CODIGOS-PRACTICAS-4-y-5.pdf

paco_porras

Sistemas Operativos

2º Grado en Ingeniería Informática

Escuela Superior de Ingeniería
Universidad de Almería

Accede al documento original

ssoo@ssoo: ~/Practica4

GNU nano 7.2

```c
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define SEMKEY 75


//Declaración de variables para el semáforo
   int semid;
   struct sembuf sem_oper;
   union semnum {
        int val;
        struct semid_ds *semstat;
        unsigned short *array;
   } arg;

//Estructura de datos de información
struct Info {
   char nombre[30];
   char apellido1[50];
   char apellido2[50];
   int num;
   char email[50];
} InfoEmisor;

//Declaración de funciones
void *funcion_Emisor(void *);
void *funcion_Receptor(void *);


// PROGRAMA PRINCIPAL
int main (void) {
   pthread_t id_Emisor, id_Receptor;

    //Creación del semáforo
   semid = semget(SEMKEY, 1, 0777 | IPC_CREAT);
   if (semid == -1) {
      printf("Error en la creación del semáforo.");
      exit(1);
   }
```

ssoo@ssoo: ~/Practica4

```
  GNU nano 7.2
   //Inicialización del semáforo
   arg.array=(unsigned short *)malloc (sizeof (short)*2);
   arg.array[0] = 0;
   semctl (semid, 0, SETVAL, arg);

   //Creación de hilos
   printf("\nCreando hilos...\n");
   fflush(stdout);
   pthread_create(&id_Emisor, NULL, funcion_Emisor, (void *) NULL);
   pthread_create(&id_Receptor, NULL, funcion_Receptor, (void *) NULL);
   pthread_join(id_Emisor, NULL);
   pthread_join(id_Receptor, NULL);
   printf("\nTerminando...\n");
   fflush(stdout);
}


// IMPLEMENTACIÓN DE FUNCIONES
void *funcion_Emisor(void *) {

   printf("\nSoy el hilo Emisor, introduzca los siguientes datos: \n");

   printf("\nIntroduzca el nombre: ");
   scanf("%s",InfoEmisor.nombre);
   printf("Introduzca el primer apellido: ");
   scanf("%s",InfoEmisor.apellido1);
   printf("Introduzca el segundo apellido: ");
   scanf("%s",InfoEmisor.apellido2);
   printf("Introduzca un teléfono: ");
   scanf("%d",&InfoEmisor.num);
   printf("Introduzca un email: ");
   scanf("%s",InfoEmisor.email);

   //Operación signal al semáforo
   sem_oper.sem_num = 0;
   sem_oper.sem_op = 1;
   sem_oper.sem_flg = SEM_UNDO;
   semop (semid, &sem_oper, 1);

   fflush(stdout);
   pthread_exit(NULL);
```

```
void *funcion_Receptor(void *) {

   //Operación wait, espera hasta
   sem_oper.sem_num = 0;
   sem_oper.sem_op = -1;
   sem_oper.sem_flg = SEM_UNDO;
   semop (semid, &sem_oper, 1);
```

ssoo@ssoo: ~/Practica5

GNU nano 7.2 Emisor.c

```c
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <dirent.h>

#define SEMKEY 75
#define SHMKEY 75
#define SHMSIZE 256

//Declaración de variables para el semáforo
   int semid;
   struct sembuf sem_oper;
   union semnum {
        int val;
        struct semid_ds *semstat;
        unsigned short *array;
   } arg;

//Manejar directorios
void manejarDirectorio(const char *nombreDirectorio, char *addr) {
    DIR *directorio;
    struct dirent *entrada;

    // Intentar abrir el directorio
    directorio = opendir(nombreDirectorio);
    if (directorio == NULL) {
        // Enviar mensaje de error al receptor
        strcpy(addr, "Directorio no existente.");
        sem_oper.sem_num = 0; // Signal al semáforo de elemento disponible
        sem_oper.sem_op = 1;
        sem_oper.sem_flg = SEM_UNDO;
        semop(semid, &sem_oper, 1);
        return;
    }
```

ssoo@ssoo: ~/Practica5

GNU nano 7.2 Emisor.c

```
    while ((entrada = readdir(directorio)) != NULL) {

        if (strcmp(entrada->d_name, ".") == 0 || strcmp(entrada->d_name, "..") == 0) {
            continue;
        }

        // Operación wait al semáforo de elemento libre
        sem_oper.sem_num = 1;
        sem_oper.sem_op = -1;
        sem_oper.sem_flg = SEM_UNDO;
        semop(semid, &sem_oper, 1);

        strcpy(addr, entrada->d_name);

        // Signal al semáforo de elemento disponible
        sem_oper.sem_num = 0;
        sem_oper.sem_op = 1;
        sem_oper.sem_flg = SEM_UNDO;
        semop(semid, &sem_oper, 1);
    }

   closedir(directorio);

// Enviar mensaje "FIN" para indicar al receptor que ya no hay más datos
strcpy(addr, "FIN");

// Signal al semáforo de elemento disponible
sem_oper.sem_num = 0;
sem_oper.sem_op = 1;
sem_oper.sem_flg = SEM_UNDO;
semop(semid, &sem_oper, 1);
}
```

