Boletín 6- Semáforos

Departamento de Lenguajes y Sistemas Informáticos

Indice

1. Introducción: Semáforos
2. Clave: ftok()
3. Creación u obtención de semáforos: semget()
4. Operaciones con semáforos: semop()
5. Operaciones de control: semctl()

Introducción: Semáforos

- **-Definición:**
  Un semáforo es un tipo abstracto de datos que permite sincronizar el acceso de dos o más procesos a un recurso compartido de manera concurrente.

- ***Recordatorio:***
  - mecanismo de sincronización basado en espera **no** ocupada
  - se compone de un contador y una lista de procesos en espera
  - dos primitivas básicas: *down* y *up*

Introducción: Semáforos (2)

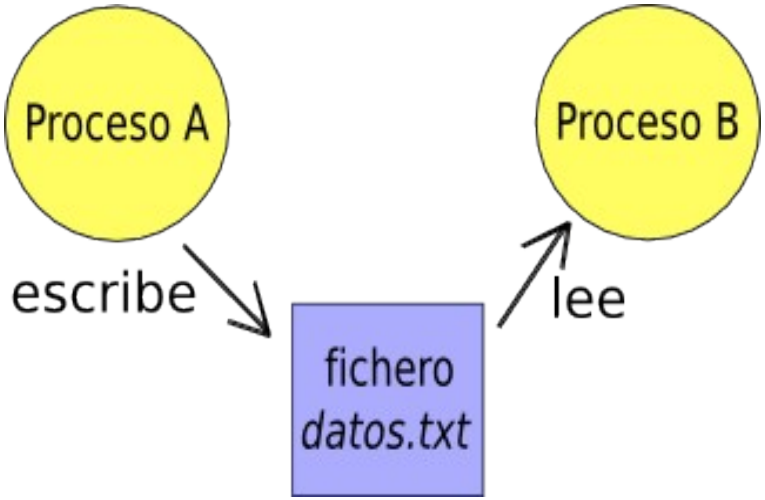
- En UNIX, los semáforos forman parte del conjunto de mecanismo de "***I**nter**P**rocess **C**ommunication" (IPC)* junto a memoria compartida y colas de mensajes.
- ¿Cómo obtener la lista de IPCs activos en el sistema?
  $ ipcs

```
---- Segmentos memoria compartida ----
key        shmid      propietario perms      bytes      nattch     estado
0x00000002 65536      root        600        655360     2
0x00000000 131073     pablo       600        393216     2          dest

------ Matrices semáforo -------
key        semid      propietario perms      nsems
0x00000000 32768      www-data    600        1

------ Colas de mensajes -----
key        msqid      propietario perms      bytes utilizados mensajes
```

$ ipcs -s # muestra únicamente los semáforos

Introducción: Semáforos (3)

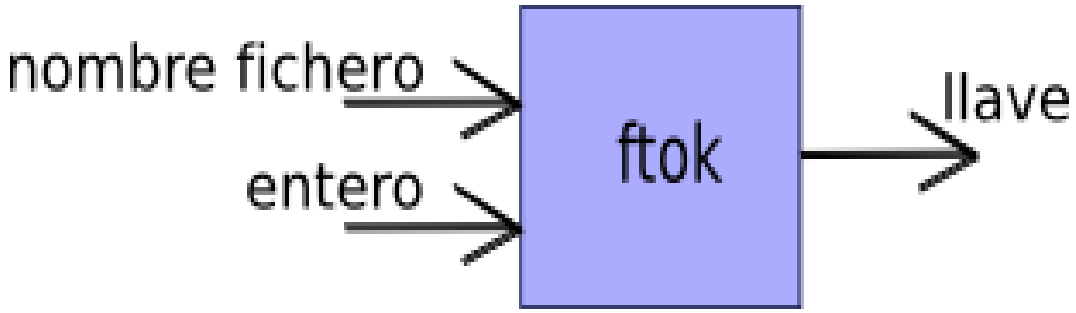
- **Ejercicio ejemplo a solucionar al final del tema:** *Dados dos procesos A y B independientes, A realiza escrituras en el fichero* ***datos.txt*** *y B realiza lecturas de dicho fichero. El proceso B únicamente deberá leer del fichero si hay datos escritos por A. Por otro lado, el proceso A no podrá sobreescribir algo que B no haya leido aún.*



Claves: ftok(), **f**ile **to** **k**ey

- **Desde el punto de vista del ser humano:** Todo mecanismo de IPC se identifica mediante un nombre de fichero, siguiendo así el paradigma de UNIX, y un entero.
- **Desde el punto de la máquina:** Todo mecanismo de IPC se identifica mediante una clave

¿Cómo se realiza la conversión? mediante **ftok**



Claves: ftok(), **f**ile **to** **k**ey (2)

- **key_t ftok(char *filename, int id)**

  @filename: nombre del fichero
  @id: número del identificador

  Valor devuelto: En caso de éxito, una clave es devuelto. En caso de error, se devuelve -1 y errno toma el valor del error apropiado.

