Departamento de Arquitectura

Super parcial de Sistemas Operativos

12 de Diciembre de 2020

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del parcial.

Formato:

- Indique su nombre completo y número de cédula en cada hoja (no se corregirán las hojas sin nombre). Numere todas las hojas e indique la cantidad total de hojas en la primera.
- Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva.
- Si se entregan varias versiones de un problema solo se corregirá la primera de ellas.

Dudas:

- Sólo se contestarán dudas de letra.
- No se aceptarán dudas en los últimos 15 minutos del parcial.

Material:

- El parcial es **SIN** material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del parcial, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.

Finalización:

- El parcial dura **3 horas**.
- Al momento de finalizar el parcial no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del parcial.

Importante: Debe justificar todas las respuestas. Siempre.

Problema 1 (10 pts)

(a) (2 pts) Describa brevemente:
   i. ¿En qué consiste Discretionary Access Control y Mandatory Access Control?
   ii. ¿Qué tipo de mecanismo de control de acceso utiliza Linux?. Justifique su respuesta.

(b) (3 pts) En Unix/Linux.
   i. ¿Qué es un soft link y un hard link? ¿cómo se representa en las estructuras del sistema de archivos?
   ii. Supongamos que tenemos un archivo llamado source.file y creamos un hardlink que apunta a este archivo llamado hardlink.file. Una vez creado el archivo hardlink.file eliminamos el archivo source.file y luego accedemos a hardlink.file. ¿Qué sucede? Justifique su respuesta.

(c) (3 pts) i. Enumere 3 instrucciones privilegiadas. Describa para qué propósito deben existir este tipo de instrucciones en un Sistema Operativo.
   ii. ¿Bajo qué condiciones pueden ejecutarse este tipo de instrucciones? Justifique su respuesta

(d) (2 pts) Considere los siguientes componentes de un proceso: data, heap, stack, variables en registros del CPU. Cuáles de estos componentes son compartidos entre:
   i. Un proceso y sus hijos.
   ii. Los threads de un mismo proceso

Problema 2 (20 pts)

Se tiene un sistema operativo multiprogramado en el cual se dispone de 2 procesadores. El planificador del sistema operativo utiliza una estrategia de planificación Round-Robin con quantum de $4ms$. Este sistema maneja hilos con modelo $M \times 1$, utilizando una planificación Round-Robin con quantum de $2ms$ a nivel de usuario.

R1	R2	R3
Ejecuta 2ms Bloquea 2ms Ejecuta 2ms	Ejecuta 4ms Bloquea 2ms Ejecuta 4ms Bloquea 2ms	Ejecuta 4ms Bloquea 4ms Ejecuta 8ms Bloquea 2ms Ejecuta 2ms

Se tiene que el proceso P1 cuenta con 2 hilos (P1H1, P1H2) que ejecutan concurrentemente de forma que un hilo ejecuta el código de R1 y el otro R2. En cambio, P2 y P3 cuentan con un único hilo de ejecución y ejecutan R2 y R3 respectivamente. En el instante de tiempo inicial ($t = 0$) la cola de listos contiene a P1, P2 y P3. En todos los casos, los empates se resuelven primero priorizando el que tenga mayor tiempo de cómputo restante y luego por menor identificador, es decir, P_i tendrá mayor prioridad que P_j si $i < j$.

Se pide:

(a) (18 pts) Realice un diagrama de planificación (tiempo vs procesos/hilos), comenzando en el tiempo $t = 0$, indicando el estado de cada uno de los procesos/hilos y su posición en cada una de las colas.

(b) (2 pts) Calcule y defina porcentaje de uso de CPU, tiempo de retorno y tiempo de espera de cada proceso/hilo.

Solución:

(a) L': Cola de listos a nivel de usuario.

L: Cola de listos a nivel de sistema.

Ex: Ejecutando en el procesador x.

B: Bloqueado.

T: Terminado.

