Semaphores and Related System Calls

Apurba Sarkar

IIEST Shibpur

February 18, 2019

What are they actualy?

- Semaphores are system-implemented data structures that are shared between processes. Some features of the semaphore are
  1. Semaphores are used to synchronize operations when processes access a common, limited, and possibly non-shareable resource.
  2. Each time a process wants to obtain the resource, the associated semaphore is tested. A positive, non-zero semaphore value indicates the resource is available.
  3. Semaphore system calls will, by default, cause the invoking process to block if the semaphore value indicates the resource is not available.

Uses of Semaphores

- Guarding Critical Section
  - Semaphores that control access to a single resource, taking the value of 0 (resource is in use) or 1 (resource is available), are often called binary semaphores.
- Precedence Enforcer
  - Semaphores ensuring an ordering of execution among concurrent processes
- Resource Counter
  - Semaphores controlling access to N (multiple) resources, thus assuming a range of non-negative values, are frequently called counting semaphores.

Things to be discussed

- Creating and Accessing Semaphore Sets
  - Semaphores are allocated in sets.
  - They are allocated with a value of 0 (blocked)
- Semaphore Operations
  - wait (P)
  - signal (V)
- Semaphore Control
  - get info on a set
  - set all semaphore values in a set or get all values
  - set one value in a set or get one value
  - remove the set

Structures to be used

- semid_ds

```
struct  semid_ds {
          . . .
          struct  sem    *sem_base; /*pointer to first semaphore in set*/
          ushort_t        sem_nsems; /* number of semaphores in set */
        };
```

structures to be used

- Structure sem

```
struct   sem {
   ushort   semval; /*semaphore value*/
   pid_t    sempid; /*pid of last operation */
   ushort   semcnt;  /* # waiting for increase in semval */
   ushort   semzcnt; /* # waiting for semval == 0 */
};
```

System Call `semget`

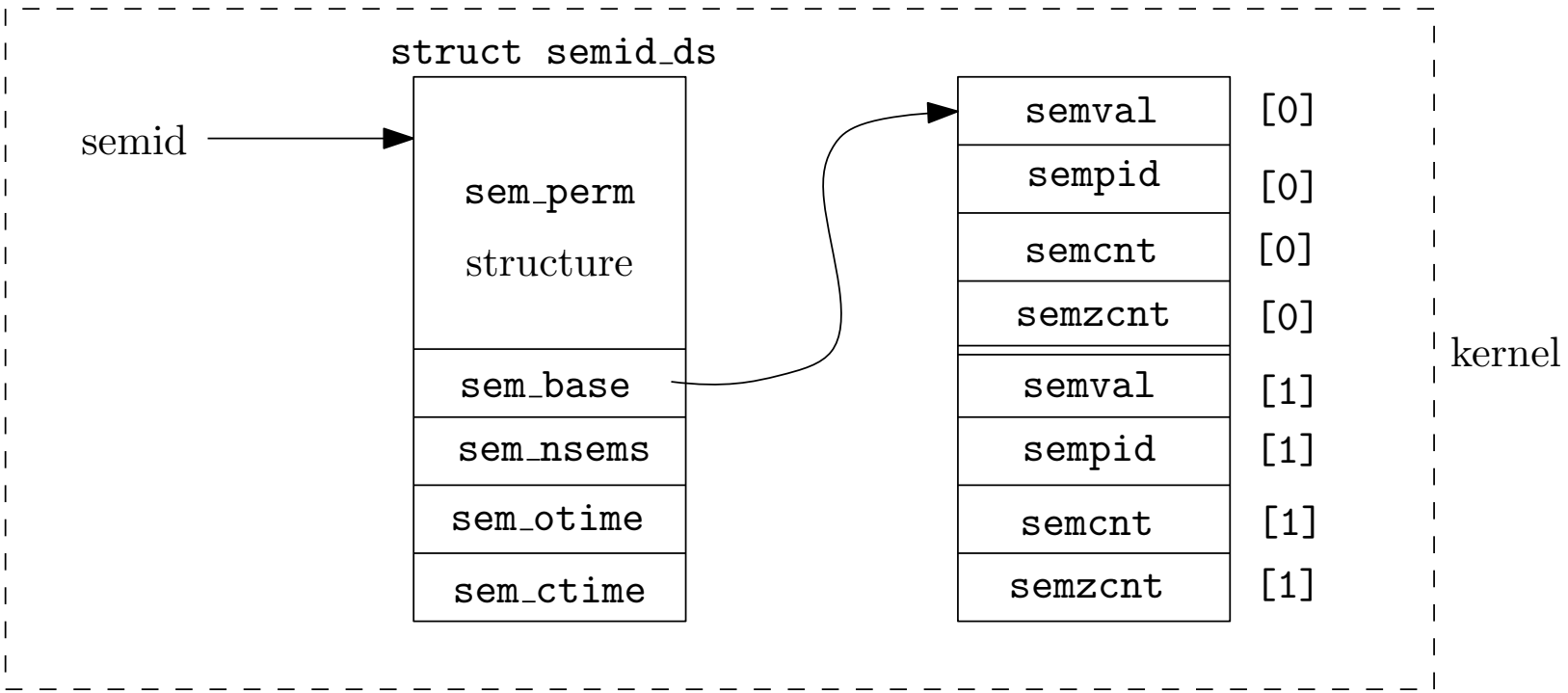
- Function of `semget`
  - To create a semaphore, or gain access to one that exists.
- to include `<sys/types.h> <sys/ipc.h> <sys/sem.h>`
- System Call:`int semget(key_t key, int nsems, int semflg);`
- Return values:
  - On Success: the semaphore identifier `(semid)`
  - On failure: -1 and Sets `errno`
- Arguments
  - `key_t key:` used to identify a semaphore set
  - `int nsems:` the number of semaphores in the set.
  - `int semflg:` specify access permissions and/or special creation condition(s).