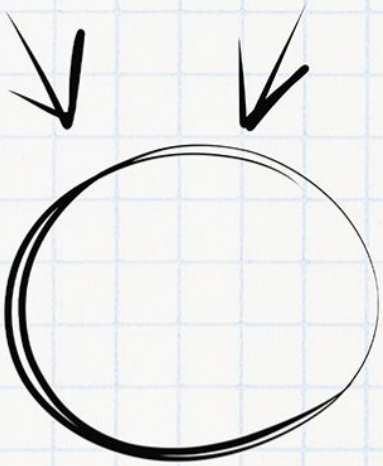
```c
// PROGRAMA PRINCIPAL
int main (int argc, char *argv[])
{

  int shmid;
  char *addr;

  if (argc != 2) {
      fprintf(stderr, "Pon una ruta de un directorio.");
      exit(EXIT_FAILURE);
  }

   // Crear la región de memoria y obtener la dirección
   shmid = shmget(SHMKEY, SHMSIZE, 0777 | IPC_CREAT);
   if (shmid == -1) {
   printf("Error en la creación de la Memoria compartida");
   exit(1);
   }

    //Creación del semáforo
   semid = semget(SEMKEY, 2, 0777 | IPC_CREAT);
   if (semid == -1) {
      printf("Error en la creación del semáforo.");
      exit(1);
   }

   //Inicialización del semáforo
   arg.array=(unsigned short *)malloc (sizeof (short)*2);
   arg.array[0] = 0;
   arg.array[1] = 1;
   semctl (semid, 2, SETVAL, arg);

   // Enlazar región al proceso
   addr = shmat(shmid, 0, 0);

   //Llamada al metodo para ir produciendo los directorios
   manejarDirectorio(argv[1], addr);
   shmdt(addr);
  return 0;
}
```

Imagínate aprobando el examen

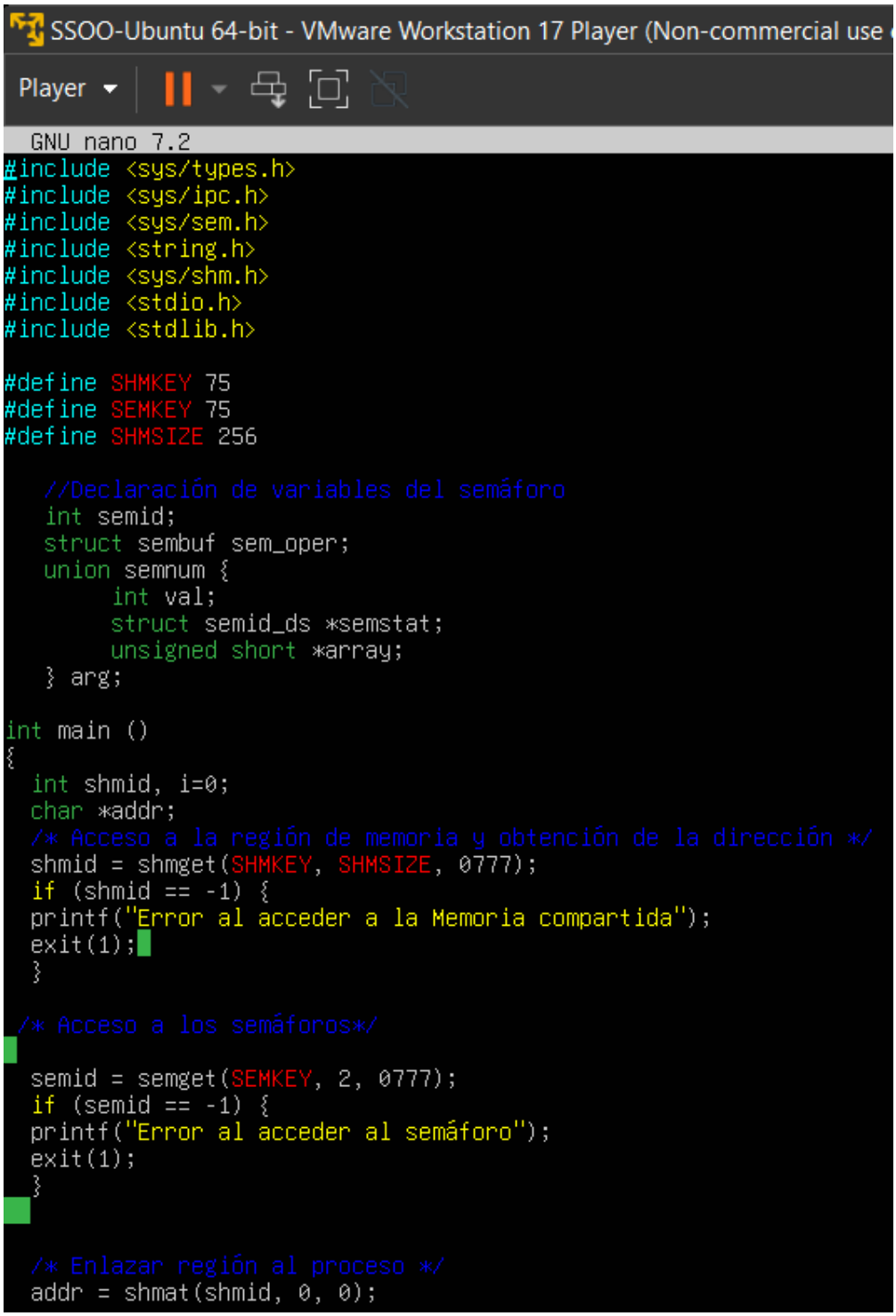
Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH




```c
while (1) {
  //Operación wait al semáforo disponible
  sem_oper.sem_num=0;
  sem_oper.sem_op=-1;
  sem_oper.sem_flg=SEM_UNDO;
  semop (semid, &sem_oper, 1);

  if (strcmp(addr, "FIN") == 0) {
     break;
    }

  printf("%s\n", addr);

  //Operación signal al semáforo elemento libre
  sem_oper.sem_num=1;
  sem_oper.sem_op=1;
  sem_oper.sem_flg=SEM_UNDO;
  semop (semid, &sem_oper, 1);
}
  shmdt(addr);
  shmctl(shmid, IPC_RMID, 0);
  return 0;
  }
```