Consignas

1) Descargar las consignas (hacer clic en el botón a la derecha del archivo):

Primer examen parcial

2) Resolver el ejercicio.

3) Enviar las resolución como respuesta a esta actividad.

Condiciones de corrección (si no se cumple lo siguiente, no se corrige)

- El programa debe compilar sin errores.
- El programa debe correr sin errores de memoria.
- Debe estar correctamente identado.
- Utilizar constantes.

Puntos a tener en cuenta

- Los nombres de las variables deben ser descriptivos de su función.
- Comentar el código todo lo posible.
- Toda variable o arreglo definido en forma dinámica debe ser liberado al finalizar el programa.

4) Para utilización de Semaforos y memoria compartida se puede usar el siguiente código (recordar utilizar y crear los correspondientes .h)

Semaforo:

```
#include <sys/ipc.h>
 #include <sys/sem.h>
#include <stdio.h>
#define ROJO 0
#define VERDE 1
#define CLAVE_BASE 33


key_t creo_clave(int r_clave) {
// Igual que en cualquier recurso compartido (memoria compartida, semaforos // o colas) se obtien una clave a partir de un fichero existente cualquiera // y de un entero cualquiera. Todos los procesos que quieran compartir este // recurso, deben usar el mismo fichero y el mismo entero.
```

```
key_t clave;
clave = ftok ("/bin/ls", r_clave);
if (clave == (key_t)-1)
{ printf("No puedo conseguir clave para memoria compartida\n");
exit(0);
}
return clave;
}


//funcion que crea el semaforo int creo_semaforo() {
key_t clave = creo_clave(CLAVE_BASE);
int id_semaforo = semget(clave, 1, 0600|IPC_CREAT); //PRIMER PARAMETRO ES LA CLAVE,
EL SEGUNDO CANT SEMAFORO, EL TERCERO 0600 LO UTILIZA CUALQUIERA, IPC ES
CONSTANTE (VEA SEMAFORO)
if(id_semaforo == -1) {
printf("Error: no puedo crear semaforo\n");
exit(0);
}
return id_semaforo;
}


//inicia el semaforo
void inicia_semaforo(int id_semaforo, int valor) {
semctl(id_semaforo, 0, SETVAL, valor);
}


//levanta el semaforo
void levanta_semaforo(int id_semaforo) {
struct sembuf operacion; printf("Levanta SEMAFORO \n");
operacion.sem_num = 0;
operacion.sem_op = 1; //incrementa el semaforo en 1 operacion.sem_flg = 0;
semop(id_semaforo,&operacion,1);
}


//espera semaforo
void espera_semaforo(int id_semaforo) {
struct sembuf operacion;


printf("Espera SEMAFORO \n"); operacion.sem_num = 0;
operacion.sem_op = -1; //decrementa el semaforo en 1 operacion.sem_flg = 0;
semop(id_semaforo,&operacion,1);
}
```

MemoriaCompartida:

```
#include <sys/shm.h>
#include <stdio.h>
#include <sys/ipc.h>


#define CLAVE_BASE 33


typedef struct tipo_dato dato; struct tipo_dato { int numero; char letra; };


key_t creo_clave(int r_clave)
{
// Igual que en cualquier recurso compartido (memoria compartida, semáforos // o colas) se obtiene una clave a partir de un fichero existente cualquiera // y de un entero cualquiera. Todos los procesos que quieran compartir esta // memoria, deben usar el mismo fichero y el mismo entero.
key_t clave;
clave = ftok ("/bin/ls", r_clave);
if (clave == (key_t)-1)
{
printf("No puedo conseguir clave para memoria compartida\n");
exit(0);
}
return clave;
}


void* creo_memoria(int size, int* r_id_memoria, int clave_base)
{
  void* ptr_memoria; int id_memoria;
 id_memoria = shmget (creo_clave(clave_base), size, 0777 | IPC_CREAT);
 if (id_memoria == -1)
{
printf("No consigo id para memoria compartida\n");
exit (0);
}
ptr_memoria = (void *)shmat (id_memoria, (char *)0, 0);
if (ptr_memoria == NULL)
{
printf("No consigo memoria compartida\n");
exit (0);
}
*r_id_memoria = id_memoria; return ptr_memoria;
}
```

5) link CoCalc:
https://cocalc.com/projects