Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Sistemas Operativos y Distribuidos

Trabajo Práctico Nº 5

Sincronización y Comunicación entre Procesos

Segundo Cuatrimestre de 2015

Ejercicios

1. ¿Qué entiende por *espera ocupada* o *busy waiting*? ¿Cuál es su objetivo? ¿Podría alcanzarse la misma meta mediante otro mecanismo?

2. Determine la veracidad de la siguiente afirmación:

   > *"La espera ocupada es un mecanismo de sincronización primitivo, no resulta conveniente y no es utilizado por ningún sistema operativo moderno."*

3. ¿Los semáforos representan una solución al problema de la sección crítica? ***Justifique adecuadamente***.

4. Muestre cómo la exclusión mutua puede ser violada si las operaciones *wait()* y *signal()* sobre un semáforo no se ejecutan de forma atómica.

5. Explique por qué los *spinlocks* son típicamente utilizados en ambientes multiprocesador. Argumente sobre la viabilidad de los mismos en un sistema con un único procesador.

6. Determine si el el siguiente *pseudo-código* representa una solución al problema de la *sección crítica*.

```
shared int turn = 1;
int myPID = 0;
int otherPID = 1 - myPID;

while (turn != myPID) do no-op;        // Entry Section

      // Critical Section

turn = otherPID;                       //Exit Section
```

7. Analice si el siguiente *pseudocódigo* representa una solución válida al problema del *productor-consumidor*. ***Justifique adecuadamente***.

```
#define MAX_LENGTH 128
typedef char* msg;

sem mutex = 1;
sem full = 0;
sem empty = MAX_LENGTH;
```

```
void Producer(){
  msg message;
  while(1){
    produce(message);
    wait(&mutex);
    wait(&empty);
    write_msg(mensaje);
    signal(&mutex);
    signal(&full);
  }
}
```

```
void Consumer(){
        msg message;
        while(1){
                wait(&full);
                wait(&mutex);
                read_msg(message);
                signal(&mutex);
                signal(&empty);
                consumir(message);
        }
}
```

8. **El problema del barbero dormilón**. Una barbería está compuesta por una sala de espera con n sillas y una sala de barbería en donde se encuentra la silla del barbero. Si no hay clientes que atender, el barbero se pone a dormir. Si un cliente entra a la barbería y todas las sillas están ocupadas, entonces el cliente se retira. Si el el barbero está ocupado pero existen sillas disponibles, entonces el cliente se sienta a esperar en una de las sillas libres. Si el barbero está dormido, el cliente lo despierta.

   Implemente una solución en *pseudo-código* permita sincronizar al barbero con sus clientes utilizando semáforos.

9. **El problema de los filósofos cenando**. Cinco filósofos están sentados alrededor de una mesa redonda. Cada filósofo posee un plato de arroz, y entre cada par de filósofos existe un palillo. Cada filósofo está en uno de tres estados posibles: *pensando*, *hambriento* o *comiendo*. En un determinado momento, un filósofo que estaba pensando siente hambre, por lo que intenta obtener uno de los palillos adyacentes y posteriormente el otro. Si un filósofo es capaz de obtener los dos palillos (no están siendo utilizados), entonces el filósofo come durante una cierta cantidad de tiempo. Luego de comer, el filósofo retorna ambos palillos a la mesa.

   Brinde una solución al problema utilizando un *monitor*.

10. **El problema de los lectores y escritores**. Un cierto número de lectores pueden leer de un recurso compartido simultáneamente. Solo un escritor por vez puede escribir sobre el recurso, y ningún lector puede leer del mismo mientras se está realizando la operación de escritura.

    Desarrolle una solución al problema utilizando semáforos, de forma tal que la misma le de prioridad a los procesos lectores.

11. Un servidor puede ser diseñado para limitar el número de conexiones abiertas. Por ejemplo, un servidor podría mantener N conexiones del tipo *socket* de forma simultánea. En cuanto se establezcan las N conexiones, el servidor no aceptará ninguna conexión adicional hasta que se libere una de las existentes.

    Explique cómo el servidor podría utilizar semáforos para limitar el número de conexiones concurrentes.

Inter-Process Communication (IPC)

12. Muestre cómo podría monitorear las facilidades IPC bajo el sistema operativo GNU/Linux. ¿A qué tipo de información puede acceder?

13. Resuelva los siguientes incisos utilizando las facilidades IPC provistas por el sistema operativo GNU/Linux:

    a) Cree una *cola de mensajes* y verifique la existencia de la misma utilizando el mecanismo descripto en el ejercicio 12. ¿A qué conclusión podría arribar?

    b) ¿Existe alguna forma de generar un identificador IPC único a nivel de sistema? En caso de que su respuesta sea afirmativa, modifique el inciso anterior de forma tal de reflejar dicha característica. Verifique tanto la clave como el id utilizando el mecanismo descripto en el ejercicio 12.

    c) Implemente un programa que permita mostrar por la salida estándar la capacidad de la cola de mensajes previamente creada, para luego modificar el tamaño de la misma a 4 KB.

    d) Desarrolle un programa que reciba por la entrada estándar un identificador de cola y remueva a la misma del sistema. ¿Puede realizar dicha acción directamente desde el intérprete de comandos?

14. Implemente el problema del *productor-consumidor con buffer limitado*, haciendo uso de una cola de mensajes. En particular, considere la existencia de dos procesos productores P_1 y P_2 que generan dos tipos particulares de mensajes cada uno. A su vez, contemple la existencia de m procesos consumidores. Cada consumidor puede consumir un único tipo de mensaje.

15. Desarrolle un programa que permita monitorear el estado de una cola de mensajes, a partir de un identificador de cola recibido como parámetro. El programa debería mostrar por la salida estándar:

    a) El número de mensajes encolados.

    b) El número de bytes en la cola.

    c) El límite de bytes para la cola.

    d) El PID del último proceso que ha escrito en la cola.

    e) El PID del último proceso que ha leído de la cola.

    f) El tiempo en que se ha almacenado el último mensaje en la cola.

    g) El tiempo en el que se ha eliminado el último mensaje de la cola.

16. Resuelva el ejercicio 11 del Trabajo Práctico 2 haciendo uso de un segmento de memoria compartido entre el proceso padre y el proceso hijo. En particular, el proceso padre debe crear el segmento de memoria compartida y el proceso hijo debe computar la secuencia de Fibonacci y almacenar los resultados en el segmento compartido. El proceso padre debe mostrar la secuencia por la salida estándar y eliminar el segmento creado antes de finalizar.

17. Escriba un progama que reciba un identificador de un segmento de memoria compartida e imprima por pantalla la información asociada al mismo.