semget notes

- If the value for key does not have a semaphore identifier associated with it, and IPC_CREAT has been specified, a new set of semaphores is created (semget).
- If key is IPC_PRIVATE, and IPC_CREAT has been specified, a new set of semaphores is created (semget).
- If IPC_CREAT is specified (but not IPC_EXCL), and the semaphore set for the indicated key value already exists, the (semget) call will return the associated semaphore identifier.
- When using semget to access an established semaphore set, the value of nsems can be set to 0 (a dont-care value).
- If a semaphore identifier exists for this key and IPC_CREAT and IPC_EXCL are both set, (semget) will fail and the "file exists" error message will be returned via errno .

semget

- We can picture a particular semaphore in the kernel as being a `semid_ds` structure that points to an array of `sem` structure.



`semop` Call - Semaphore Operation

- Function of `semop`
  - to perform operations on individual semaphores..
- to include `<sys/types.h> <sys/ipc.h> <sys/sem.h>`
- System Call: `int semop (int semid, struct sembuf *sops, size_t nsops);`
- Return values:
  - On Success: 0
  - On failure: -1 and Sets `errno`
- Arguments
  - `int semid:` semaphore identifier.
  - `struct sembuf *sops:` a reference to the address of an array of semaphore operations that will be performed on the semaphore set denoted by the semid value.
  - `size_t nsops:` the number of elements in the array of semaphore operations.

Semaphore Buffer for `semop` Call

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

- `ushort sem_num:` semaphore number (the index into the array of sem structures referenced by the `sem_base`) .
- `short sem_op:` operation to be performed on the semaphore.
- `short sem_flg:`
  - `IPC_NOWAIT:` if the semaphore operation can not be performed, the call will return immediately.
  - `SEM_UNDO:` allows an operation to be undone if a blocked operation subsequently fails.
  - `0:` process blocks until semaphore available S

Actions Taken by `semop`

If `sem_op:` value:

- **is positive:** Add `sem_op:` to `semval:`. This is a release of a resource.
- **is zero:** The caller will block until the semaphores value becomes zero.
- **is negative:** The caller is blocked until the semaphores value (`semval:`) becomes greater than or equal to the absolute value of `sem_op:`. Then, the absolute value of `sem_op:` is subtracted from`semval:`.

`semctl` System Call - Semaphore Control

- Function
  - To perform a variety of generalized control operations on the system semaphore structure, on the semaphores as a set and on individual semaphores.
- to include `<sys/types.h> <sys/ipc.h> <sys/sem.h>`
- System Call: `int semctl( int semid, int semnum, int cmd, /* union semun arg*/ ...);`
- Return values:
  - On Success: 0
  - On failure: -1 and Sets `errno`
- Arguments
  - `int semid:` a valid semaphore identifier.
  - `semnum:` the number of semaphores in the semaphore set.
  - `int cmd:` an integer command value (`IPC_STAT, IPC_SET, ..`).
  - `arg:` union of type `semun`.

cmd values of semctl Call

- IPC_STAT: return the current values of the semid_ds structure for the indicated semaphore identifier.
- IPC_SET: modify a restricted number of members in the semid_ds structure.
- IPC_RMID: remove the semaphore set.
- GETAll: return the current values of the semaphore set.
- SETALL: initialize all semaphores in a set to the values stored in the array referenced by the fourth arguments to semctl.
- GETVAL: return the current of the individual semaphore referenced by the value of the semnum argument.
- SETVAL: set the value of a single semaphore in a set

union `semun` in the `semctl` call

- A union is is a later-day version of the Pascal variant `record`. It is a data structure that can take on multiple forms.
- `semctl()` requires a union to handle the different kinds of data that can be provided to it or received from it.

```
union  semun {
       int                 val;
       struct              semid_ds  *buf;
       ushort              * array;
    } arg; // declares a semun named arg
```

- The value in arg is one of:
  - `int val`: an integer (0 or others),
  - `struct semid_ds *buf`: a reference to a `semid_ds` structure,
  - `ushort *array`: the base address of an array of short integers (the values for the semaphore(s)).