNSORT };
        _Exit;
        getisax;
        _getisax;
        getopt_clip;
        _getopt_clip;
        getopt_long;
        _getopt_long;
        getopt_long_only;
        _getopt_long_only;
        getpeerucred;
        _getpeerucred;
        getpflags;
        _getpflags;
        getppriv;
        _getppriv;
        getprivimplinfo;
        _getprivimplinfo;
        getzoneid;
        getzoneidbyname;
        getzonenamebyid;
        imaxabs;
        imaxdiv;
        isblank;
        iswblank;
        port_alert;
        port_associate;
        port_create;
        port_dissociate;
        port_get;
        port_getn;
        port_send;
        port_sendn;
        posix_openpt;
        posix_spawn;
        posix_spawnattr_destroy;
        posix_spawnattr_getflags;
```



```
        posix_spawnattr_getpgroup;
        posix_spawnattr_getschedparam;
        posix_spawnattr_getschedpolicy;
        posix_spawnattr_getsigdefault;
        posix_spawnattr_getsigmask;
        posix_spawnattr_init;
        posix_spawnattr_setflags;
        posix_spawnattr_setpgroup;
        posix_spawnattr_setschedparam;
        posix_spawnattr_setschedpolicy;
        posix_spawnattr_setsigdefault;
        posix_spawnattr_setsigmask;
        posix_spawn_file_actions_addclose;
        posix_spawn_file_actions_adddup2;
        posix_spawn_file_actions_addopen;
        posix_spawn_file_actions_destroy;
        posix_spawn_file_actions_init;
        posix_spawnp;
        priv_addset;
        _priv_addset;
        priv_allocset;
        _priv_allocset;
        priv_copyset;
        _priv_copyset;
        priv_delset;
        _priv_delset;
        priv_emptyset;
        _priv_emptyset;
        priv_fillset;
        _priv_fillset;
        __priv_free_info;
        priv_freeset;
        _priv_freeset;
        priv_getbyname;
        _priv_getbyname;
        __priv_getbyname;
        priv_getbynum;
        _priv_getbynum;
        __priv_getbynum;
        __priv_getdata;
        priv_getsetbyname;
        _priv_getsetbyname;
        __priv_getsetbyname;
        priv_getsetbynum;
        _priv_getsetbynum;
        __priv_getsetbynum;
        priv_gettext;
        _priv_gettext;
        priv_ineffect;
        _priv_ineffect;
        priv_intersect;
        _priv_intersect;
        priv_inverse;
        _priv_inverse;
        priv_isemptyset;
        _priv_isemptyset;
        priv_isequalset;
        _priv_isequalset;
        priv_isfullset;
        _priv_isfullset;
        priv_ismember;
        _priv_ismember;
        priv_issubset;
        _priv_issubset;
        __priv_parse_info;
        priv_set;
```

```
656         _priv_set;
657         priv_set_to_str;
658         _priv_set_to_str;
659         __priv_set_to_str;
660         priv_str_to_set;
661         _priv_str_to_set;
662         priv_union;
663         _priv_union;
664         pselect;
665         pthread_attr_getstack;
666         pthread_attr_setstack;
667         pthread_barrierattr_destroy;
668         pthread_barrierattr_getpshared;
669         pthread_barrierattr_init;
670         pthread_barrierattr_setpshared;
671         pthread_barrier_destroy;
672         pthread_barrier_init;
673         pthread_barrier_wait;
674         pthread_condattr_getclock;
675         pthread_condattr_setclock;
676         pthread_mutexattr_getrobust_np  { FLAGS = NODYNSORT };
677         pthread_mutexattr_setrobust_np  { FLAGS = NODYNSORT };
678         pthread_mutex_consistent_np     { FLAGS = NODYNSORT };
679         pthread_mutex_reltimedlock_np;
680         pthread_mutex_timedlock;
681         pthread_rwlock_reltimedrdlock_np;
682         pthread_rwlock_reltimedwrlock_np;
683         pthread_rwlock_timedrdlock;
684         pthread_rwlock_timedwrlock;
685         pthread_setschedprio;
686         pthread_spin_destroy;
687         pthread_spin_init;
688         pthread_spin_lock;
689         pthread_spin_trylock;
690         pthread_spin_unlock;
691         rctlblk_set_recipient_pid;
692         scandir;
693         _scandir;
694         schedctl_exit;
695         schedctl_init;
696         schedctl_lookup;
697         sema_reltimedwait;
698         sema_timedwait;
699         setenv;
700         setpflags;
701         _setpflags;
702         setppriv;
703         _setppriv;
704         strerror_r;
705         strtof;
706         strtoimax;
707         strtold;
708         strtoumax;
709         ucred_free;
710         _ucred_free;
711         ucred_get;
712         _ucred_get;
713         ucred_getegid;
714         _ucred_getegid;
715         ucred_geteuid;
716         _ucred_geteuid;
717         ucred_getgroups;
718         _ucred_getgroups;
719         ucred_getpflags;
720         _ucred_getpflags;
721         ucred_getpid;
```



```
722         _ucred_getpid;
723         ucred_getprivset;
724         _ucred_getprivset;
725         ucred_getprojid;
726         _ucred_getprojid;
727         ucred_getrgid;
728         _ucred_getrgid;
729         ucred_getruid;
730         _ucred_getruid;
731         ucred_getsgid;
732         _ucred_getsgid;
733         ucred_getsuid;
734         _ucred_getsuid;
735         ucred_getzoneid;
736         _ucred_getzoneid;
737         ucred_size;
738         _ucred_size;
739         unsetenv;
740         wcstof;
741         wcstoimax;
742         wcstold;
743         wcstoll;
744         wcstoull;
745         wcstoumax;

747 $if lf64
748         alphasort64;
749         _alphasort64;
750         pselect_large_fdset;
751         scandir64;
752         _scandir64;
753 $endif

755 $if _ELF64
756         walkcontext;
757 $endif

759 $if _sparc
760         # Note: atomic_add_[16,32,64]_nv are also defined above. Here, we add
761         # the NODYNSORT attribute to them. On this platform, they are aliases
762         # for the non-_nv versions. If that is changed, these lines should be
763         # removed.
764         atomic_add_16_nv        { FLAGS = NODYNSORT };
765         atomic_add_32_nv        { FLAGS = NODYNSORT };
766         atomic_add_64_nv        { FLAGS = NODYNSORT };
767 $endif

769 $if i386
770         # Note: atomic_add_64_nv is also defined above. Here, we add the
771         # NODYNSORT attribute to it. On this platform, it is an aliases for
772         # atomic_add_64. If that is changed, this line should be removed.
773         atomic_add_64_nv        { FLAGS = NODYNSORT };
774 $endif

776 $if amd64
777         # Exception unwind APIs required by the amd64 ABI
778         _SUNW_Unwind_DeleteException;
779         _SUNW_Unwind_ForcedUnwind;
780         _SUNW_Unwind_GetCFA;
781         _SUNW_Unwind_GetGR;
782         _SUNW_Unwind_GetIP;
783         _SUNW_Unwind_GetLanguageSpecificData;
784         _SUNW_Unwind_GetRegionStart;
785         _SUNW_Unwind_RaiseException;
786         _SUNW_Unwind_Resume;
787         _SUNW_Unwind_SetGR;
```

```
788         _SUNW_Unwind_SetIP;
789         _UA_CLEANUP_PHASE;
790         _UA_FORCE_UNWIND;
791         _UA_HANDLER_FRAME;
792         _UA_SEARCH_PHASE;
793         _Unwind_DeleteException;
794         _Unwind_ForcedUnwind;
795         _Unwind_GetCFA;
796         _Unwind_GetGR;
797         _Unwind_GetIP;
798         _Unwind_GetLanguageSpecificData;
799         _Unwind_GetRegionStart;
800         _Unwind_RaiseException;
801         _Unwind_Resume;
802         _Unwind_SetGR;
803         _Unwind_SetIP;
804 $endif
805 } SUNW_1.21.3;

807 SYMBOL_VERSION SUNW_1.21.3 {    # SunOS 5.9 (Solaris 9) patch additions
808     protected:
809         forkall;
810 } SUNW_1.21.2;

812 SYMBOL_VERSION SUNW_1.21.2 {    # SunOS 5.9 (Solaris 9) patch additions
813     protected:
814         getustack;
815         _getustack;
816         setustack;
817         _setustack;
818         stack_getbounds;
819         _stack_getbounds;
820         _stack_grow;
821         stack_inbounds;
822         _stack_inbounds;
823         stack_setbounds;
824         _stack_setbounds;
825         stack_violation;
826         _stack_violation;

828 $if _sparc
829         __makecontext_v2;
830         ___makecontext_v2;
831 $endif
832 } SUNW_1.21.1;

834 SYMBOL_VERSION SUNW_1.21.1 {    # SunOS 5.9 (Solaris 9) patch additions
835     protected:
836         crypt_gensalt;
837 } SUNW_1.21;

