Parcial Corrección

Ejercicio 1:

1) Cual/es de las siguientes acciones nunca generan una llamada al sistema?

a. Asignar memoria a una variable de tipo puntero
b. Leer la hora del sistema
c. Acceder a una variable de tipo puntero
d. Leer los datos de un buzón (mail box)
e. Incrementar en 1 una variable
d. Abrir un archivo

2) El descriptor de un proceso PCB

a. Sirve para almacenar el contexto completo del proceso cuando no este en ejecución
b. Entre otras cosas, sirve para almacenar el contenido de la pila del proceso
c. En un sistema tipo Unix contendrá, entre otras cosas, los i-nodos de los archivos que el proceso tiene abiertos
d. Contiene específicamente información de la identificación y estado del proceso
e. Ninguna de las anteriores

3) Indicar cuales de las afirmaciones siguientes son falsas:

a. En un sistema con prioridades siempre esta ejecutando el proceso con mayor prioridad
b. En un sistema de tiempo compartido cuando un proceso se bloquea en una operación de E/S, se selecciona otro proceso para que se ejecute
c. Si un sistema soporta multiprogramación también ofrece tiempo compartido
d. En un sistema de multiprogramación cuando un proceso se bloquea en una operación de E/S, se selecciona otro proceso para que se ejecute
e. Una de las características que distingue un sistema paralelo (muy acoplado) de uno distribuido (poco acoplado) es la cantidad de recursos que tienen en común los distintos procesadores

4) En un sistema con una política de planificación no apropiativa ¿Cuáles de las siguientes transiciones entre los estados de los procesos no son normales que se produzca?

a. Ejecución -> Espera
b. Ejecución -> Listo
c. Listo -> Terminación
d. Listo -> Ejecución
e. Ejecución -> Terminación
f. Espera -> Listo

5) En relación con las llamadas al sistema ¿Cuáles de las siguientes afirmaciones son ciertas?

a. Se realizan mediante sencillos saltos a subrutinas (a las rutinas del sistema operativo que implementan cada servicio)
b. Las llamadas al sistema no comprenden solo tipo de servicios que proporciona

el sistema operativo

c. Normalmente, las llamadas al sistema se realizan directamente desde el programa que escribe el usuario en los lenguajes convencionales de alto nivel

d. Las llamadas al sistema requieren cambio de modo.

e. Ninguna de las afirmaciones anteriores es cierta.

Ejercicio 2:

1) Con respecto a los modelos de comunicación de los procesos, se puede decir que:
   a. El modelo directo es siempre síncrono.
   b. El modelo indirecto no puede ser síncrono.
   c. El modelo directo simétrico es siempre asíncrono.
   d. El modelo directo asimétrico unidireccional puede ser síncrono.
   e. El pasaje de mensajes es el único modo de comunicación en múltiples procesadores.
   f. La memoria compartida permite comunicación indirecta
2) En relación con la planificación de procesos es cierto que:
   a. La política de planificación por Round-Robin genera problemas de inanición.
   b. El Proceso Ocioso sólo tiene sentido en las políticas de planificación apropiativas
   c. La técnica del envejecimiento en la planificación por prioridades consiste en aumentar gradualmente la prioridad de los procesos que están en el estado de Esperando = (bloqueado).
   d. La política de planificación El más corto primero (SJF), tiene en cuenta el tiempo que los procesos han estado en la cola de Listo.
   e. Las opciones 1, 3 y 4 son ciertas
   f. Ninguna de las anteriores es cierta
3) En cuanto a los hilos (threads) y al mecanismo de sincronización de procesos, es cierto que:
   a. Un sistema operativo que soporte hilos no utiliza inhibición de interrupciones.
   b. Los pipes sólo pueden usarse para comunicar hilos de un mismo proceso.
   c. Es suficiente el uso de la instrucción Test&Set, en un entorno multiprocesador monousuario que soporte hilos, para evitar las condiciones de carrera.
   d. En la planificación Round-Robin, en un semáforo un hilo que se desbloquea por la operación subir no tiene prioridad para acceder inmediatamente a la CPU.
   e. Los mutex tienen el mismo comportamiento que los semáforos en los hilos.
4) Indicar cuál de las siguientes afirmaciones es cierta en referencia a las llamadas al sistema:
   a. El código escrito por el usuario llama a la rutina de interfaz de usuario mediante la instrucción de excepción trap.
   b. La rutina de tratamiento de interrupción se encarga de llamar a la rutina de servicio del sistema operativo adecuada a la solicitud del programa de usuario.
   c. La rutina de servicio del sistema operativo es una rutina que sirve de interfaz entre el sistema operativo y el hardware del ordenador
   d. Ninguna de las anteriores
5) En relación con las llamadas al sistema es cierto que:
   a. Las llamadas al sistema sólo pueden cambiar el estado del proceso que las

invoca

b. Las llamadas al sistema mediante una rutina de interfaz que utilice un Trap, sólo funciona en una máquina monoprocesador.

c. Un proceso que esté Listo no podrá invocar una llamada al sistema hasta que no pase al estado de Ejecutando.

d. El número de llamadas al sistema está limitado por el número de interrupciones software del procesador.

