Sistemas Operacionais

Semáforos

André Luis Martinotto

UCS UNIVERSIDADE DE CAXIAS DO SUL

Semáforos

- Um semáforo não é usado para trocar dados entre processos mas para sincronizar as suas operações.
- Um semáforo não é um mecanismo de comunicação de dados entre processos, mas uma primitiva de sincronização.

Semáforos

- Um semáforo é uma variável inteira que funciona como contador de recursos.
  - O valor corrente desta variável representa o número de unidades disponíveis de um dado recurso num dado instante.

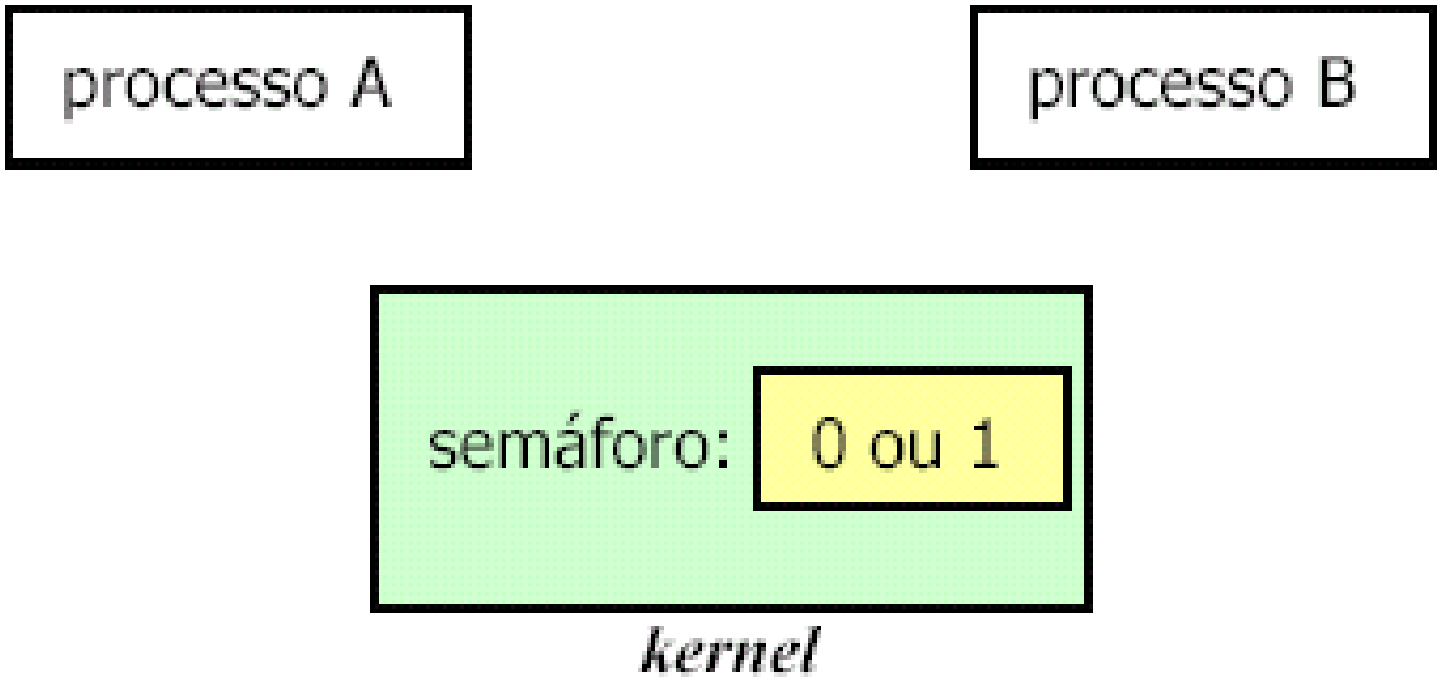
Semáforos

- Para obter um recurso que é controlado por um semáforo:
  - Testar o valor corrente do semáforo.
  - Se o valor for positivo, o processo pode usar o recurso.
  - O processo decrementa o semáforo para indicar que vai usar uma unidade do recurso.
  - Se o valor for 0, o processo vai dormir até que o valor fique positivo.
  - Na utilização de um recurso por um processo requer que este decremente o valor do semáforo
  - Na liberação de um recurso requer que o processo incremente o valor do semáforo

Semáforos

- A implementação correta de semáforos requer que as operacões sejam atômicas
  - Desta forma semáforos são normalmente implementados dentro do kernel do sistema operacional;
  - No kernel é possível garantir que um conjunto de operações sobre um semáforo seja feita atomicamente.