839 SYMBOL_VERSION SUNW_1.21 {      # SunOS 5.9 (Solaris 9)
840     protected:
841         attropen;
842         _attropen;
843         bind_textdomain_codeset;
844         closefrom;
845         _closefrom;
846         cond_reltimedwait;
847         dcngettext;
848         dngettext;
849         fchownat;
850         _fchownat;
851         fdopendir;
852         _fdopendir;
853         fdwalk;
```



```
854         _fdwalk;
855         fstatat;
856         _fstatat;
857         futimesat;
858         _futimesat;
859         getcpuid;
860         _getcpuid;
861         gethomelgroup;
862         _gethomelgroup          { FLAGS = NODYNSORT };
863         getpagesizes;
864         getrctl;
865         _getrctl;
866         issetugid;
867         _issetugid;
868         _lwp_cond_reltimedwait;
869         meminfo;
870         _meminfo;
871         ngettext;
872         openat;
873         _openat;
874         printstack;
875         priocntl;
876         priocntlset;
877         pset_getattr;
878         pset_getloadavg;
879         pset_list;
880         pset_setattr;
881         pthread_cond_reltimedwait_np;
882         rctlblk_get_enforced_value;
883         rctlblk_get_firing_time;
884         rctlblk_get_global_action;
885         rctlblk_get_global_flags;
886         rctlblk_get_local_action;
887         rctlblk_get_local_flags;
888         rctlblk_get_privilege;
889         rctlblk_get_recipient_pid;
890         rctlblk_get_value;
891         rctlblk_set_local_action;
892         rctlblk_set_local_flags;
893         rctlblk_set_privilege;
894         rctlblk_set_value;
895         rctlblk_size;
896         rctl_walk;
897         renameat;
898         setrctl;
899         _setrctl;
900         unlinkat;
901         _unlinkat;
902         vfscanf;
903         _vfscanf;
904         vfwscanf;
905         vscanf;
906         _vscanf;
907         vsscanf;
908         _vsscanf;
909         vswscanf;
910         vwscanf;

912 $if _ELF32
913         walkcontext;
914 $endif

916 $if lf64
917         attropen64;
918         _attropen64;
919         fstatat64;
```

```
920         _fstatat64;
921         openat64;
922         _openat64;
923 $endif
924 } SUNW_1.20.4;

926 SYMBOL_VERSION SUNW_1.20.4 {    # SunOS 5.8 (Solaris 8) patch additions
927     protected:
928         semtimedop;
929         _semtimedop;
930 } SUNW_1.20.1;

932 SYMBOL_VERSION SUNW_1.20.1 {    # SunOS 5.8 (Solaris 8) patch additions
933     protected:
934         getacct;
935         _getacct;
936         getprojid;
937         _getprojid;
938         gettaskid;
939         _gettaskid;
940         msgids;
941         _msgids;
942         msgsnap;
943         _msgsnap;
944         putacct;
945         _putacct;
946         semids;
947         _semids;
948         settaskid;
949         _settaskid;
950         shmids;
951         _shmids;
952         wracct;
953         _wracct;
954 } SUNW_1.20;

956 SYMBOL_VERSION SUNW_1.20 {      # SunOS 5.8 (Solaris 8)
957     protected:
958         getextmntent;
959         resetmnttab;
960 } SUNW_1.19;

962 SYMBOL_VERSION SUNW_1.19 {
963     protected:
964         strlcat;
965         strlcpy;
966         umount2;
967         _umount2;
968 } SUNW_1.18.1;

970 SYMBOL_VERSION SUNW_1.18.1 {
971     protected:
972         __fsetlocking;
973 } SUNW_1.18;

975 SYMBOL_VERSION SUNW_1.18 {      # SunOS 5.7 (Solaris 7)
976     protected:
977         btowc;
978         __fbufsize;
979         __flbf;
980         _flushlbf;
981         __fpending;
982         __fpurge;
983         __freadable;
984         __freading;
985         fwide;
```



```
986         fwprintf;
987         __fwritable;
988         __fwriting;
989         fwscanf;
990         getloadavg;
991         isaexec;
992         mbrlen;
993         mbrtowc;
994         mbsinit;
995         mbsrtowcs;
996         pcsample;
997         pthread_attr_getguardsize;
998         pthread_attr_setguardsize;
999         pthread_getconcurrency;
1000        pthread_mutexattr_gettype;
1001        pthread_mutexattr_settype;
1002        pthread_rwlockattr_destroy;
1003        pthread_rwlockattr_getpshared;
1004        pthread_rwlockattr_init;
1005        pthread_rwlockattr_setpshared;
1006        pthread_rwlock_destroy;
1007        pthread_rwlock_init;
1008        pthread_rwlock_rdlock;
1009        pthread_rwlock_tryrdlock;
1010        pthread_rwlock_trywrlock;
1011        pthread_rwlock_unlock;
1012        pthread_rwlock_wrlock;
1013        pthread_setconcurrency;
1014        swprintf;
1015        swscanf;
1016        __sysconf_xpg5;
1017        vfwprintf;
1018        vswprintf;
1019        vwprintf;
1020        wcrtomb;
1021        wcsrtombs;
1022        wcsstr;
1023        wctob;
1024        wmemchr;
1025        wmemcmp;
1026        wmemcpy;
1027        wmemmove;
1028        wmemset;
1029        wprintf;
1030        wscanf;

1032 $if _ELF32
1033         select_large_fdset;
1034 $endif
1035 } SUNW_1.17;

1037 # The empty versions SUNW_1.2 through SUNW_1.17 must be preserved because
1038 # applications built on Solaris 2.6 Beta (when they did contain symbols)
1039 # may depend on them.  All symbol content for SunOS 5.6 is now in SUNW_1.1

1041 SYMBOL_VERSION SUNW_1.17 {
1042     protected:
1043         SUNW_1.17;
1044 } SUNW_1.16;

1046 SYMBOL_VERSION SUNW_1.16 {
1047     protected:
1048         SUNW_1.16;
1049 } SUNW_1.15;

1051 SYMBOL_VERSION SUNW_1.15 {
```

```
1052     protected:
1053         SUNW_1.15;
1054 } SUNW_1.14;

1056 SYMBOL_VERSION SUNW_1.14 {
1057     protected:
1058         SUNW_1.14;
1059 } SUNW_1.13;

1061 SYMBOL_VERSION SUNW_1.13 {
1062     protected:
1063         SUNW_1.13;
1064 } SUNW_1.12;

1066 SYMBOL_VERSION SUNW_1.12 {
1067     protected:
1068         SUNW_1.12;
1069 } SUNW_1.11;

1071 SYMBOL_VERSION SUNW_1.11 {
1072     protected:
1073         SUNW_1.11;
1074 } SUNW_1.10;

1076 SYMBOL_VERSION SUNW_1.10 {
1077     protected:
1078         SUNW_1.10;
1079 } SUNW_1.9;

1081 SYMBOL_VERSION SUNW_1.9 {
1082     protected:
1083         SUNW_1.9;
1084 } SUNW_1.8;

1086 SYMBOL_VERSION SUNW_1.8 {
1087     protected:
1088         SUNW_1.8;
1089 } SUNW_1.7;

1091 SYMBOL_VERSION SUNW_1.7 {
1092     protected:
1093         SUNW_1.7;
1094 } SUNW_1.6;

1096 SYMBOL_VERSION SUNW_1.6 {
1097     protected:
1098         SUNW_1.6;
1099 } SUNW_1.5;

1101 SYMBOL_VERSION SUNW_1.5 {
1102     protected:
1103         SUNW_1.5;
1104 } SUNW_1.4;

1106 SYMBOL_VERSION SUNW_1.4 {
1107     protected:
1108         SUNW_1.4;
1109 } SUNW_1.3;

1111 SYMBOL_VERSION SUNW_1.3 {
1112     protected:
1113         SUNW_1.3;
1114 } SUNW_1.2;

1116 SYMBOL_VERSION SUNW_1.2 {
1117     protected:
```



```
1118         SUNW_1.2;
1119 } SUNW_1.1;

1121 SYMBOL_VERSION SUNW_1.1 {       # SunOS 5.6 (Solaris 2.6)
1122     global:
1123         __loc1;
1124     protected:
1125         basename;
1126         bindtextdomain;
1127         bsd_signal;
1128         dbm_clearerr;
1129         dbm_error;
1130         dcgettext;
1131         dgettext;
1132         directio;
1133         dirname;
1134         endusershell;
1135         _exithandle;
1136         fgetwc;
1137         fgetws;
1138         fpgetround;
1139         fpsetround;
1140         fputwc;
1141         fputws;
1142         fseeko;
1143         ftello;
1144         ftrylockfile;
1145         getexecname;
1146         _getexecname;
1147         getpassphrase;
1148         gettext;
1149         getusershell;
1150         getwc;
1151         getwchar;
1152         getws;
1153         isenglish;
1154         isideogram;
1155         isnumber;
1156         isphonogram;
1157         isspecial;
1158         iswalnum;
1159         iswalpha;
1160         iswcntrl;
1161         iswctype;
1162         iswdigit;
1163         iswgraph;
1164         iswlower;
1165         iswprint;
1166         iswpunct;
1167         iswspace;
1168         iswupper;
1169         iswxdigit;
1170         ___loc1;
1171         _longjmp;
1172         _lwp_sema_trywait;
1173         ntp_adjtime;
1174         _ntp_adjtime;
1175         ntp_gettime;
1176         _ntp_gettime;
1177         __posix_asctime_r;
1178         __posix_ctime_r;
1179         __posix_getgrgid_r;
1180         __posix_getgrnam_r;
1181         __posix_getlogin_r;
1182         __posix_getpwnam_r;
1183         __posix_getpwuid_r;
```

