```
#ifndef _PROC_IPC_SEM_H    /* wrapper symbol for kernel use */
#define _PROC_IPC_SEM_H    /* subject to change without notice */
...

/*
 * Permission Definitions:
 */
#define    SEM_A        IPC_W    /* alter permission */
#define    SEM_R        IPC_R    /* read permission */

/*
 * Semaphore Operation Flags:
 *    Value must be unique w.r.t. the common IPC definition
 *    for IPC_NOWAIT.
 */
#define    SEM_UNDO    010000    /* set up adjust on exit entry */

/*
 * Semctl Command Definitions:
 */
#define    GETNCNT        3    /* get semncnt */
#define    GETPID        4    /* get sempid */
#define    GETVAL        5    /* get semval */
#define    GETALL        6    /* get all semval's */
#define    GETZCNT        7    /* get semzcnt */
#define    SETVAL        8    /* set semval */
#define    SETALL        9    /* set all semval's */

/*
 * Structure Definitions.
 */

#if !defined(_STYPES)

/*
 * There is one semaphore id data structure (semid_ds) for each set of
 * semaphores in the system.
 */
struct semid_ds {
    struct ipc_perm sem_perm;    /* operation permission struct */
    struct sem    *sem_base;    /* ptr to first semaphore in set */
    ushort_t    sem_nsems;    /* # of semaphores in set */
    time_t        sem_otime;    /* last semop time */
    long        sem_pad1;    /* reserved for time_t expansion */
    time_t        sem_ctime;    /* last change time */
    long        sem_pad2;        /* time_t expansion */
    long        sem_pad3[SEM_PAD];    /* reserve area */
};

#else    /* _STYPES */

/* SVR3 binary compatibility semid_ds structure */
struct semid_ds {
    struct ipc_perm sem_perm;    /* operation permission struct */
    struct sem    *sem_base;    /* ptr to first semaphore in set */
    ushort_t    sem_nsems;    /* # of semaphores in set */
    time_t        sem_otime;    /* last semop time */
    time_t        sem_ctime;    /* last change time */
};

#endif /* _STYPES */
 ...

/*
 * Macro to convert from the address of the embedded ipc_perm structure
 * to the encapsulating ksemid_ds structure.
 */
#define    IPC_TO_SEMDS(ipcp)    ((struct ksemid_ds *)(ipcp))

/* SVR3 structure */
struct o_semid_ds {
    struct o_ipc_perm sem_perm;    /* operation permission struct */
    void        *sem_base;    /* ptr to first semaphore in set */
    ushort_t    sem_nsems;    /* # of semaphores in set */
    time_t        sem_otime;    /* last semop time */
    time_t        sem_ctime;    /* last change time */
};
```

```
#ifndef _LINUX_SEM_H
#define _LINUX_SEM_H

#include <linux/ipc.h>

/* semop flags */
#define SEM_UNDO        0x1000  /* undo the operation on exit */

/* semctl Command Definitions. */
#define GETPID  11       /* get sempid */
#define GETVAL  12       /* get semval */
#define GETALL  13       /* get all semval's */
#define GETNCNT 14       /* get semncnt */
#define GETZCNT 15       /* get semzcnt */
#define SETVAL  16       /* set semval */
#define SETALL  17       /* set all semval's */

/* ipcs ctl cmds */
#define SEM_STAT 18
#define SEM_INFO 19

/* Obsolete, used only for backwards compatibility and libc5 compiles */
struct semid_ds {
    struct ipc_perm    sem_perm;        /* permissions .. see ipc.h */
    __kernel_time_t    sem_otime;        /* last semop time */
    __kernel_time_t    sem_ctime;        /* last change time */
    struct sem    *sem_base;        /* ptr to first semaphore in array */
    struct sem_queue *sem_pending;        /* pending operations to be processed */
    struct sem_queue **sem_pending_last;    /* last pending operation */
    struct sem_undo    *undo;            /* undo requests on this array */
    unsigned short    sem_nsems;        /* no. of semaphores in array */
};

/* Include the definition of semid64_ds */
#include <asm/sembuf.h>

/* semop system calls takes an array of these. */
struct sembuf {
    unsigned short  sem_num;    /* semaphore index in array */
    short        sem_op;        /* semaphore operation */
    short        sem_flg;    /* operation flags */
};

/* arg for semctl system calls. */
union semun {
    int val;            /* value for SETVAL */
    struct semid_ds *buf;        /* buffer for IPC_STAT & IPC_SET */
    unsigned short *array;        /* array for GETALL & SETALL */
    struct seminfo *__buf;        /* buffer for IPC_INFO */
    void *__pad;
};

struct  seminfo {
    int semmap;
    int semmni;
    int semmns;
    int semmnu;
    int semmsl;
    int semopm;
    int semume;
    int semusz;
    int semvmx;
    int semaem;
};

#define SEMMNI  128             /* <= IPCMNI  max # of semaphore identifiers */
#define SEMMSL  250             /* <= 8 000 max num of semaphores per id */
#define SEMMNS  (SEMMNI*SEMMSL) /* <= INT_MAX max # of semaphores in system */
#define SEMOPM  32              /* <= 1 000 max num of ops per semop call */
#define SEMVMX  32767           /* <= 32767 semaphore maximum value */
#define SEMAEM  SEMVMX          /* adjust on exit max value */

/* unused */
#define SEMUME  SEMOPM          /* max num of undo entries per process */
#define SEMMNU  SEMMNS          /* num of undo structures system wide */
#define SEMMAP  SEMMNS          /* # of entries in semaphore map */
#define SEMUSZ  20        /* sizeof struct sem_undo */
```


(For readability, not all matches are indicated.)