	2	4	6	8	10	12	14	16	18	20	22	24	26
P1H1	L'1	E1	B	L'1	L'1	E2	T						
P1H2	E1	L'1	L'1	E2	B	L'1	E2	L'1	E1	B	T		
P1	E1	E1	B	E2	B	E2	E2	L1	E1	B	T		
P2	L1	L1	E1	E1	B	L1	E1	E1	B	T			
P3	E2	E2	B	B	E1	E1	L1	E2	E2	B	E1	T	

(b) Uso de CPU: Porcentaje de tiempo que la CPU está siendo utilizada para procesar alguna tarea del usuario o del sistema.

Porcentaje de uso CPU1: (20 ms / 22 ms) * 100 = 90.9%

Porcentaje de uso CPU2: (14 ms / 22 ms) * 100 = 63.6%

Tiempo de retorno: Intervalo de tiempo desde que un proceso/hilo es cargado hasta que este finaliza su ejecución.

Tiempo de retorno P1H1: 12 ms – 0 ms = 12 ms

Tiempo de retorno P1H2: 20 ms – 0 ms = 20 ms

Tiempo de retorno P1: 20 ms – 0 ms = 20 ms

Tiempo de retorno P2: 18 ms – 0 ms = 18 ms

Tiempo de retorno P3: 22 ms – 0 ms = 22 ms

Tiempo de espera: Tiempo total que durante el cual el proceso/hilo se encuentra en la cola de listos.

Tiempo de espera P1H1: 6 ms

Tiempo de espera P1H2: 8 ms

Tiempo de espera P1: 2 ms

Tiempo de espera P2: 6 ms

Tiempo de espera P3: 2 ms

Problema 3 (20 pts)

Se tiene un gestor de memoria de un sistema operativo que utiliza memoria virtual con paginación bajo demanda con las siguientes características:

- Estructura jerárquica, de dos niveles de paginación.
- Páginas de 1024 bytes.
- Tablas que ocupan una página completa y con entradas de 4 bytes.

Se pide:

(a) (3 pts) ¿Cuántos bits necesitan las direcciones?

(b) (3 pts) ¿Cómo se distribuyen los bits en la dirección? Indique para qué se utilizan. Incluya un esquema gráfico para una dirección particular.

(c) (3 pts) Indique el tamaño máximo de memoria virtual direccionable por proceso.

(d) (4 pts) En caso de trabajar con procesos que utilicen como máximo 256KB, ¿Qué cambio debería hacer en la política de paginación para ganar en desempeño? Justifique.

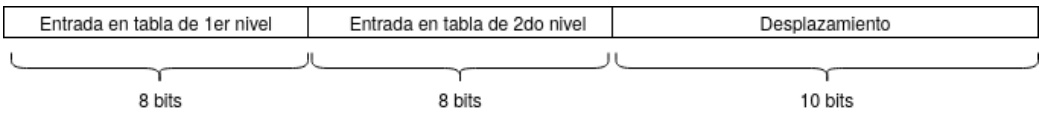
(e) (5 pts) Si el sistema posee solamente 2MB de memoria RAM física, ¿Puede ejecutar 5 procesos, que gestionan la memoria como se describe anteriormente y potencialmente pueden utilizar toda la memoria direccionable, al mismo tiempo? Justifique.

(f) (2 pts) Describa brevemente la técnica de reemplazo de Segunda Oportunidad. Modifique dicha técnica para contemplar un sistema donde los procesos tienen como característica que sus páginas son accedidas como máximo en dos oportunidades. Justifque.

Solución:

(a) Se tienen tablas de 1024 bytes. Cada tabla tiene 1024/4 = $2^{10}/2^2$ = 256 entradas, por lo que se necesitan 8 bits para direccionar la tabla de paginas de 1er nivel, y 8 bits para direccionar la tabla de paginas de 2do nivel. Luego, como los frames son de 1024 bytes, se necesitan 10 bits para direccionar un byte adentro de un frame. En total se precisan 8 + 8 + 10 = 26 bits.