```
1184         __posix_sigwait;
1185         __posix_ttyname_r;
1186         pset_assign;
1187         pset_bind;
1188         pset_create;
1189         pset_destroy;
1190         pset_info;
1191         pthread_atfork;
1192         pthread_attr_destroy;
1193         pthread_attr_getdetachstate;
1194         pthread_attr_getinheritsched;
1195         pthread_attr_getschedparam;
1196         pthread_attr_getschedpolicy;
1197         pthread_attr_getscope;
1198         pthread_attr_getstackaddr;
1199         pthread_attr_getstacksize;
1200         pthread_attr_init;
1201         pthread_attr_setdetachstate;
1202         pthread_attr_setinheritsched;
1203         pthread_attr_setschedparam;
1204         pthread_attr_setschedpolicy;
1205         pthread_attr_setscope;
1206         pthread_attr_setstackaddr;
1207         pthread_attr_setstacksize;
1208         pthread_cancel;
1209         __pthread_cleanup_pop;
1210         __pthread_cleanup_push;
1211         pthread_create;
1212         pthread_detach;
1213         pthread_equal;
1214         pthread_exit;
1215         pthread_getschedparam;
1216         pthread_getspecific;
1217         pthread_join;
1218         pthread_key_create;
1219         pthread_key_delete;
1220         pthread_kill;
1221         pthread_once;
1222         pthread_self;
1223         pthread_setcancelstate;
1224         pthread_setcanceltype;
1225         pthread_setschedparam;
1226         pthread_setspecific;
1227         pthread_sigmask;
1228         pthread_testcancel;
1229         putwc;
1230         putwchar;
1231         putws;
1232         regcmp;
1233         regex;
1234         resolvepath;
1235         _resolvepath;
1236         rwlock_destroy          { FLAGS = NODYNSORT };
1237         _rwlock_destroy         { FLAGS = NODYNSORT };
1238         sema_destroy;
1239         _sema_destroy;
1240         _setjmp;
1241         setusershell;
1242         siginterrupt;
1243         sigstack;
1244         snprintf;
1245         strtows;
1246         sync_instruction_memory;
1247         textdomain;
1248         thr_main;
1249         towctrans;
```



```
1250         towlower;
1251         towupper;
1252         ungetwc;
1253         vsnprintf;
1254         watoll;
1255         wcscat;
1256         wcschr;
1257         wcscmp;
1258         wcscoll;
1259         wcscpy;
1260         wcscspn;
1261         wcsftime;
1262         wcslen;
1263         wcsncat;
1264         wcsncmp;
1265         wcsncpy;
1266         wcspbrk;
1267         wcsrchr;
1268         wcsspn;
1269         wcstod;
1270         wcstok;
1271         wcstol;
1272         wcstoul;
1273         wcswcs;
1274         wcswidth;
1275         wcsxfrm;
1276         wctrans;
1277         wctype;
1278         wcwidth;
1279         wscasecmp;
1280         wscat;
1281         wschr;
1282         wscmp;
1283         wscol;
1284         wscoll;
1285         wscpy;
1286         wscspn;
1287         wsdup;
1288         wslen;
1289         wsncasecmp;
1290         wsncat;
1291         wsncmp;
1292         wsncpy;
1293         wspbrk;
1294         wsprintf;
1295         wsrchr;
1296         wsscanf;
1297         wsspn;
1298         wstod;
1299         wstok;
1300         wstol;
1301         wstoll;
1302         wstostr;
1303         wsxfrm;
1304         __xpg4_putmsg;
1305         __xpg4_putpmsg;

1307 $if lf64
1308         creat64;
1309         _creat64;
1310         fgetpos64;
1311         fopen64;
1312         freopen64;
1313         fseeko64;
1314         fsetpos64;
1315         fstat64;
```

```
1316         _fstat64;
1317         fstatvfs64;
1318         _fstatvfs64;
1319         ftello64;
1320         ftruncate64;
1321         _ftruncate64;
1322         ftw64;
1323         _ftw64;
1324         getdents64;
1325         _getdents64;
1326         getrlimit64;
1327         _getrlimit64;
1328         lockf64;
1329         _lockf64;
1330         lseek64;
1331         _lseek64;
1332         lstat64;
1333         _lstat64;
1334         mkstemp64;
1335         _mkstemp64;
1336         mmap64;
1337         _mmap64;
1338         nftw64;
1339         _nftw64;
1340         open64;
1341         _open64;
1342         __posix_readdir_r;
1343         pread64;
1344         _pread64;
1345         pwrite64;
1346         _pwrite64;
1347         readdir64;
1348         _readdir64;
1349         readdir64_r;
1350         _readdir64_r;
1351         setrlimit64;
1352         _setrlimit64;
1353         s_fcntl;
1354         _s_fcntl                { FLAGS = NODYNSORT };
1355         s_ioctl;
1356         stat64;
1357         _stat64;
1358         statvfs64;
1359         _statvfs64;
1360         tell64;
1361         _tell64;
1362         tmpfile64;
1363         truncate64;
1364         _truncate64;
1365         _xftw64;
1366 $endif

1368 $if _sparc
1369         __flt_rounds;
1370 $endif
1371 } SUNW_0.9;

1373 SYMBOL_VERSION SUNW_0.9 {       # SunOS 5.5 (Solaris 2.5)
1374     protected:
1375         acl;
1376         bcmp;
1377         bcopy;
1378         bzero;
1379         facl;
1380         ftime;
1381         getdtablesize;
```



```
1382         gethostid;
1383         gethostname;
1384         getpagesize;
1385         getpriority;
1386         getrusage;
1387         getwd;
1388         index;
1389         initstate;
1390         killpg;
1391         _nsc_trydoorcall;
1392         pthread_condattr_destroy;
1393         pthread_condattr_getpshared;
1394         pthread_condattr_init;
1395         pthread_condattr_setpshared;
1396         pthread_cond_broadcast;
1397         pthread_cond_destroy;
1398         pthread_cond_init;
1399         pthread_cond_signal;
1400         pthread_cond_timedwait;
1401         pthread_cond_wait;
1402         pthread_mutexattr_destroy;
1403         pthread_mutexattr_getprioceiling;
1404         pthread_mutexattr_getprotocol;
1405         pthread_mutexattr_getpshared;
1406         pthread_mutexattr_init;
1407         pthread_mutexattr_setprioceiling;
1408         pthread_mutexattr_setprotocol;
1409         pthread_mutexattr_setpshared;
1410         pthread_mutex_destroy;
1411         pthread_mutex_getprioceiling;
1412         pthread_mutex_init;
1413         pthread_mutex_lock;
1414         pthread_mutex_setprioceiling;
1415         pthread_mutex_trylock;
1416         pthread_mutex_unlock;
1417         random;
1418         reboot;
1419         re_comp;
1420         re_exec;
1421         rindex;
1422         setbuffer;
1423         sethostname;
1424         setlinebuf;
1425         setpriority;
1426         setregid;
1427         setreuid;
1428         setstate;
1429         srandom;
1430         thr_min_stack;
1431         thr_stksegment;
1432         ualarm;
1433         usleep;
1434         wait3;
1435         wait4;
1436 } SUNW_0.8;

1438 SYMBOL_VERSION SUNW_0.8 {       # SunOS 5.4 (Solaris 2.4)
1439     global:
1440         __xpg4                  { FLAGS = NODIRECT };
1441     protected:
1442         addsev;
1443         cond_broadcast          { FLAGS = NODYNSORT };
1444         cond_destroy            { FLAGS = NODYNSORT };
1445         cond_init;
1446         cond_signal             { FLAGS = NODYNSORT };
1447         cond_timedwait;
```

```
1448         cond_wait;
1449         confstr;
1450         fnmatch;
1451         _getdate_err_addr;
1452         glob;
1453         globfree;
1454         iconv;
1455         iconv_close;
1456         iconv_open;
1457         lfmt;
1458         mutex_destroy           { FLAGS = NODYNSORT };
1459         mutex_init;
1460         mutex_lock              { FLAGS = NODYNSORT };
1461         mutex_trylock           { FLAGS = NODYNSORT };
1462         mutex_unlock            { FLAGS = NODYNSORT };
1463         pfmt;
1464         regcomp;
1465         regerror;
1466         regexec;
1467         regfree;
1468         rwlock_init;
1469         rw_rdlock               { FLAGS = NODYNSORT };
1470         rw_read_held;
1471         rw_tryrdlock            { FLAGS = NODYNSORT };
1472         rw_trywrlock            { FLAGS = NODYNSORT };
1473         rw_unlock               { FLAGS = NODYNSORT };
1474         rw_write_held;
1475         rw_wrlock               { FLAGS = NODYNSORT };
1476         sema_held;
1477         sema_init;
1478         sema_post;
1479         sema_trywait;
1480         sema_wait;
1481         setcat;
1482         sigfpe;
1483         strfmon;
1484         strptime;
1485         thr_continue;
1486         thr_create;
1487         thr_exit;
1488         thr_getconcurrency;
1489         thr_getprio;
1490         thr_getspecific;
1491         thr_join;
1492         thr_keycreate;
1493         thr_kill;
1494         thr_self                { FLAGS = NODYNSORT };
1495         thr_setconcurrency;
1496         thr_setprio;
1497         thr_setspecific;
1498         thr_sigsetmask;
1499         thr_suspend;
1500         thr_yield;
1501         vlfmt;
1502         vpfmt;
1503         wordexp;
1504         wordfree;
1505 } SUNW_0.7;