```
174: /*
175:  * There is one semaphore structure (sem) for each semaphore in the system.
176:  */
177:
178: struct sem {
179:     ushort_t semval;    /* semaphore value */
180:     pid_t    sempid;        /* pid of last operation */
181:     ushort_t semncnt;    /* # awaiting semval > cval */
182:     ushort_t semzcnt;    /* # awaiting semval = 0 */
183:     sv_t    semn_sv;    /* synch variable for semncnt */
184:     sv_t    semz_sv;    /* synch variable for semzcnt */
185: };
186:
187: /*
188:  * There is one undo structure per process in the system which has
189:  * done a semop(2) with the SEM_UNDO flag set.
190:  * This structure is pointed to by the p_semundo member of the proc
191:  * structure.
192:  */
193: struct sem_undo {
194:     list_t        un_np;        /* list of active sem_undo structures */
195:     short        un_cnt;        /* # active entries in the undo array */
196:     fspin_t        un_mutex;    /* mutex for this sem_undo struct */
197:     struct undo {
198:         short    un_aoe;        /* adjust on exit value */
199:         ushort_t un_num;    /* semaphore # within semaphore set */
200:         int    un_id;        /* semid */
201:     } un_ent[1];            /* undo entries (one minimum) */
202: };
203:
204: /*
205:  * Semaphore information structure.
206:  *    For SVR4.2, the semmap, semmnu, and semusz fields
207:  *    of the seminfo structure are not used.  This is
208:  *    because everything is dynamically allocated.
209:  */
210: struct    seminfo    {
211:     int    semmap;        /* XXX # of entries in semaphore map */
212:     int    semmni;        /* # of semaphore identifiers */
213:     int    semmns;        /* # of semaphores in system */
214:     int    semmnu;        /* XXX # of undo structures in system */
215:     int    semmsl;        /* max # of semaphores per id */
216:     int    semopm;        /* max # of operations per semop call */
217:     int    semume;        /* max # of undo entries per process */
218:     int    semusz;        /* XXX size in bytes of undo structure */
219:     int    semvmx;        /* semaphore maximum value */
220:     int    semaem;        /* adjust on exit max value */
221: };
222:
223: #endif    /* _KERNEL || _KMEMUSER */
224:
225:
226: /*
227:  * User semaphore template for semop(2):
228:  */
229: struct sembuf {
230:     ushort_t sem_num;    /* semaphore # */
231:     short    sem_op;        /* semaphore operation */
232:     short    sem_flg;    /* operation flags */
233: };
234:
235:
236: #ifdef _KERNEL
 ...
261: #endif /* _PROC_IPC_SEM_H */
```

```
 79: #ifdef __KERNEL__
 80:
 81: /* One semaphore structure for each semaphore in the system. */
 82: struct sem {
 83:     int    semval;        /* current value */
 84:     int    sempid;        /* pid of last operation */
 85: };
 86:
 87: /* One sem_array data structure for each set of semaphores in the system. */
 88: struct sem_array {
 89:     struct kern_ipc_perm    sem_perm;    /* permissions .. see ipc.h */
 90:     time_t            sem_otime;    /* last semop time */
 91:     time_t            sem_ctime;    /* last change time */
 92:     struct sem        *sem_base;    /* ptr to first semaphore in array */
 93:     struct sem_queue    *sem_pending;    /* pending operations to be processed */
 94:     struct sem_queue    **sem_pending_last; /* last pending operation */
 95:     struct sem_undo        *undo;        /* undo requests on this array */
 96:     unsigned long        sem_nsems;    /* no. of semaphores in array */
 97: };
 98:
 99: /* One queue for each sleeping process in the system. */
100: struct sem_queue {
101:     struct sem_queue *    next;     /* next entry in the queue */
102:     struct sem_queue **    prev;     /* previous entry in the queue, *(q->prev) == q */
103:     struct task_struct*    sleeper; /* this process */
104:     struct sem_undo *    undo;     /* undo structure */
105:     int                pid;     /* process id of requesting process */
106:     int                status;     /* completion status of operation */
107:     struct sem_array *    sma;     /* semaphore array for operations */
108:     int            id;     /* internal sem id */
109:     struct sembuf *        sops;     /* array of pending operations */
110:     int            nsops;     /* number of operations */
111:     int            alter;     /* operation will alter semaphore */
112: };
113:
114: /* Each task has a list of undo requests. They are executed automatically
115:  * when the process exits.
116:  */
117: struct sem_undo {
118:     struct sem_undo *    proc_next;    /* next entry on this process */
119:     struct sem_undo *    id_next;    /* next entry on this semaphore set */
120:     int            semid;        /* semaphore set identifier */
121:     short *            semadj;        /* array of adjustments, one per semaphore */
122: };
123:
124: asmlinkage long sys_semget (key_t key, int nsems, int semflg);
125: asmlinkage long sys_semop (int semid, struct sembuf *sops, unsigned nsops);
126: asmlinkage long sys_semctl (int semid, int semnum, int cmd, union semun arg);
127: asmlinkage long sys_semtimedop (int semid, struct sembuf *sops,
128:             unsigned nsops, const struct timespec *timeout);
129:
130: #endif /* __KERNEL__ */
131:
132: #endif /* _LINUX_SEM_H */
```


(For readability, not all matches are indicated.)