Ejercicio 3:

1) ¿Cuál de las siguientes acciones no genera una llamada al sistema?
   a. Pedir la hora.
   b. Abrir un archivo.
   c. Dormirse durante n segundos.
   d. Acceder a datos de la pila.
2) En relación con la planificación de procesos es cierto que:
   a. Dos procesos pueden compartir el mismo descriptor de proceso.
   b. La invocación de una llamada al sistema por parte de un proceso, puede provocar que otro proceso pase del estado Esperando a Listo.
   c. Siempre que se invoca una llamada al sistema, se produce un cambio de contexto
   d. La política de planificación el más corto primero es apropiativa
3) Tenemos un planificador de procesos por colas de prioridad apropiativo. Los procesos pueden tener 3 prioridades: 3 (la mayor), 2 y 1. Para los procesos con prioridad 1 y 2, a igualdad de prioridad se gestiona por FCFS (Primero en Llegar Primero en Servir). Para los de prioridad 3 se gestiona por RR (Round Robin). Suponemos que se está ejecutando un proceso de prioridad 2. Indicar cuál de las siguientes situaciones provocaría necesariamente un cambio de contexto.
   a. Llega un proceso nuevo de prioridad 3 a la cola de listos
   b. Llega el fin de un quantum de tiempo
   c. El proceso en ejecución realiza un Bajar (Wait) sobre un semáforo
   d. Ninguna de las anteriores
4) Supóngase que se dispone de una pista de aterrizaje controlada por un controlador aéreo que da órdenes a los pilotos que se lo pidan. Suponiendo que sólo hay un controlador, varios pilotos, que todos ellos están simulados con procesos y que se sigue una política de planificación FIFO, ¿qué ocurrirá en el siguiente caso?

Controlador	Piloto
signal(disponible)	wait(disponible)
signal(trabajo)	signal(trabajo)
Establecen contacto	
wait(pista)	wait(aterriza)
signal(aterriza)	aterrizaje
Espera fin aterrizaje	Fin aterrizaje
wait(finaterriza)	signal(finaterriza)
signal(pista)	

Con valores para todos los semáforos = 0 excepto para pista que vale 1.

   a. Si el controlador está libre y dos pilotos piden atención al controlador

simultáneamente, pueden llegar a estrellarse.

b. Es posible que, algún avión no aterrice nunca

c. Los aviones aterrizan uno a uno y sin problemas.

d. Ninguna

Ejercicio 4:

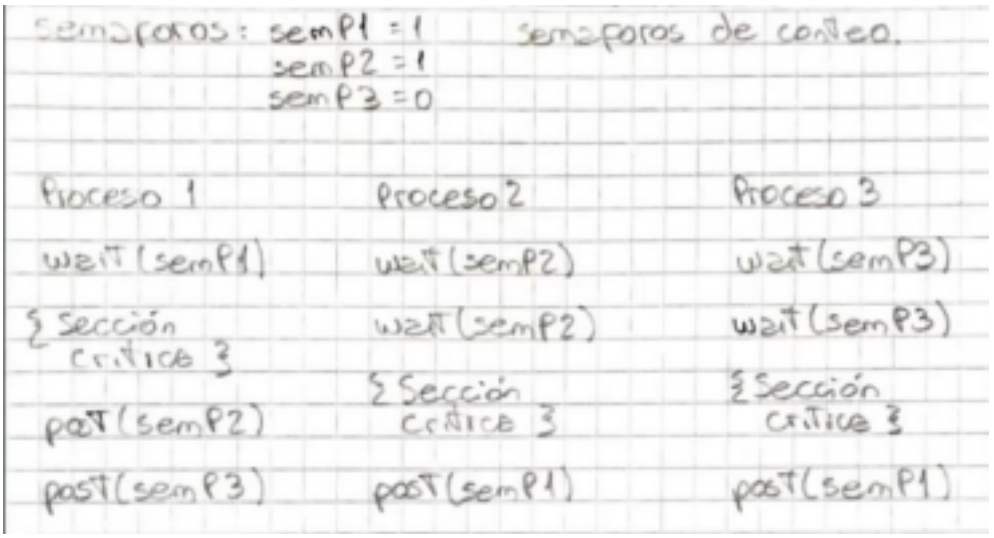
1) Sincronizar mediante semáforos tres procesos de tal manera que luego de ejecutarse la sección critica del proceso 1 se ejecute la del proceso 2 o del proceso 3 en forma alternada, es decir:
(P1, P2), (P1, P3), (P1, P2), (P1, P3)
Indicar claramente el tipo y el valor de inicial de los semáforos. Recuerde que no puede utilizar variables globales para la sincronización

```
//SEMAFOROS BINARIOS
Semaforo_p1 = 1;
Semaforo_p2 = 0;
Semafoto_p3 = 0;

Proceso_p1
While(1){
        Wait(semáforo_p1);
                Tarea P1;
        Signal(semáforo_p2);
        Wait(semáforo_p1);
                Tarea P1;
        Signal(semáforo_p3);
}
Proceso_p2
While(1){
        Wait(semáforo_p2);
                Tarea P2;
        Signal(semáforo_p1);
}
Proceso_p3
While(1){
        Wait(semáforo_p3);
                Tarea P3;
        Signal(semáforo_p1);
}
```