1507 SYMBOL_VERSION SUNW_0.7 {       # SunOS 5.3 (Solaris 2.3)
1508     global:
1509         altzone;
1510         _ctype;
1511         isnanf                  { TYPE = FUNCTION; FILTER = libm.so.2 };
1512         lone;
1513         lten;
```



```
1514         lzero;
1515         memalign                { FLAGS = NODIRECT };
1516         modff                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1517         nss_default_finders;
1518         _sibuf;
1519         _sobuf;
1520         _sys_buslist;
1521         _sys_cldlist;
1522         _sys_fpelist;
1523         _sys_illlist;
1524         _sys_segvlist;
1525         _sys_siginfolistp;
1526         _sys_siglist;
1527         _sys_siglistn;
1528         _sys_siglistp;
1529         _sys_traplist;
1530         valloc                  { FLAGS = NODIRECT };

1532 $if _ELF32
1533         _bufendtab;
1534         _lastbuf;
1535         sys_errlist;
1536         sys_nerr;
1537         _sys_nsig;
1538 $endif

1540     protected:
1541         a64l;
1542         adjtime;
1543         ascftime;
1544         _assert;
1545         atoll;
1546         brk;
1547         __builtin_alloca;
1548         cftime;
1549         closelog;
1550         csetcol;
1551         csetlen;
1552         ctermid_r;
1553         dbm_close;
1554         dbm_delete;
1555         dbm_fetch;
1556         dbm_firstkey;
1557         dbm_nextkey;
1558         dbm_open;
1559         dbm_store;
1560         decimal_to_double;
1561         decimal_to_extended;
1562         decimal_to_quadruple;
1563         decimal_to_single;
1564         double_to_decimal;
1565         drand48;
1566         econvert;
1567         ecvt;
1568         endnetgrent;
1569         endspent;
1570         endutent;
1571         endutxent;
1572         erand48;
1573         euccol;
1574         euclen;
1575         eucscol;
1576         extended_to_decimal;
1577         fchroot;
1578         fconvert;
1579         fcvt;
```

```
1580        ffs;
1581        fgetspent;
1582        fgetspent_r;
1583        _filbuf;
1584        file_to_decimal;
1585        finite;
1586        _flsbuf;
1587        fork1                   { FLAGS = NODYNSORT };
1588        fpclass;
1589        fpgetmask;
1590        fpgetsticky;
1591        fpsetmask;
1592        fpsetsticky;
1593        fstatfs;
1594        ftruncate;
1595        ftw;
1596        func_to_decimal;
1597        gconvert;
1598        gcvt;
1599        getdents;
1600        gethrtime;
1601        gethrvtime;
1602        getmntany;
1603        getmntent;
1604        getnetgrent;
1605        getnetgrent_r;
1606        getpw;
1607        getspent;
1608        getspent_r;
1609        getspnam;
1610        getspnam_r;
1611        getutent;
1612        getutid;
1613        getutline;
1614        getutmp;
1615        getutmpx;
1616        getutxent;
1617        getutxid;
1618        getutxline;
1619        getvfsany;
1620        getvfsent;
1621        getvfsfile;
1622        getvfsspec;
1623        getwidth;
1624        gsignal;
1625        hasmntopt;
1626        innetgr;
1627        insque;
1628        _insque;
1629        jrand48;
1630        l64a;
1631        ladd;
1632        lckpwdf;
1633        lcong48;
1634        ldivide;
1635        lexp10;
1636        llabs;
1637        lldiv;
1638        llog10;
1639        llseek;
1640        lltostr;
1641        lmul;
1642        lrand48;
1643        lshiftl;
1644        lsub;
1645        _lwp_cond_broadcast;
```



```
1646        _lwp_cond_signal;
1647        _lwp_cond_timedwait;
1648        _lwp_cond_wait;
1649        _lwp_continue;
1650        _lwp_info;
1651        _lwp_kill;
1652        _lwp_mutex_lock;
1653        _lwp_mutex_trylock;
1654        _lwp_mutex_unlock;
1655        _lwp_self;
1656        _lwp_sema_init;
1657        _lwp_sema_post;
1658        _lwp_sema_wait;
1659        _lwp_suspend;
1660        madvise;
1661        __major;
1662        __makedev;
1663        mincore;
1664        __minor;
1665        mkstemp;
1666        _mkstemp;
1667        mlockall;
1668        mrand48;
1669        munlockall;
1670        _mutex_held             { FLAGS = NODYNSORT };
1671        _mutex_lock             { FLAGS = NODYNSORT };
1672        nrand48;
1673        _nss_netdb_aliases;
1674        _nss_XbyY_buf_alloc;
1675        _nss_XbyY_buf_free;
1676        __nsw_extended_action;
1677        __nsw_freeconfig;
1678        __nsw_getconfig;
1679        openlog;
1680        plock;
1681        p_online;
1682        pread;
1683        __priocntl;
1684        __priocntlset;
1685        processor_bind;
1686        processor_info;
1687        psiginfo;
1688        psignal;
1689        putpwent;
1690        putspent;
1691        pututline;
1692        pututxline;
1693        pwrite;
1694        qeconvert;
1695        qecvt;
1696        qfconvert;
1697        qfcvt;
1698        qgconvert;
1699        qgcvt;
1700        quadruple_to_decimal;
1701        realpath;
1702        remque;
1703        _remque;
1704        _rw_read_held;
1705        _rw_write_held;
1706        seconvert;
1707        seed48;
1708        select;
1709        _sema_held;
1710        setegid;
1711        seteuid;
```

```
1712         setlogmask;
1713         setnetgrent;
1714         setspent;
1715         settimeofday;
1716         setutent;
1717         setutxent;
1718         sfconvert;
1719         sgconvert;
1720         sig2str;
1721         sigwait;
1722         single_to_decimal;
1723         srand48;
1724         ssignal;
1725         statfs;
1726         str2sig;
1727         strcasecmp;
1728         string_to_decimal;
1729         strncasecmp;
1730         strsignal;
1731         strtoll;
1732         strtoull;
1733         swapctl;
1734         _syscall;
1735         sysfs;
1736         syslog;
1737         _syslog;
1738         tmpnam_r;
1739         truncate;
1740         ttyslot;
1741         uadmin;
1742         ulckpwdf;
1743         ulltostr;
1744         unordered;
1745         updwtmp;
1746         updwtmpx;
1747         ustat;
1748         utimes;
1749         utmpname;
1750         utmpxname;
1751         vfork;
1752         vhangup;
1753         vsyslog;
1754         yield;

1756 $if i386
1757         # Note: _syscall is also defined above. Here, we add the NODYNSORT
1758         # attribute to it. On this platform, it is an alias to syscall.
1759         # If that is changed, this lines should be removed.
1760         _syscall                { FLAGS = NODYNSORT };
1761 $endif

1763 # The 32-bit sparc ABI requires SISCD_2.3. On other platforms, those symbols
1764 # go directly into SUNW_0.7.
1765 $if sparc32
1766 } SISCD_2.3;

1768 SYMBOL_VERSION SISCD_2.3 {
1769 $endif

1771     global:
1772         errno                   { FLAGS = NODIRECT };
1773         _iob;

1775     protected:
1776         addseverity;
1777         _addseverity;
```



```
1778         asctime_r;
1779         crypt;
1780         _crypt;
1781         ctime_r;
1782         encrypt;
1783         _encrypt;
1784         endgrent;
1785         endpwent;
1786         ___errno;
1787         fgetgrent;
1788         fgetgrent_r;
1789         fgetpwent;
1790         fgetpwent_r;
1791         flockfile;
1792         funlockfile;
1793         getchar_unlocked;
1794         getc_unlocked;
1795         getgrent;
1796         getgrent_r;
1797         getgrgid_r;
1798         getgrnam_r;
1799         getitimer;
1800         _getitimer;
1801         getlogin_r;
1802         getpwent;
1803         getpwent_r;
1804         getpwnam_r;
1805         getpwuid_r;
1806         gettimeofday;
1807         _gettimeofday;
1808         gmtime_r;
1809         localtime_r;
1810         putchar_unlocked;
1811         putc_unlocked;
1812         rand_r;
1813         readdir_r;
1814         setgrent;
1815         setitimer;
1816         _setitimer;
1817         setkey;
1818         _setkey;
1819         setpwent;
1820         strtok_r;
1821         sysinfo;
1822         _sysinfo;
1823         ttyname_r;

