Prácticas de Sistemas operativos

David Arroyo Guardeño

Escuela Politécnica Superior de la Universidad Autónoma de Madrid

Octava semana: semáforos

Tercera práctica

1. Memoria compartida: ejercicio 2
2. Semáforos: ejercicios 4 y 5
3. Ejercicio final: ejercicios 6 y 7 (opcional)

Jueves 3 abril

Semáforos

- ✓ Mecanismo para implementar sincronización
- ✓ Número entero: si se intenta hacer menor que cero, el proceso queda en espera
- ✓ Biblioteca sem.h $\rightarrow$ **man**
  - ✗ **semget**
  - ✗ **semop**
  - ✗ **semctl**

```
main()
{
  int sem_id,i;
  struct sembuf sem_oper;
  union semun{
    int val;
    struct semid_ds *semstat;
    unsigned short *array;
  }arg;
  sem_id = semget(SEMKEY,N_SEMAFOROS,IPC_CREAT|IPC_EXCL|SHM_R|SHM_W);
  if(sem_id==-1){
    perror("semget.");
    exit(errno);
  }
  if(N_SEMAFOROS<2){
    fprintf(stderr,"El numero de semaforos ha ser mayor que dos\n");
    exit(0);
  }
  arg.array = (unsigned short*)malloc(sizeof(short)*N_SEMAFOROS);

  for(i=0;i<N_SEMAFOROS;i++)
    arg.array[i] = 1;

  semctl(sem_id,N_SEMAFOROS,SETALL,arg);

  sem_oper.sem_num = 0;
  sem_oper.sem_op = -1;
  sem_oper.sem_flg = SEM_UNDO;
  semop(sem_id,&sem_oper,1);

  sem_oper.sem_num = 1;  sem_oper.sem_op = 1;  semop(sem_id,&sem_oper,1);

  semctl(sem_id,N_SEMAFOROS,GETALL,arg);
  printf("Los valores de los semaforos son [%d",arg.array[0]);
  for(i=1;i<N_SEMAFOROS;i++)
    printf(",%d",arg.array[i]);
  printf("]\n");
  semctl(sem_id,N_SEMAFOROS,IPC_RMID,0);    free(arg.array);
}
```

Ejemplo 2: productor

```
#define SEMKEY 76102
union semun
{
  int val;
  struct semid_ds * buf;
  ushort *array;
  struct seminfo *__buf;
};
main()
{
  key_t key = SEMKEY;
  int sem_id;
  union semun arg;
  enum {NS,EO};
  static struct sembuf sig = {0,1,SEM_UNDO}, wait={0,-1,SEM_UNDO};
  static ushort start_value [2] = {0,0};
  int clock=3;

  if((sem_id = semget (key,2,IPC_CREAT|0666))==-1){
    perror ("control: semget .");
    exit(1);
  }

  arg.array =  start_value;

  if(semctl(sem_id,0,SETALL,arg)==-1){
    perror ("control: semctl --inicializacion."); exit(2);
  }

  printf ("control: Estoy listo.\n");

  while(clock){
    sig.sem_num = NS;

    if(semop(sem_id,&sig,1)==-1){
      perror ("control : semop --signal.");
```

Ejemplo 2: productor

```
main()
{
   ...

   arg.array =  start_value;

   if(semctl(sem_id,0,SETALL,arg)==-1){
     perror ("control: semctl --inicializacion."); exit(2);
   }

   printf ("control: Estoy listo.\n");

   while(clock){
     sig.sem_num = NS;

     if(semop(sem_id,&sig,1)==-1){
       perror ("control : semop --signal.");
       exit(3);
     }


     wait.sem_num = EO;
     if(semop(sem_id,&wait,1)==-1){
       perror ("control : semop --wait.");
       exit(4);
     }

     printf ("control: La carretera este-oeste esta abierta.\n");
     sleep(1);
     clock--;
   }

   if(semctl(sem_id,0,IPC_RMID,0)==-1){
     perror ("control : semctl --remove.");
     exit(5);
   }
```