NOTA= esta solución no es tan eficiente porque se duplica la sección critica, pero es válida ósea te la toman bien.

La otra opción "***EFICIENTE***" sería esta:



2) Cada uno de los eventos enumerados a continuación interrumpe el flujo de ejecución. Responda las siguientes preguntas sobre cada evento:

a. (¿El evento es sincrónico o asincrónico?)
b. ¿Qué estado debe salvarse y restaurarse?
c. ¿Quién (persona que llama, destinatario de la llamada, etc.) debe guardar y restaurar este estado?
Su respuesta debe analizar los componentes genéricos de las maquinas (por ejemplo, "el puntero de instrucción") en lugar de estar orientado hacia una arquitectura de maquina en particular

a. Llamada a procedimiento
(**Sincrónico**, Deben salvarse ***registros, variables locales, dirección de retorno (IP), parámetros que se le pasan***, etc. **Guarda el llamador**)

b. Llamada al sistema
(**Sincrónico**, Debe guardar **el estado PCB proceso llamador**, lo guarda el kernel (**Destinatario**))

c. Cambio de proceso causado por un intervalo de tiempo que expira.
(**Asincrónico** (nunca sabe cuándo lo van a interrumpir), Debe guardar **todo el PCB**, **lo guarda el manejador de eventos** (que no es ni el llamador ni el destinatario))

Flor

Ejercicio 1:

3) en un multiprocesador, ¿cual/es de los siguientes mecanismos de exclusión mutua son apropiados?

a. inhibición de interrupciones
b. variable de condición
c. monitores
d. spinlock
e. paso de mensajes
f. todos estos mecanismos se pueden utilizar normalmente en un ordenador multiprocesador

5) en lo referente al concepto de multiprogramación ¿cual/es de las siguientes afirmaciones son ciertas?

a. solamente tiene sentido en un sistema de tiempo compartido
b. los sistemas de tiempo real suelen ser multiprogramados
c. no tiene sentido en las arquitecturas paralelas (multiprogramación soporta el paralelismo)
d. su utilización depende de la política de planificación del sistema (se usa para todas)
e. la multiprogramación requiere considerar aspectos de protección en el sistema

Ejercicio 2:

1) ¿Cuáles de los siguientes tipos de sistemas les afecta más el criterio “tiempo de retorno” (o turnaround time) a la hora de diseñar su planificador?
    a. sistemas de tiempo real
    b. sistemas de tiempo compartido
    c. sistemas paralelos
    d. sistemas batch (libro tanem 164)
    e. a todos por igual

2) en cuanto a los procesos y los archivos es cierto que:

a. el resultado de la escritura concurrente de dos procesos no cooperantes sobre un archivo es impredecible ya que depende de la política de planificación de procesos que aplique el sistema

b. el campo “contador de aperturas” de un archivo en la tabla de

correctas: e, a

3) en cuanto a la evolución histórica de los SO, indique qué afirmación es cierta:

a. el sistema off-line surgió para acelerar las operaciones de entrada/salida en sistemas operativos basados en spool → tarjetas perforadas
b. el secuenciador automático de trabajos no evitaba el tiempo que la cpu pasaba ociosa mientras el proceso hacia una operación de e/s → damian V, sebastian F
c. la técnica de spooling se usa actualmente únicamente en sistemas batch
d. los sistemas de tiempo compartido no dieron origen a los sistemas multiprogramados

Ejercicio 4:

2. Se ha visto en teoría un conjunto de políticas de planificación y un conjunto de mecanismos para implementar estas políticas. Los objetivos de un algoritmo de planificación son:

a. maximizar la terminación de tareas por unidad de tiempo. throw put
b. maximizar el número de usuarios interactivos recibiendo respuesta en un tiempo aceptable. tiempo compartido
c. ser predecible.
d. minimizar la sobrecarga
e. balancear la utilización de recursos
f. balancear respuesta y utilización
g. evitar tareas pospuestas indefinidamente
h. ordenar las tareas en función de su importancia y requerimientos de eficiencia
i. favorecer a procesos que mantienen recursos clave no apropiables
j. favorecer procesos con comportamiento deseable (bajos page faul, por ej)
k. reducir el servicio cuando la carga resulta excesiva

cual/es de los objetivos anteriores se está teniendo o no en cuenta cuando: explique

i. los procesos compitiendo por la cpu se complementan en sus requerimientos de recursos → g) al estar aplicando envejecimiento

ii. un sistema de control de procesos de tiempo real monitoreando a una estación de servicios requiere rápida respuesta → c) el planificador intenta también cumplir con ser predecible, para un conjunto de procesos con prioridades asignadas
ordenar los requerimientos - eficiencia

iii. un proceso devuelve frecuentemente la cpu antes de que su quantum expire.
no h) en este caso, se debería desalojar al proceso de menor importancia para poder liberar el recurso que está alocado por el mismo. así se podría ejecutar el proceso que arribó. no al revés, ya que cuando hay una competencia entre dos procesos de diferentes prioridades el proceso de menor prioridad es el que se ejecuta ???????

Extra:

1) en un sistema con algoritmo de planificación por prioridades apropiativo con colas multinivel, donde cada cola de prioridad está gestionada mediante el algoritmo FIFO.
    a. cuando un proceso pasa al estado de Ejecutando, permanecerá en este estado hasta que se bloquee en espera de una E/S o termine su ejecución ⟶ F, es apropiativo y si llega un proceso a una cola con mayor prioridad debería apropiarse para que se ejecute el recién llegado
    b. El sistema operativo no desbloquea a un proceso (pasa de esperando a listo) si existe otro proceso de mayor prioridad que también tiene que ser bloqueado. ⟶ F? creo que la prioridad solo afecta a listo -> ejecutando y viceversa
    c. este algoritmo de planificación evita la inanición -> F, al ser por prioridad si constantemente llegan procesos de mayor prioridad, aquellos con menor prioridad deberán esperar continuamente
    d. ninguna, V

3) Dado el siguiente programa que maneja dos semáforos, uno B, binario, y otro C, de cuenta:

```
for (i=1;i<=N;i++) {
    wait(B); wait(C);
    { más código }
    signal(B); signal(B);
    signal(C); signal(C); }
```

a) El valor final de B es N+1 y el de C también.

b) El valor final de B es 1 y el de C también.

c) El programa no acaba nunca.

d) Ninguna es correcta.

Según Sebastian

b es binario y c de cuenta, cuando el semáforo es binario el valor va a ser 0 o 1 aunque haga dos signal(b)

d. es la respuesta correcta

4) Un proceso que está bloqueado (en espera) puede salir de tal estado si:

a. Algún proceso realiza cualquier llamada al sistema.

b. Se produce una interrupción.

c. Cualquier proceso independiente finaliza su ejecución.

d. Cualquiera de las anteriores.

a) Falso, por ej, un proceso realiza fopen() y no afecta en nada.
b) Verdadero, por ej, el proceso termina de realizar una E/S y se realiza una interrupción.
c) falso
d) Falso.

1.- Al resolver algunos problemas con varios hilos (threads), a veces es útil que todos los hilos se encuentren en un lugar del código. Esto normalmente se hace con una operación llamada *barrera*. Funciona de modo que cuando los hilos llaman a la función Barrera (), ninguno regresa hasta que todos los hilos han llamado a la función. Por ejemplo, puede encontrar un código similar a:

```
{
Hilosejecutando(); // Los hilos pueden terminar en diferentes momentos
Barrera(ThreadID());
HilosejecutandoII(); // Los hilos deben comenzar aquí juntos
}
```

Su trabajo consiste en escribir la función Barrera () utilizando solo semáforos. No se le permite utilizar variables compartidas que no sean los semáforos. Asegúrese de mostrar sus valores iniciales para el semáforo. Puede asumir que hay una constante global NUM_HILOS que indica la cantidad de hilos en el sistema. Los ID de hilos son números enteros que comienzan en 0 y van a NUM_HILOS-1. NUM_HILOS es menor que 100. Para simplificar las cosas, puede asumir que Barrera () solo se llama una vez y que todos los hilos participan en la barrera.

```
sem_t barrera, contadorDesloqueados;
sem_init(&contadorDesloqueados,0, NUM_HILOS);
sem_init(&barrera,0, 0);
Barrera(int id){
        If(sem_try_wait(&contadorDesloqueados)<0){ // Si soy el ultimo hilo
                for(int i=0;i<NUMHILOS;i++){// Libero a todos los hilos en la barrera
                        sem_post(&barrera);
                }
        }else{// Si no soy el ultimo hilo
                Sem_wait((&barrera);//Me quedo en la barrera
        }
}
```

Una de las mejores respuestas → segun Federico