1825 $if _ELF32
1826         __div64;
1827         __mul64;
1828         __rem64;
1829         __udiv64;
1830         __urem64;
1831 $endif

1833 $if sparc32
1834         __dtoll;
1835         __dtoull;
1836         __ftoll;
1837         __ftoull;
1838         _Q_lltoq;
1839         _Q_qtoll;
1840         _Q_qtoull;
1841         _Q_ulltoq;
1842         sbrk;
1843         _sbrk;
```

```
1844         __umul64                { FLAGS = NODYNSORT };  # Same address as __mul6
1845 $endif

1847 # On 32-bit platforms, the following symbols go into SYSVABI_1.3, but on
1848 # other platforms they go directly into the current version (which will be
1849 # either SUNW_0.7, or SISCD_2.3, depending on the similar issue described above.
1850 $if _ELF32
1851 } SYSVABI_1.3;

1853 SYMBOL_VERSION SYSVABI_1.3 {
1854 $endif

1856     global:
1857         _altzone;
1858         calloc                  { FLAGS = NODIRECT };
1859         __ctype;
1860         daylight;
1861         _daylight;
1862         environ                 { FLAGS = NODIRECT };
1863         _environ                { FLAGS = NODIRECT };
1864         free                    { FLAGS = NODIRECT };
1865         frexp                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1866         getdate_err;
1867         _getdate_err;
1868         getenv;
1869         __huge_val;
1870         __iob;
1871         isnan                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1872         _isnan                  { TYPE = FUNCTION; FILTER = libm.so.2 };
1873         isnand                  { TYPE = FUNCTION; FILTER = libm.so.2 };
1874         _isnand                 { TYPE = FUNCTION; FILTER = libm.so.2 };
1875         ldexp                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1876         logb                    { TYPE = FUNCTION; FILTER = libm.so.2 };
1877         malloc                  { FLAGS = NODIRECT };
1878         memcmp;
1879         memcpy;
1880         memmove;
1881         memset;
1882         modf                    { TYPE = FUNCTION; FILTER = libm.so.2 };
1883         _modf                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1884         nextafter               { TYPE = FUNCTION; FILTER = libm.so.2 };
1885         _nextafter              { TYPE = FUNCTION; FILTER = libm.so.2 };
1886         _numeric;
1887         optarg;
1888         opterr;
1889         optind;
1890         optopt;
1891         realloc                 { FLAGS = NODIRECT };
1892         scalb                   { TYPE = FUNCTION; FILTER = libm.so.2 };
1893         _scalb                  { TYPE = FUNCTION; FILTER = libm.so.2 };
1894         timezone;
1895         _timezone;
1896         tzname;
1897         _tzname;
1898 $if i386
1899         _fp_hw;
1900 $endif

1902     protected:
1903         abort;
1904         abs;
1905         access;
1906         _access;
1907         acct;
1908         _acct;
1909         alarm;
```



```
1910         _alarm;
1911         asctime;
1912         __assert;
1913         atexit;
1914         atof;
1915         atoi;
1916         atol;
1917         bsearch;
1918         catclose;
1919         _catclose;
1920         catgets;
1921         _catgets;
1922         catopen;
1923         _catopen;
1924         cfgetispeed;
1925         _cfgetispeed;
1926         cfgetospeed;
1927         _cfgetospeed;
1928         cfsetispeed;
1929         _cfsetispeed;
1930         cfsetospeed;
1931         _cfsetospeed;
1932         chdir;
1933         _chdir;
1934         chmod;
1935         _chmod;
1936         chown;
1937         _chown;
1938         chroot;
1939         _chroot;
1940         _cleanup;
1941         clearerr;
1942         clock;
1943         _close;
1944         close;
1945         closedir;
1946         _closedir;
1947         creat;
1948         _creat;
1949         ctermid;
1950         ctime;
1951         cuserid;
1952         _cuserid;
1953         difftime;
1954         div;
1955         dup;
1956         _dup;
1957         dup2;
1958         _dup2;
1959         execl;
1960         _execl;
1961         execle;
1962         _execle;
1963         execlp;
1964         _execlp;
1965         execv;
1966         _execv;
1967         execve;
1968         _execve;
1969         execvp;
1970         _execvp;
1971         exit;
1972         _exit;
1973         fattach;
1974         _fattach;
1975         fchdir;
```

```
1976         _fchdir;
1977         fchmod;
1978         _fchmod;
1979         fchown;
1980         _fchown;
1981         fclose;
1982         fcntl;
1983         _fcntl;
1984         fdetach;
1985         _fdetach;
1986         fdopen;
1987         _fdopen;
1988         feof;
1989         ferror;
1990         fflush;
1991         fgetc;
1992         fgetpos;
1993         fgets;
1994         __filbuf;
1995         fileno;
1996         _fileno;
1997         __flsbuf;
1998         fmtmsg;
1999         _fmtmsg;
2000         fopen;
2001         _fork;
2002         fork;
2003         fpathconf;
2004         _fpathconf;
2005         fprintf;
2006         fputc;
2007         fputs;
2008         fread;
2009         freopen;
2010         fscanf;
2011         fseek;
2012         fsetpos;
2013         fstat;
2014         _fstat;
2015         fstatvfs;
2016         _fstatvfs;
2017         fsync;
2018         _fsync;
2019         ftell;
2020         ftok;
2021         _ftok;
2022         fwrite;
2023         getc;
2024         getchar;
2025         getcontext;
2026         _getcontext;
2027         getcwd;
2028         _getcwd;
2029         getdate;
2030         _getdate;
2031         getegid;
2032         _getegid;
2033         geteuid;
2034         _geteuid;
2035         getgid;
2036         _getgid;
2037         getgrgid;
2038         getgrnam;
2039         getgroups;
2040         _getgroups;
2041         getlogin;
```



```
2042         getmsg;
2043         _getmsg;
2044         getopt;
2045         _getopt;
2046         getpass;
2047         _getpass;
2048         getpgid;
2049         _getpgid;
2050         getpgrp;
2051         _getpgrp;
2052         getpid;
2053         _getpid;
2054         getpmsg;
2055         _getpmsg;
2056         getppid;
2057         _getppid;
2058         getpwnam;
2059         getpwuid;
2060         getrlimit;
2061         _getrlimit;
2062         gets;
2063         getsid;
2064         _getsid;
2065         getsubopt;
2066         _getsubopt;
2067         gettxt;
2068         _gettxt;
2069         getuid;
2070         _getuid;
2071         getw;
2072         _getw;
2073         gmtime;
2074         grantpt;
2075         _grantpt;
2076         hcreate;
2077         _hcreate;
2078         hdestroy;
2079         _hdestroy;
2080         hsearch;
2081         _hsearch;
2082         initgroups;
2083         _initgroups;
2084         ioctl;
2085         _ioctl;
2086         isalnum;
2087         isalpha;
2088         isascii;
2089         _isascii;
2090         isastream;
2091         _isastream;
2092         isatty;
2093         _isatty;
2094         iscntrl;
2095         isdigit;
2096         isgraph;
2097         islower;
2098         isprint;
2099         ispunct;
2100         isspace;
2101         isupper;
2102         isxdigit;
2103         kill;
2104         _kill;
2105         labs;
2106         lchown;
2107         _lchown;
```

```
        ldiv;
        lfind;
        _lfind;
        link;
        _link;
        localeconv;
        localtime;
        lockf;
        _lockf;
        longjmp;
        lsearch;
        _lsearch;
        lseek;
        _lseek;
        lstat;
        _lstat;
        makecontext;
        _makecontext;
        mblen;
        mbstowcs;
        mbtowc;
        memccpy;
        _memccpy;
        memchr;
        memcntl;
        _memcntl;
        mkdir;
        _mkdir;
        mkfifo;
        _mkfifo;
        mknod;
        _mknod;
        mktemp;
        _mktemp;
        mktime;
        mlock;
        _mlock;
        mmap;
        _mmap;
        monitor;
        _monitor;
        mount;
        _mount;
        mprotect;
        _mprotect;
        msgctl;
        _msgctl;
        msgget;
        _msgget;
        msgrcv;
        _msgrcv;
        msgsnd;
        _msgsnd;
        msync;
        _msync;
        munlock;
        _munlock;
        munmap;
        _munmap;
        nftw;
        _nftw;
        nice;
        _nice;
        nl_langinfo;
        _nl_langinfo;
        open;
```



```
        _open;
        opendir;
        _opendir;
        pathconf;
        _pathconf;
        pause;
        _pause;
        pclose;
        _pclose;
        perror;
        pipe;
        _pipe;
        poll;
        _poll;
        popen;
        _popen;
        printf;
        profil;
        _profil;
        ptsname;
        _ptsname;
        putc;
        putchar;
        putenv;
        _putenv;
        putmsg;
        _putmsg;
        putpmsg;
        _putpmsg;
        puts;
        putw;
        _putw;
        qsort;
        raise;
        rand;
        read;
        _read;
        readdir;
        _readdir;
        readlink;
        _readlink;
        readv;
        _readv;
        remove;
        rename;
        _rename;
        rewind;
        rewinddir;
        _rewinddir;
        rmdir;
        _rmdir;
        scanf;
        seekdir;
        _seekdir;
        semctl;
        _semctl;
        semget;
        _semget;
        semop;
        _semop;
        setbuf;
        setcontext;
        _setcontext             { FLAGS = NODYNSORT };
        setgid;
        _setgid;
        setgroups;
```