  Pero, **¿qué nombre de fichero y qué número de identificación elijo para identificar mis semáforos?** Cualquiera, siempre que:

- El fichero exista en el sistema: por ejemplo, /bin/ls
- Si estás creando un **nuevo** semáforo, **asegúrate** de que no eliges un nombre de fichero y un identificador ya usado

Creación u Obtención de semáforos: semget()

- **int semget(key_t key, int nsems, int semflg)**
  @key: clave que identifica a un **conjunto** de semáforos
  @nsems: número de semáforos que forman parte del conjunto
  @semflags: permisos y banderas (IPC_CREAT, IPC_EXCL)

  Valor devuelto: En caso de éxito, el valor del identificador del semáforo es devuelto. En caso de error, se devuelve -1 y errno toma el valor del error apropiado.

  Pero, **¿qué relación hay entre la clave y el identificador de un conjunto de semáforos?** Existe una relación 1:1, por cada clave hay asociada un único conjunto de semáforos. Véase que la clave identifica un conjunto de IPCs en los que se puede incluir un semáforo.

Ejemplo: ftok y semget

```
key_t llave;
int semid;
key = ftok("/bin/ls", 1);
if (key == -1) {
        perror("ftok");
        exit(EXIT_FAILURE);
}
semid = semget(key, 1, IPC_CREAT);
if (semid == -1) {
        perror("semget");
        exit(EXIT_FAILURE);
}
```

Operando con semáforos: semop()

- **int semop(int semid, struct sembuf *ops, int nops)**
  @semid: identificador del conjunto de semáforos (¡ojo!, la clave **no**)
  @ops: array de operaciones a realizar sobre un semáforo
  @nops: longitud del array de operaciones

```
struct sembuf {
    ushort_t sem_num; /* número del semáforo en el conjunto 0..N-1*/
    short    sem_op;  /* valor negativo equivale a down,
                         valor positivo equivale a up */
    short    sem_flag;  /* dejar a cero */
}
```

Valor devuelto: En caso de éxito, se devuelve 0. En caso de error, se devuelve -1 y errno toma el valor del error apropiado.

Ejemplo: semop

```
...
semid = semget(key, 1, IPC_CREAT);
if (semid == -1) {
        perror("semget");
        exit(EXIT_FAILURE);
}
struct sembuf op_down = { 0, -1, 0 };

if (semop(semid, &op_down, 1) == -1) {
       perror("semop");
       exit(EXIT_FAILURE);
}
```

Control sobre semáforos: semctl()

- **int semctl(int semid, int semnum,int cmd, ...)**
  @semid: identificador del conjunto de semáforos (¡ojo!, la clave **no**)
  @semnum:número de semáforo en el conjunto
  @cmd: operación de control a realizar
  @**...**: argumento opcional, **union semun arg**

```
union semun {
        int val;
        ...         /* no trasciende */
}
```

Valor devuelto: En caso de éxito, se devuelve 0. En caso de error, se devuelve -1 y errno toma el valor del error apropiado.

Control sobre semáforos: semctl()

- ¿Qué operaciones de control son posibles? Las más importantes:
  - **GETVAL**: obtiene el valor actual del contador
  - **SETVAL**: establece el valor actual del contador
  - **IPC_RMID**: destruye un conjunto de semáforos del sistema
- Otras operaciones:
  - **GETNCNT**: número de procesos en la lista de espera del semáforo
  - **GETPID**: PID del último proceso que actuó sobre el semáforo
  - ...

Ejemplo: semctl

```
...
union semun {
    int val;
} arg;

arg.val = 1;
/* Inicializa el contador del semáforo a 1 */
if (semctl(semid, 0, SETVAL, arg) == -1) {
    perror("semctl");
    exit(EXIT_FAILURE);
}
```