Semáforos



Semáforos

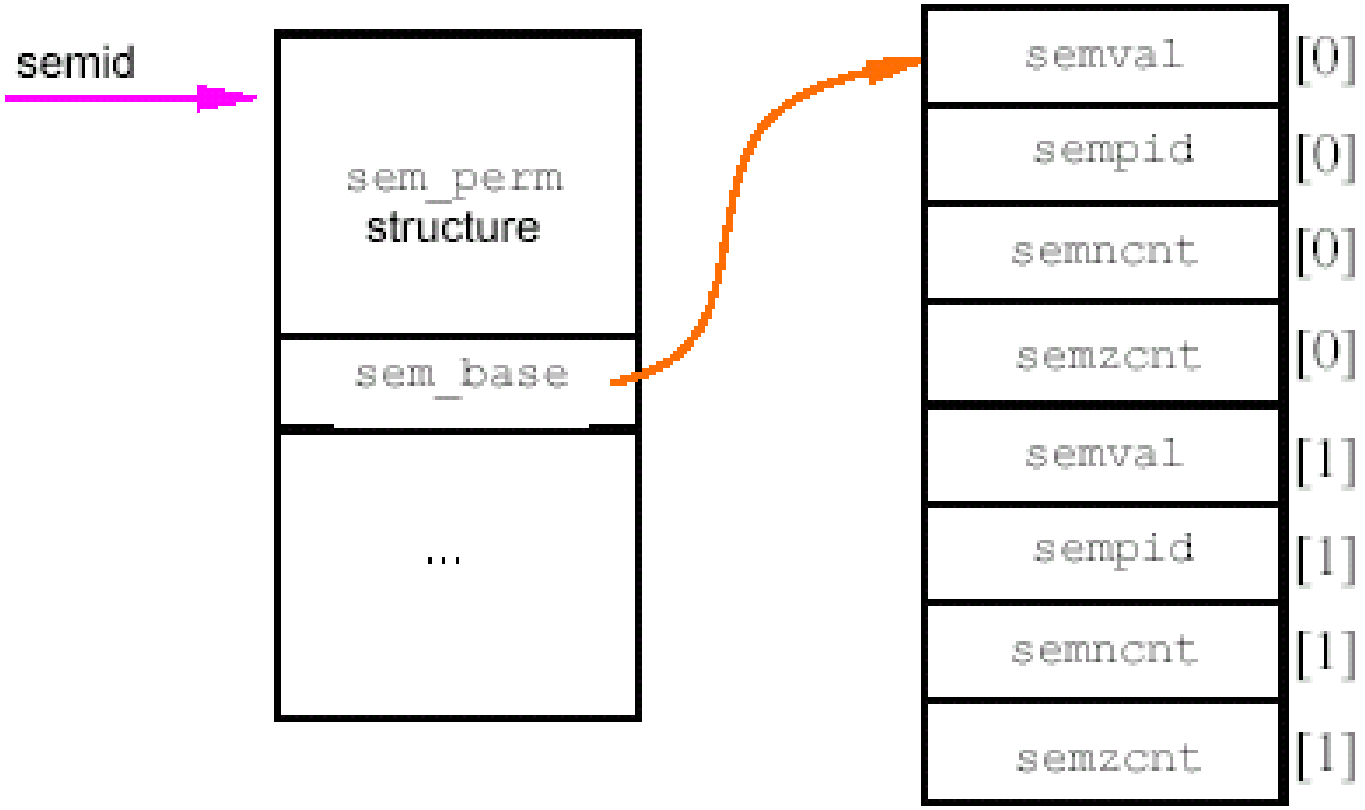
- A primitiva semget permite criar um conjunto de semáforos.
  - Requer três parâmetros:
    - O identificador global para um array de semáforos;
    - O número de semáforos a ser criados
    - Um flag estipulando ações relativas a permissões e criação.
    - Retorna o identificador do array de semáforos

```
int semget(key_t key, int nsems, int flag);
```