```
2240         _setgroups;
2241         setjmp;
2242         setlabel;
2243         setlocale;
2244         setpgid;
2245         _setpgid;
2246         setpgrp;
2247         _setpgrp;
2248         setrlimit;
2249         _setrlimit;
2250         setsid;
2251         _setsid;
2252         setuid;
2253         _setuid;
2254         setvbuf;
2255         shmat;
2256         _shmat;
2257         shmctl;
2258         _shmctl;
2259         shmdt;
2260         _shmdt;
2261         shmget;
2262         _shmget;
2263         sigaction;
2264         _sigaction              { FLAGS = NODYNSORT };
2265         sigaddset;
2266         _sigaddset;
2267         sigaltstack;
2268         _sigaltstack;
2269         sigdelset;
2270         _sigdelset;
2271         sigemptyset;
2272         _sigemptyset;
2273         sigfillset;
2274         _sigfillset;
2275         sighold;
2276         _sighold;
2277         sigignore;
2278         _sigignore;
2279         sigismember;
2280         _sigismember;
2281         siglongjmp;
2282         _siglongjmp;
2283         signal;
2284         sigpause;
2285         _sigpause;
2286         sigpending;
2287         _sigpending;
2288         sigprocmask;
2289         _sigprocmask;
2290         sigrelse;
2291         _sigrelse;
2292         sigsend;
2293         _sigsend;
2294         sigsendset;
2295         _sigsendset;
2296         sigset;
2297         _sigset;
2298         sigsetjmp;
2299         _sigsetjmp              { FLAGS = NODYNSORT };
2300         sigsuspend;
2301         _sigsuspend;
2302         sleep;
2303         _sleep;
2304         sprintf;
2305         srand;
```



```
2306         sscanf;
2307         stat;
2308         _stat;
2309         statvfs;
2310         _statvfs;
2311         stime;
2312         _stime;
2313         strcat;
2314         strchr;
2315         strcmp;
2316         strcoll;
2317         strcpy;
2318         strcspn;
2319         strdup;
2320         _strdup;
2321         strerror;
2322         strftime;
2323         strlen;
2324         strncat;
2325         strncmp;
2326         strncpy;
2327         strpbrk;
2328         strrchr;
2329         strspn;
2330         strstr;
2331         strtod;
2332         strtok;
2333         strtol;
2334         strtoul;
2335         strxfrm;
2336         swab;
2337         _swab;
2338         swapcontext;
2339         _swapcontext;
2340         symlink;
2341         _symlink;
2342         sync;
2343         _sync;
2344         sysconf;
2345         _sysconf;
2346         system;
2347         tcdrain;
2348         _tcdrain;
2349         tcflow;
2350         _tcflow;
2351         tcflush;
2352         _tcflush;
2353         tcgetattr;
2354         _tcgetattr;
2355         tcgetpgrp;
2356         _tcgetpgrp;
2357         tcgetsid;
2358         _tcgetsid;
2359         tcsendbreak;
2360         _tcsendbreak;
2361         tcsetattr;
2362         _tcsetattr;
2363         tcsetpgrp;
2364         _tcsetpgrp;
2365         tdelete;
2366         _tdelete;
2367         tell;
2368         _tell;
2369         telldir;
2370         _telldir;
2371         tempnam;
```

```
2372         _tempnam;
2373         tfind;
2374         _tfind;
2375         time;
2376         _time;
2377         times;
2378         _times;
2379         tmpfile;
2380         tmpnam;
2381         toascii;
2382         _toascii;
2383         tolower;
2384         _tolower;
2385         toupper;
2386         _toupper;
2387         tsearch;
2388         _tsearch;
2389         ttyname;
2390         twalk;
2391         _twalk;
2392         tzset;
2393         _tzset;
2394         ulimit;
2395         _ulimit;
2396         umask;
2397         _umask;
2398         umount;
2399         _umount;
2400         uname;
2401         _uname;
2402         ungetc;
2403         unlink;
2404         _unlink;
2405         unlockpt;
2406         _unlockpt;
2407         utime;
2408         _utime;
2409         vfprintf;
2410         vprintf;
2411         vsprintf;
2412         wait;
2413         _wait;
2414         waitid;
2415         _waitid;
2416         waitpid;
2417         _waitpid;
2418         wcstombs;
2419         wctomb;
2420         write;
2421         _write;
2422         writev;
2423         _writev;
2424         _xftw;

2426 $if _ELF32
2427         ptrace;
2428         _ptrace;
2429 $endif

2431 $if i386
2432         _fxstat;
2433         _lxstat;
2434         nuname;
2435         _nuname;
2436         _xmknod;
2437         _xstat;
```



```
2438 $endif

2440 $if !sparc32
2441         sbrk;
2442 $endif

2444 $if _sparc
2445         __dtou;
2446         __ftou;
2447 $endif

2449 $if sparc32
2450         .div;
2451         .mul;
2452         .rem;
2453         .stret1;
2454         .stret2;
2455         .stret4;
2456         # .stret4 and .stret8 are the same thing
2457         .stret8                 { FLAGS = NODYNSORT };
2458         .udiv;
2459         .umul;
2460         .urem;
2461         _Q_add;
2462         _Q_cmp;
2463         _Q_cmpe;
2464         _Q_div;
2465         _Q_dtoq;
2466         _Q_feq;
2467         _Q_fge;
2468         _Q_fgt;
2469         _Q_fle;
2470         _Q_flt;
2471         _Q_fne;
2472         _Q_itoq;
2473         _Q_mul;
2474         _Q_neg;
2475         _Q_qtod;
2476         _Q_qtoi;
2477         _Q_qtos;
2478         _Q_qtou;
2479         _Q_sqrt;
2480         _Q_stoq;
2481         _Q_sub;
2482         _Q_utoq;
2483 $endif

2485 $if sparcv9
2486         # __align_cpy_1 is an alias for memcpy. Filter it out of
2487         # the .SUNW_dynsymsort section
2488         __align_cpy_1           { FLAGS = NODYNSORT };
2489         __align_cpy_16;
2490         __align_cpy_2;
2491         __align_cpy_4;
2492         # __align_cpy_8 is same as __align_cpy_16
2493         __align_cpy_8           { FLAGS = NODYNSORT };
2494         __dtoul;
2495         __ftoul;
2496         _Qp_add;
2497         _Qp_cmp;
2498         _Qp_cmpe;
2499         _Qp_div;
2500         _Qp_dtoq;
2501         _Qp_feq;
2502         _Qp_fge;
2503         _Qp_fgt;
```

```
        _Qp_fle;
        _Qp_flt;
        _Qp_fne;
        _Qp_itoq;
        _Qp_mul;
        _Qp_neg;
        _Qp_qtod;
        _Qp_qtoi;
        _Qp_qtos;
        _Qp_qtoui;
        _Qp_qtoux;
        _Qp_qtox;
        _Qp_sqrt;
        _Qp_stoq;
        _Qp_sub;
        _Qp_uitoq;
        _Qp_uxtoq;
        _Qp_xtoq;
        __sparc_utrap_install;
$endif

# On amd64, we also have SYSVABI_1.3, but it contains a small subset of
# the symbols put in that version on other platforms.
$if amd64
} SYSVABI_1.3;

SYMBOL_VERSION SYSVABI_1.3 {
$endif
    global:
$if !_sparc
        __flt_rounds;
$endif

    protected:
        _ctermid;
        _getgrgid;
        _getgrnam;
        _getlogin;
        _getpwnam;
        _getpwuid;
        _ttyname;

$if !sparc32
        _sbrk;
$endif

$if _x86
        _fpstart;
        __fpstart;
$endif
};



# There should never be more than one SUNWprivate version.
# Don't add any more.  Add new private symbols to SUNWprivate_1.1

SYMBOL_VERSION SUNWprivate_1.1 {
    global:
        ___Argv                 { FLAGS = NODIRECT };
        cfree                   { FLAGS = NODIRECT };
        _cswidth;
        __ctype_mask;
        __environ_lock          { FLAGS = NODIRECT };
        __inf_read;
        __inf_written;
```



