Ingeniería en Sistemas Computacionales

Fundamentos de Sistemas Operativos

Actividad 16

IS727272 - Marco Ricardo Cordero Hernández

Jal., 20 de junio de 2023

1.- Demostrar que los semáforos y los mensajes son equivalentes de la siguiente forma:

A. A partir de un sistema de paso de mensajes hay que implementar un semáforo entero, las primitivas que deberás definir son:

a. `waitsem(buzón)`: Realizar la operación equivalente a wait para un semáforo

```
waitsem(buzón) {
     type msg;
     receive(buzón, &msg, BLOQUEANTE);
}
```

b. `signalsem(buzón)`: Realizar la operación equivalente a signal para un semáforo

```
signalsem(buzón) {
     type msg;
     send(buzón, &msg, NO_BLOQUEANTE);
}
```

c. `initsem(buzón,n)`: Realizar la operación equivalente a la inicialización del semáforo donde n será el valor inicial del semáforo.

```
initsem(buzón, n) {
     if (buzón no existe) {
          crear_buzón(buzón);

          type msg;
          for (i = 0; i < n; ++) {
               send(buzón, &msg, NO_BLOQUEANTE);
          }
     }
}
```

B. A partir de los semáforos, hay que implementar un sistema de paso de mensajes. Pista: haga uso de un área de elementos compartidos para contener un buzón, cada uno de ellos formado por un arreglo de espacios para los mensajes, y usar los semáforos para asegurar la exclusión mutua en el arreglo, asegurar que send no almacenará un nuevo mensaje en el arreglo y asegurar que receive no avanzará si el buzón está vacío.

```
#define SIZE_BUZON 50
int id_buzon;

crear_buzon(){
     //imaginemos que tiene sus semáforos, tamaño y dirección de memoria como
atributos
     struct buzon mi_buzon;
```

```
	//espacio de memoria donde estará el arreglo de mensajes para el buzón
	mi_buzon.mem = shmget(0x1111, SIZE_BUZON * sizeof(struct mensaje), IPC_CREAT
| 0666);

	//crear semáforo entero para no pasarnos del tamaño del buzón
	mi_buzon.sem_max  =  semget(0x2222,1,0666|IPC_CREAT);semctl(mibuzon.sem_max,
0,SETVAL,SIZE_BUZON);

	//crear semáforo binario para no leer y escribir a la vez
	mi_buzon.sem_s = semget(0x3333,1,0666|IPC_CREAT);
	semctl(mibuzon.sem_max, 0,SETVAL,1);

	//crear semáforo binario para que no pueda leer del buzón si está vacío
	mi_buzon.sem_retraso = semget(0x4444,1,0666|IPC_CREAT);
	semctl(mibuzon.sem_retraso, 0,SETVAL,0);

	mi_buzon.tam = 0;

	return mi_buzon;
}

send(struct buzon b, struct mensaje msg){
	//no dejar que se puede añadir mensajes si ya no hay espacio en el buzón
	wait(b.sem_max);

	//no dejar que se pueda añadir mensajes si un proceso esta leyendo
	wait(b.sem_s);

	//escribir el mensaje en el buzón
	b.mem + b.tam = msg;
	b.tam ++;

	//liberar semaforo para leer una vez que entra un dato
	if (b.tam==1)
		signal(b.sem_retraso);
	signal(b.sem_s);
}

receive(struct buzon b, struct mensaje msg){
	//no dejar que intente leer mensajes si no hay en el buzón
	wait(b.sem_retraso);

	//no dejar que lea un mensaje si se esta escribiendo en el buzón
```

```
        wait(b.sem_s);
        b.tam--;
        encontrar el mensaje a leer;
        reacomodar el buzon;//cuando se lee un mensaje se incrementa el semáforo de espacio disponible
        signal(b.sem_max);
}
```

2.-¿Qué aprendiste?

Se vieron sistemas de mensajes y demás cositas.