Semáforos

- O endereçador aponta para memória reservada do kernel (aponta para um array de semáforos)
  - Cada semáforo contém:
    - O valor do semáforo,
    - O identificador do processo que fez a última operação sobre este valor;
    - O contador do número de processos à bloqueados;
    - O contador do número de processos dos processos esperando que o valor do semáforo se torne 0.

Semáforos


semid
sem_perm
structure
sem_base
...
semval
sempid
semncnt
semzcnt
[0]
[0]
[0]
[0]
semval
sempid
semncnt
semzcnt
[1]
[1]
[1]
[1]

Semáforos

- As operações sobre semáforos se processam através da chamada de sistema semop.
  - Requer três parâmetros:
    - O identificador global para um array de semáforos;
    - Um array de estruturas contendo uma estrutura sembuf que especifica a operação a ser executada;
    - O número de semáforos nos quais serão executadas as operações;

```
int semop(int semid, struct sembuf semarray[ ], size_t nops);
```

Semáforos

- Estrutura de sembuf (estrutura de operações);

```
struct sembuf {
        short sem_num;
        short sem_op;  % <0 executa uma   operação DOWN
                       % >0 executa uma   operação UP
        short sem_flag (SEM_UNDO);
};
```

Semáforos

·A chamada de sistema semctl é empregada para obter e alterar o status de semáforos (inicialização, permissões, desativação, etc).

- O identificador global para um array de semáforos;
- O identificador do elemento do semáforo;
- Especifica um comando a ser executado (IPC_RMID, GETVAL, SETVAL)
- O quarto argumento é variável do tipo union semun com os argumentos de entrada ou saída;

int semctl(int semid, int semnum, int cmd, union semun arg);

Semáforos

- União de semun (estrutura de operacoes);

```
union semun {
        int val;
        struct semid_ds *buf;
        ushort *array;
};
```

Semáforos

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define KEYSEM 1

union semun{
    int valor;
};

void main(){

    pid_t pid;
    int semid;
    key_t keysem = KEYSEM;
    union semun arg;
    struct sembuf semaforo;
```

Semáforos

```
int i;

if ( ( semid = semget ( keysem, 1, IPC_CREAT | 0600 )) < 0 )
        printf("Erro na criacao do semaforo");

arg.valor = 1;
semctl(semid, 0, SETVAL,arg);
printf ("%i\n", semctl(semid, 0, GETVAL, arg));

pid = fork();
```

Semáforos

```
if ( pid > 0){
        sleep(1);
        semaforo.sem_num = 0;
        semaforo.sem_op = -1;
        semaforo.sem_flg = SEM_UNDO;
        semop(semid,&semaforo,1);

        printf("Pai entrou\n");
        sleep(1);
        printf("Pai saiu\n");

        semaforo.sem_num=0;
        semaforo.sem_op = 1;
        semaforo.sem_flg = SEM_UNDO;
        semop(semid,&semaforo,1);

        wait(NULL);
        semctl(semid,IPC_RMID,0);
}
```

Semáforos

```
else{
        semaforo.sem_num = 0;
        semaforo.sem_op = -1;
        semaforo.sem_flg = SEM_UNDO;
        semop(semid,&semaforo,1);

        printf("Filho entrou\n");
        sleep(2);
        printf("Filho saiu\n");

        semaforo.sem_num=0;
        semaforo.sem_op = 1;
        semaforo.sem_flg = SEM_UNDO;
        semop(semid,&semaforo,1);

    }
}
```