```
        __i_size;
        _isnanf                 { TYPE = FUNCTION; FILTER = libm.so.2 };
        __iswrune;
        __libc_threaded;
        _lib_version            { FLAGS = NODIRECT };
        _logb                   { TYPE = FUNCTION; FILTER = libm.so.2 };
        _lone                   { FLAGS = NODYNSORT };
        _lten                   { FLAGS = NODYNSORT };
        _lzero                  { FLAGS = NODYNSORT };
        __malloc_lock;
        _memcmp;
        _memcpy                 { FLAGS = NODYNSORT };
        _memmove;
        _memset;
        _modff                  { TYPE = FUNCTION; FILTER = libm.so.2 };
        __nan_read;
        __nan_written;
        __nextwctype;
        __nis_debug_bind;
        __nis_debug_calls;
        __nis_debug_file;
        __nis_debug_rpc;
        __nis_prefsrv;
        __nis_preftype;
        __nis_server;
        _nss_default_finders;
        __progname              { FLAGS = NODIRECT };
        _smbuf;
        _sp;
        __strdupa_str           { FLAGS = NODIRECT };
        __strdupa_len           { FLAGS = NODIRECT };
        _tdb_bootstrap;
        __threaded;
        thr_probe_getfunc_addr;
        __trans_lower;
        __trans_upper;
        _uberdata;
        __xpg6                  { FLAGS = NODIRECT };

$if _ELF32
        _dladdr                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dladdr1                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlclose                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dldump                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlerror                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlinfo                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlmopen                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlopen                 { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _dlsym                  { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _ld_libc                { TYPE = FUNCTION; FILTER = /usr/lib/ld.so.1 };
        _sys_errlist;
        _sys_errs;
        _sys_index;
        _sys_nerr               { FLAGS = NODYNSORT };
        _sys_num_err;
$elif sparcv9
        _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
        _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/sparcv9/ld.so.1 };
```

```
$elif amd64
        _dladdr         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dladdr1        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlamd64getunwind { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlclose        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dldump         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlerror        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlinfo         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlmopen        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlopen         { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _dlsym          { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
        _ld_libc        { TYPE = FUNCTION; FILTER = /usr/lib/amd64/ld.so.1 };
$else
$error unknown platform
$endif

$if _sparc
        __lyday_to_month;
        __mon_lengths;
        __yday_to_month;
$endif
$if i386
        _sse_hw;
$endif

    protected:
        acctctl;
        allocids;
        _assert_c99;
        __assert_c99;
        _assfail;
        attr_count;
        attr_to_data_type;
        attr_to_name;
        attr_to_option;
        attr_to_xattr_view;
        _autofssys;
        _bufsync;
        _cladm;
        __class_quadruple;
        core_get_default_content;
        core_get_default_path;
        core_get_global_content;
        core_get_global_path;
        core_get_options;
        core_get_process_content;
        core_get_process_path;
        core_set_default_content;
        core_set_default_path;
        core_set_global_content;
        core_set_global_path;
        core_set_options;
        core_set_process_content;
        core_set_process_path;
        dbm_close_status;
        dbm_do_nextkey;
        dbm_setdefwrite;
        _D_cplx_div;
        _D_cplx_div_ix;
        _D_cplx_div_rx;
        _D_cplx_mul;
        defclose_r;
        defcntl;
        defcntl_r;
        defopen;
        defopen_r;
```



```
        defread;
        defread_r;
        _delete;
        _dgettext;
        _doprnt;
        _doscan;
        _errfp;
        _errxfp;
        exportfs;
        _F_cplx_div;
        _F_cplx_div_ix;
        _F_cplx_div_rx;
        _F_cplx_mul;
        __fgetwc_xpg5;
        __fgetws_xpg5;
        _findbuf;
        _findiop;
        __fini_daemon_priv;
        _finite;
        _fork1                  { FLAGS = NODYNSORT };
        _forkall                { FLAGS = NODYNSORT };
        _fpclass;
        _fpgetmask;
        _fpgetround;
        _fpgetsticky;
        _fprintf;
        _fpsetmask;
        _fpsetround;
        _fpsetsticky;
        __fputwc_xpg5;
        __fputws_xpg5;
        _ftw;
        _gcvt;
        _getarg;
        __getcontext;
        _getdents;
        _get_exit_frame_monitor;
        _getfp;
        _getgroupsbymember;
        _getlogin_r;
        _getsp;
        __gettsp;
        getvmusage;
        __getwchar_xpg5;
        __getwc_xpg5;
        gtty;
        __idmap_flush_kcache;
        __idmap_reg;
        __idmap_unreg;
        __init_daemon_priv;
        __init_suid_priv;
        _insert;
        inst_sync;
        _iswctype;
        klpd_create;
        klpd_getpath;
        klpd_getport;
        klpd_getucred;
        klpd_register;
        klpd_register_id;
        klpd_unregister;
        klpd_unregister_id;
        _lgrp_home_fast         { FLAGS = NODYNSORT };
        _lgrpsys;
        _lltostr;
        _lock_clear;
```

```
2768        _lock_try;
2769        _ltzset;
2770        lwp_self;
2771        makeut;
2772        makeutx;
2773        _mbftowc;
2774        mcfiller;
2775        mntopt;
2776        modctl;
2777        modutx;
2778        msgctl64;
2779        __multi_innetgr;
2780        _mutex_destroy          { FLAGS = NODYNSORT };
2781        mutex_held;
2782        _mutex_init             { FLAGS = NODYNSORT };
2783        _mutex_unlock           { FLAGS = NODYNSORT };
2784        name_to_attr;
2785        nfs_getfh;
2786        nfssvc;
2787        _nfssys;
2788        __nis_get_environment;
2789        _nss_db_state_destr;
2790        nss_default_key2str;
2791        nss_delete;
2792        nss_endent;
2793        nss_getent;
2794        _nss_initf_group;
2795        _nss_initf_netgroup;
2796        _nss_initf_passwd;
2797        _nss_initf_shadow;
2798        nss_packed_arg_init;
2799        nss_packed_context_init;
2800        nss_packed_getkey;
2801        nss_packed_set_status;
2802        nss_search;
2803        nss_setent;
2804        _nss_XbyY_fgets;
2805        __nsw_extended_action_v1;
2806        __nsw_freeconfig_v1;
2807        __nsw_getconfig_v1;
2808        __nthreads;
2809        __openattrdirat;
2810        option_to_attr;
2811        __priv_bracket;
2812        __priv_relinquish;
2813        pset_assign_forced;
2814        pset_bind_lwp;
2815        _psignal;
2816        _pthread_setcleanupinit;
2817        __putwchar_xpg5;
2818        __putwc_xpg5;
2819        rctlctl;
2820        rctllist;
2821        _realbufend;
2822        _resume;
2823        _resume_ret;
2824        _rpcsys;
2825        _sbrk_grow_aligned;
2826        scrwidth;
2827        semctl64;
2828        _semctl64;
2829        set_setcontext_enforcement;
2830        _setbufend;
2831        __set_errno;
2832        setprojrctl;
2833        _setregid;
```



```
2834        _setreuid;
2835        setsigacthandler;
2836        shmctl64;
2837        _shmctl64;
2838        sigflag;
2839        _signal;
2840        _sigoff;
2841        _sigon;
2842        _so_accept;
2843        _so_bind;
2844        _sockconfig;
2845        _so_connect;
2846        _so_getpeername;
2847        _so_getsockname;
2848        _so_getsockopt;
2849        _so_listen;
2850        _so_recv;
2851        _so_recvfrom;
2852        _so_recvmsg;
2853        _so_send;
2854        _so_sendmsg;
2855        _so_sendto;
2856        _so_setsockopt;
2857        _so_shutdown;
2858        _so_socket;
2859        _so_socketpair;
2860        str2group;
2861        str2passwd;
2862        str2spwd;
2863        __strptime_dontzero;
2864        stty;
2865        syscall;
2866        _sysconfig;
2867        __systemcall;
2868        thr_continue_allmutators;
2869        _thr_continue_allmutators;
2870        thr_continue_mutator;
2871        _thr_continue_mutator;
2872        thr_getstate;
2873        _thr_getstate;
2874        thr_mutators_barrier;
2875        _thr_mutators_barrier;
2876        thr_probe_setup;
2877        _thr_schedctl;
2878        thr_setmutator;
2879        _thr_setmutator;
2880        thr_setstate;
2881        _thr_setstate;
2882        thr_sighndlrinfo;
2883        _thr_sighndlrinfo;
2884        _thr_slot_offset;
2885        thr_suspend_allmutators;
2886        _thr_suspend_allmutators;
2887        thr_suspend_mutator;
2888        _thr_suspend_mutator;
2889        thr_wait_mutator;
2890        _thr_wait_mutator;
2891        __tls_get_addr;
2892        tpool_create;
2893        tpool_dispatch;
2894        tpool_destroy;
2895        tpool_wait;
2896        tpool_suspend;
2897        tpool_suspended;
2898        tpool_resume;
2899        tpool_member;
```