Ejemplo 2: consumidor

```
...
main()
{
  key_t key = SEMKEY;
  int semid ;
  int car_num =1;


  static struct sembuf sig ={0,1,SEM_UNDO}, wait={0,-1,SEM_UNDO};


  enum {NS,EO};

  if((semid = semget(key,2,IPC_CREAT|0666))==-1){
    perror ("coche: semget .");
    exit(1);
  }

  while(1){
    wait.sem_num = NS;
    if(semop(semid,&wait,1) == -1){
   perror ("coche: semop --wait.");
       exit(2);
    }

    printf("coche: La carretera norte-sur esta abierta.\n");
    sleep(1);
    printf("coche: Coche %d paso.\n",car_num ++);

    sig.sem_num = EO;
    if(semop (semid,&sig,1)==-1){
      perror ("coche: semop --signal.");
      exit(3);
    }
  }
}
```

Ejemplo 3: productor

```
#define SHMKEY 6723
#define SEMKEY 7543
#define NUMPEDIDOS 10

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

main() {
  key_t key= SHMKEY;
  key_t key2 = SEMKEY;

  int shmid;
  int *sopa = 0;
  int orden = NUMPEDIDOS;

  int sem_id;
  union semun arg;
  enum {COOK,PICK};
  static struct sembuf cook = {0,1,SEM_UNDO}, pick = {0,-1,SEM_UNDO};
  static ushort start_value [2] = {0,0};

  if((sem_id = semget (key2,2,IPC_CREAT|0666))==-1){
      perror ("cocina: semget .");
      exit(1);
  }

  arg.array =  start_value;

  if(semctl(sem_id,0,SETALL,arg)==-1){
      perror ("cocina: semctl --inicializacion."); exit(2);
```

Ejemplo 3: productor

```
main() {
  ...

  if((shmid = shmget(key,50,0600|IPC_CREAT))==-1) {
    perror("cocina: shmget"); exit(1);
  }

  sopa = (int *)shmat(shmid,(char *)0,0);
  if (sopa == (int *)-1) {
    perror("cocina: shmat"); exit(2);
  }

  printf("Cocinero: He empezado a hacer la sopa.\n");
  while(orden) {
    sleep(1);
    (*sopa)++;

    cook.sem_num = COOK;
    if(semop(sem_id,&cook,1)==-1){
    perror ("cocina : semop --cook.");
    exit(3);
    }

    pick.sem_num = PICK;
    if(semop(sem_id,&pick,1)==-1){
      perror ("control : semop --wait."); exit(4);
    }
    printf("Oido cocina...\n"); orden--;
  }
  printf("Cocinero: Se acabo el tiempo. He preparado %d platos de sopa. Quedan %d sin recoger.\n"
     ,NUMPEDIDOS,* sopa); shmctl(shmid,IPC_RMID,(struct shmid_ds *) 0);

  if(semctl(sem_id,0,IPC_RMID,0)==-1){ perror ("cocina : semctl --remove."); exit(5); }
  exit(0);
}
```

Ejemplo 3: consumidor

```
void manejador() {
  exit(0);
}
main() {
  void signal_catcher();
  key_t key = SHMKEY;
  key_t key_sem = SEMKEY;
  int shmid;
  int *sopa;


  int sem_id;
  static struct sembuf cook ={0,1,SEM_UNDO}, pick={0,-1,SEM_UNDO};

  enum {COOK,PICK};

  if((sem_id = semget(key_sem,2,IPC_CREAT|0666))==-1){  perror ("camarero: semget .");        exit(1);   }

  sigset(SIGALRM,manejador);
  alarm(30);

  if((shmid = shmget(key,50,IPC_CREAT|0600))==-1) {    perror("shmget");    exit(1);  }
  if((sopa = (int *)shmat(shmid,0,0))==(int *)-1) {    perror("shmat");    exit(2);  }

  while(1){
    pick.sem_num = COOK;
    if(semop(sem_id,&pick,1) == -1){       perror ("camarero: semop --pick."); exit(2);     }
    (*sopa)--;
    printf("Camarero: plato recogido\n");
    sleep(rand()%2+1);
    printf("Camarero: un nuevo plato de sopa, cocina!!!\n");
    cook.sem_num = PICK;
    if(semop (sem_id,&cook,1)==-1){       perror ("camarero: semop --cook.");       exit(3);
    }
  }
}
```

Referencias

- Francisco M. Márquez. Unix, Programación Avanzada. Editorial: Ra-Ma. 3ª Edición. ISBN: 84-7897-603-5