(b)



(c) El tamaño máximo direccionable por un proceso es 2^{26} bytes = 2^6 MB = 64 MB

(d) Se podría eliminar el 2do nivel de paginación y usar uno solo. De esta forma las direcciones serian de 18 bits y el maximo direccionable por proceso seria de 2^{18} = 2^8 KB = 256 KB como se pide. A su vez, utilizar un nivel menos disminuye en 1 la cantidad de lecturas a memoria principal (o a TLB) a la hora de acceder a una dirección, ganando en performance.

(e) Sí, siempre y cuando el espacio en memoria de swap sea suficiente para los 5 procesos. Utilizando tecnicas de reemplazo, se pueden cargar frames a memoria principal a demanda y bajarlos a disco duro cuando no estan siendo utilizados.

(f) El algoritmo de Segunda Oportunidad funciona como el FIFO, pero dando una chance extra a aquellas paginas que hayan sido referenciadas luego de ser cargadas a memoria. Esto se realiza utilizando un bit de referencia en la tabla de páginas. Se podría modificar el algoritmo para que en vez de prender el bit de referencia si la pagina es accedida luego de cargada en memoria, que baje esta pagina a memoria de swap, ya que sabemos que esta pagina ya fue accedida dos veces y no volvera a serlo en el futuro.

Problema 4 (35 pts)

Se desea modelar una cervecería que expende 15 tipos de cervezas diferentes. Para ello dispone de una única canilla por tipo de cerveza y 5 vendedores.

Los clientes que desean comprar cerveza deberán esperar en una única fila por orden de llegada. El primer vendedor libre atenderá al primero de la fila. La longitud de la fila no está acotada.

Se pide:

(a) (35 pts) Modelar en Ada las tareas Cliente y Vendedor.

Se dispone de los siguientes procedimientos:

- `quiero_cerveza(): integer(1..15)`: Invocada por los clientes para saber el tipo de cerveza que quieren tomar.
- `servir(tipo_cerveza: integer(1..15))` : Invocada por los vendedores para servir la cerveza pedida por el cliente.
- `pagar_cobrar()` : Deberá ser ejecutada por el cliente o el vendedor para que se realice el pago por la cerveza (solamente debe ser llamada por una de las tareas).
- `tomar()` : Ejecutada por el cliente una vez que le sirvieron la cerveza y se efectuó el pago.

Nota: Se pueden utilizar tareas auxiliares.

Solución:

```
task type cliente;
task body cliente is
tipo_cerveza:integer;
nro_vendedor:integer;
Begin
    tipo_cerveza = quiero_cerveza();
    manejador_vendedores.requerir_atencion(nro_vendedor);
    vendedores[nro_vendedor].pedir(tipo_cerveza);
    tomar();
end cliente;

task type vendedor is
    entry set_id(identificador:integer);
end vendedor;
task body vendedor is
id:integer;
Begin
```

```
    accept set_id(identificador:integer) do
        id:=identificador;
    end aceptar_id;
    while true loop
        manejador_vendedores.vendedor_libre(id);
        accept pedir(tipo_cerveza:integer) do
            canillas[tipo_cerveza].adquirir_canilla();
            servir(tipo_cerveza);
            canillas[tipo_cerveza].dejar_canilla();
            pagar_cobrar();
        end pedir;
    end loop;
end vendedor;

task manejador_vendedores is
    entry requerir_atencion(out nro_vendedor:integer);
    entry vendedor_libre(id:integer);
Begin
    while true loop
        accept requerir_atencion(out nro_vendedor:integer) do
            accept vendedor_libre(id:integer) do
                nro_vendedor := id;
            end vendedor_libre;
        end requerir_atencion;
    end loop;

end manejador_vendedores;

task type canilla is
    entry adquirir_canilla();
    entry dejar_canilla();
end canilla;
task body canilla is
    while true loop
        accept adquirir_canilla();
        accept dejar_canilla();
    end loop;
end canilla;


vendedores: array[1..5] of vendedor;
canillas: array[1..15] of canilla;

Begin
    for id in 1..5  loop
        vendedores[id].set_id(id);
    end loop;
end;
```