```
2900         _ttyname_dev;
2901         _ucred_alloc;
2902         ucred_getamask;
2903         _ucred_getamask;
2904         ucred_getasid;
2905         _ucred_getasid;
2906         ucred_getatid;
2907         _ucred_getatid;
2908         ucred_getauid;
2909         _ucred_getauid;
2910         _ulltostr;
2911         _uncached_getgrgid_r;
2912         _uncached_getgrnam_r;
2913         _uncached_getpwnam_r;
2914         _uncached_getpwuid_r;
2915         __ungetwc_xpg5;
2916         _unordered;
2917         utssys;
2918         _verrfp;
2919         _verrxfp;
2920         _vwarnfp;
2921         _vwarnxfp;
2922         _warnfp;
2923         _warnxfp;
2924         __wcsftime_xpg5;
2925         __wcstok_xpg5;
2926         wdbindf;
2927         wdchkind;
2928         wddelim;
2929         _wrtchk;
2930         _xflsbuf;
2931         _xgetwidth;
2932         zone_add_datalink;
2933         zone_boot;
2934         zone_check_datalink;
2935         zone_create;
2936         zone_destroy;
2937         zone_enter;
2938         zone_getattr;
2939         zone_get_id;
2940         zone_list;
2941         zone_list_datalink;
2942         zonept;
2943         zone_remove_datalink;
2944         zone_setattr;
2945         zone_shutdown;
2946         zone_version;

2948 $if _ELF32
2949         __divdi3;
2950         _file_set;
2951         _fprintf_c89;
2952         _fscanf_c89;
2953         _fwprintf_c89;
2954         _fwscanf_c89;
2955         _imaxabs_c89;
2956         _imaxdiv_c89;
2957         __moddi3;
2958         _printf_c89;
2959         _scanf_c89;
2960         _snprintf_c89;
2961         _sprintf_c89;
2962         _sscanf_c89;
2963         _strtoimax_c89;
2964         _strtoumax_c89;
2965         _swprintf_c89;
```



```
2966         _swscanf_c89;
2967         __udivdi3;
2968         __umoddi3;
2969         _vfprintf_c89;
2970         _vfscanf_c89;
2971         _vfwprintf_c89;
2972         _vfwscanf_c89;
2973         _vprintf_c89;
2974         _vscanf_c89;
2975         _vsnprintf_c89;
2976         _vsprintf_c89;
2977         _vsscanf_c89;
2978         _vswprintf_c89;
2979         _vswscanf_c89;
2980         _vwprintf_c89;
2981         _vwscanf_c89;
2982         _wcstoimax_c89;
2983         _wcstoumax_c89;
2984         _wprintf_c89;
2985         _wscanf_c89;
2986 $endif

2988 $if _sparc
2989         _cerror;
2990         install_utrap;
2991         _install_utrap;
2992         nop;
2993         _Q_cplx_div;
2994         _Q_cplx_div_ix;
2995         _Q_cplx_div_rx;
2996         _Q_cplx_lr_div;
2997         _Q_cplx_lr_div_ix;
2998         _Q_cplx_lr_div_rx;
2999         _Q_cplx_lr_mul;
3000         _Q_cplx_mul;
3001         _QgetRD;
3002         _xregs_clrptr;
3003 $endif

3005 $if sparc32
3006         __ashldi3;
3007         __ashrdi3;
3008         _cerror64;
3009         __cmpdi2;
3010         __floatdidf;
3011         __floatdisf;
3012         __floatundidf;
3013         __floatundisf;
3014         __lshrdi3;
3015         __muldi3;
3016         __ucmpdi2;
3017 $endif

3019 $if _x86
3020         _D_cplx_lr_div;
3021         _D_cplx_lr_div_ix;
3022         _D_cplx_lr_div_rx;
3023         _F_cplx_lr_div;
3024         _F_cplx_lr_div_ix;
3025         _F_cplx_lr_div_rx;
3026         __fltrounds;
3027         sysi86;
3028         _sysi86;
3029         _X_cplx_div;
3030         _X_cplx_div_ix;
3031         _X_cplx_div_rx;
```

```
3032         _X_cplx_lr_div;
3033         _X_cplx_lr_div_ix;
3034         _X_cplx_lr_div_rx;
3035         _X_cplx_mul;
3036         __xgetRD;
3037         __xtol;
3038         __xtoll;
3039         __xtoul;
3040         __xtoull;
3041 $endif

3043 $if i386
3044         __divrem64;
3045         ___tls_get_addr;
3046         __udivrem64;
3047 $endif

3049 # The following functions should not be exported from libc,
3050 # but /lib/libm.so.2, some older versions of the Studio
3051 # compiler/debugger components, and some ancient programs
3052 # found in /usr/dist reference them.  When we no longer
3053 # care about these old and broken binary objects, these
3054 # symbols should be deleted.
3055         _brk                                    { FLAGS = NODYNSORT };
3056         _cond_broadcast                         { FLAGS = NODYNSORT };
3057         _cond_init                              { FLAGS = NODYNSORT };
3058         _cond_signal                            { FLAGS = NODYNSORT };
3059         _cond_wait                              { FLAGS = NODYNSORT };
3060         _ecvt                                   { FLAGS = NODYNSORT };
3061         _fcvt                                   { FLAGS = NODYNSORT };
3062         _getc_unlocked                          { FLAGS = NODYNSORT };
3063         _llseek                                 { FLAGS = NODYNSORT };
3064         _pthread_attr_getdetachstate            { FLAGS = NODYNSORT };
3065         _pthread_attr_getinheritsched           { FLAGS = NODYNSORT };
3066         _pthread_attr_getschedparam             { FLAGS = NODYNSORT };
3067         _pthread_attr_getschedpolicy            { FLAGS = NODYNSORT };
3068         _pthread_attr_getscope                  { FLAGS = NODYNSORT };
3069         _pthread_attr_getstackaddr              { FLAGS = NODYNSORT };
3070         _pthread_attr_getstacksize              { FLAGS = NODYNSORT };
3071         _pthread_attr_init                      { FLAGS = NODYNSORT };
3072         _pthread_condattr_getpshared            { FLAGS = NODYNSORT };
3073         _pthread_condattr_init                  { FLAGS = NODYNSORT };
3074         _pthread_cond_init                      { FLAGS = NODYNSORT };
3075         _pthread_create                         { FLAGS = NODYNSORT };
3076         _pthread_getschedparam                  { FLAGS = NODYNSORT };
3077         _pthread_join                           { FLAGS = NODYNSORT };
3078         _pthread_key_create                     { FLAGS = NODYNSORT };
3079         _pthread_mutexattr_getprioceiling       { FLAGS = NODYNSORT };
3080         _pthread_mutexattr_getprotocol          { FLAGS = NODYNSORT };
3081         _pthread_mutexattr_getpshared           { FLAGS = NODYNSORT };
3082         _pthread_mutexattr_init                 { FLAGS = NODYNSORT };
3083         _pthread_mutex_getprioceiling           { FLAGS = NODYNSORT };
3084         _pthread_mutex_init                     { FLAGS = NODYNSORT };
3085         _pthread_sigmask                        { FLAGS = NODYNSORT };
3086         _rwlock_init                            { FLAGS = NODYNSORT };
3087         _rw_rdlock                              { FLAGS = NODYNSORT };
3088         _rw_unlock                              { FLAGS = NODYNSORT };
3089         _rw_wrlock                              { FLAGS = NODYNSORT };
3090         _sbrk_unlocked                          { FLAGS = NODYNSORT };
3091         _select                                 { FLAGS = NODYNSORT };
3092         _sema_init                              { FLAGS = NODYNSORT };
3093         _sema_post                              { FLAGS = NODYNSORT };
3094         _sema_trywait                           { FLAGS = NODYNSORT };
3095         _sema_wait                              { FLAGS = NODYNSORT };
3096         _sysfs                                  { FLAGS = NODYNSORT };
3097         _thr_create                             { FLAGS = NODYNSORT };
```



```
3098         _thr_exit                               { FLAGS = NODYNSORT };
3099         _thr_getprio                            { FLAGS = NODYNSORT };
3100         _thr_getspecific                        { FLAGS = NODYNSORT };
3101         _thr_join                               { FLAGS = NODYNSORT };
3102         _thr_keycreate                          { FLAGS = NODYNSORT };
3103         _thr_kill                               { FLAGS = NODYNSORT };
3104         _thr_main                               { FLAGS = NODYNSORT };
3105         _thr_self                               { FLAGS = NODYNSORT };
3106         _thr_setspecific                        { FLAGS = NODYNSORT };
3107         _thr_sigsetmask                         { FLAGS = NODYNSORT };
3108         _thr_stksegment                         { FLAGS = NODYNSORT };
3109         _ungetc_unlocked                        { FLAGS = NODYNSORT };

3111     local:
3112         __imax_lldiv                            { FLAGS = NODYNSORT };
3113         _ti_thr_self                            { FLAGS = NODYNSORT };
3114         *;

3116 $if lf64
3117         _seekdir64              { FLAGS = NODYNSORT };
3118         _telldir64              { FLAGS = NODYNSORT };
3119 $endif

3121 $if _sparc
3122         __cerror                { FLAGS = NODYNSORT };
3123 $endif

3125 $if sparc32
3126         __cerror64              { FLAGS = NODYNSORT };
3127 $endif

3129 $if sparcv9
3130         __cleanup               { FLAGS = NODYNSORT };
3131 $endif

3133 $if i386
3134         _syscall6               { FLAGS = NODYNSORT };
3135         __systemcall6           { FLAGS = NODYNSORT };
3136 $endif

3138 $if amd64
3139         ___tls_get_addr         { FLAGS = NODYNSORT };
3140 $endif
3141 };
```