semid_is

- One semid data structure for each set of semaphores in the system.

linux/sem.h

```
struct semid_ds {
struct ipc_perm sem_perm;/* permissions .. see ipc.h */
time_t sem_otime;        /* last semop time */
time_t sem_ctime;        /* last change time */
struct sem *sem_base;    /* ptr to first semaphore in array */
struct wait_queue *eventn;
struct wait_queue *eventz;
struct sem_undo *undo;   /* undo requests on this array */
ushort sem_nsems;        /* no. of semaphores in array */
};
```

Kernel shm_ds structure

- In the semid_ds structure, there exists a pointer to the base of the semaphore array itself. Each array member is of the sem structure type. It is also defined in linux/sem.h:

```
/* One semaphore structure for each semaphore in the system. */
struct sem {
   short   sempid;          /*pid of last operation*/
   ushort  semval;          /*current value*/
   ushort  semncnt;         /*num procs awaiting increase in semval*/
   ushort  semzcnt;         /* num procs awaiting semval = 0 */
};
```

SYSTEM CALL: semget()

```
SYSTEM CALL: semget();
PROTOTYPE: int semget ( key_t key, int nsems, int semflg );
RETURNS: semaphore set IPC identifier on success
       -1 on error: errno = EACCESS (permission denied)
          EEXIST (set exists, cannot create (IPC_EXCL))
          EIDRM (set is marked for deletion)
          ENOENT (set does not exist, no IPC_CREAT was used)
          ENOMEM (Not enough memory to create new set)
          ENOSPC (Maximum set limit exceeded)
```

Primer:

```
int open_semaphore_set( key_t keyval, int numsems )
{  int     sid;
   if (!numsems) return(-1);
   if((sid = semget( mykey, numsems, IPC_CREAT | 0660 )) == -1)
     {  return(-1);  }
   return(sid);
}
```

SYSTEM CALL: semop()

```
PROTOTYPE:
   int semop ( int semid, struct sembuf *sops, unsigned nsops);
   RETURNS: 0 on success (all operations performed)
            -1 on error: errno
```

The sops argument points to an array of type sembuf. This structure is declared in linux/sem.h as follows:

```
struct sembuf {
        ushort  sem_num;        /* semaphore index in array */
        short   sem_op;         /* semaphore operation */
        short   sem_flg;        /* operation flags */
        };
```

Primer:

```
 struct sembuf sem_lock = { 0, -1, IPC_NOWAIT };

 if((semop(sid, &sem_lock, 1) == -1) perror("semop");
```

SYSTEM CALL: semctl()

- deliver a message to a queue

```
SYSTEM CALL: semctl();
 PROTOTYPE:
  int semctl ( int semid, int semnum, int cmd, union semun arg );
 RETURNS: positive integer on success
  -1 on error: errno = EACCESS (permission denied)
               EFAULT (invalid address pointed to by arg argument)
               EIDRM (semaphore set was removed)
               EINVAL (set doesn't exist, or semid is invalid)
               EPERM (EUID has no privileges for cmd in arg)
               ERANGE (semaphore value out of range)
  NOTES: Performs control operations on a semaphore set
```

SYSTEM CALL: semctl()

- Valid command values are:

IPC_STAT
Retrieves the semid_ds structure for a set, and stores it in the address of the buf argument in the semun union.

IPC_SET
Sets the value of the ipc_perm member of the semid_ds structure for a set. Takes the values from the buf argument of the semun union.

IPC_RMID
Removes the set from the kernel.

GETALL
Used to obtain the values of all semaphores in a set. The integer values are stored in an array of unsigned short integers pointed to by the array member of the union.

GETNCNT
Returns the number of processes currently waiting for resources.

GETPID
Returns the PID of the process which performed the last semop call.

SYSTEM CALL: `semctl()`

- Valid command values are:

GETVAL

Returns the value of a single semaphore within the set.

GETZCNT

Returns the number of processes currently waiting for 100% resource utilization.

SETALL

Sets all semaphore values with a set to the matching values contained in the array member of the union.

SETVAL

Sets the value of an individual semaphore within the set to the val member of the union.

SYSTEM CALL: semctl()

- The arg argument represents an instance of type semun. This particular union is declared in linux/sem.h as follows:

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

val

Used when the SETVAL command is performed. Specifies the value to set the semaphore to.

buf

Used in the IPC_STAT/IPC_SET commands. Represents a copy of the internal semaphore data structure used in the kernel.

array

A pointer used in the GETALL/SETALL commands. Should point to an array of integer values to be used in setting or retrieving all semaphore values in a set.

SYSTEM CALL: semctl()

- PRIMER

```
int get_sem_val( int sid, int semnum )
  { return( semctl(sid, semnum, GETVAL, 0)); }

#define MAX_PRINTERS 5
printer_usage()
  { int x;
    for(x=0; x<MAX_PRINTERS; x++)
      printf("Printer %d: %d\n\r", x, get_sem_val( sid, x ));
  }
```

- PRIMER

```
void init_semaphore( int sid, int semnum, int initval)
{
        union semun semopts;
        semopts.val = initval;
        semctl( sid, semnum, SETVAL, semopts);
}
```