Departamento de Arquitectura

Examen de Sistemas Operativos

13 de febrero de 2019

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

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 30 minutos del examen.

Material

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

Aprobación

- Para aprobar el examen se debe tener un mínimo de 60 puntos.

Finalización

- El examen dura 4 horas.
- Al momento de finalizar el examen 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 examen.

Pregunta 1 (32 pts)

(a) (4 pts) En qué se diferencia un sistema de tiempo real hard y uno soft.

(b) (4 pts) En los sistemas operativos modernos, ¿de qué forma se garantiza que todos los accesos a los dispositivos, por parte de un proceso de usuario, se deban hacer a través del sistema operativo?

(c) (4 pts) Explique qué desventaja particular tienen los hilos implementados a nivel de usuario en un sistema con paginación bajo demanda pura.

(d) (4 pts) En un sistema con un modelo de hilos 1x1, describa los pasos a seguir al realizar un cambio de contexto. Discuta según si se trata de hilos del mismo proceso o de diferentes procesos.

(e) (4 pts) En un sistema con memoria virtual, describa el algoritmo de reemplazo de segunda chance (second-chance algorithm).

(f) (4 pts) Defina y compare los métodos de E/S programada y por interrupciones. Indique ventajas y desventajas de cada método.

(g) (4 pts) Explique muy brevemente las 4 condiciones necesarias para la existencia del bloqueo mutuo (deadlock).

(h) (4 pts) Defina y compare sistemas operativos simétricos y asimétricos.

Pregunta 2 (33 pts)

Se tiene un sistema con un único procesador sobre el que ejecuta un sistema operativo con un planificador Round Robin con un cuanto de 10 ms y un modelo de hilos 1x1. Inicialmente en el sistema no se tiene ningún proceso de usuario ejecutando y en tiempo t=0 se lanza la ejecución del siguiente programa de usuario:

```
begin
    Ejecuta por 10 ms
    create_thread(proc1)
    A = 24
    A = A + A
    Espera E/S por 10 ms
    A = A - 1
    Ejecuta por 20 ms
    print A
end
```

```
procedure proc1()
begin
    Ejecuta por 5 ms
    if (A = 24) then
        A = 10
        fork()
        A = A + 1
        Espera E/S por 10 ms
    else
        Ejecuta por 15 ms
    end if
    print A
end
```

Se sabe que la operacion fork() es estilo Unix y al igual que en Unix, luego de un fork() el proceso hijo siempre tiene un único hilo de ejecución. La operación create_thread() crea un nuevo hilo de ejecución y comienza su ejecución en el procedimiento recibido como argumento. Salvo que se indique explícitamente, el tiempo de ejecución de todas las funciones es de 5 ms (fork, create_thread, if, print, asignación, suma, etc.). Se asume que todos las funciones del sistema operativo requieren un tiempo despreciable para su ejecución. Por último, se sabe que al ingresar a la cola de listos, siempre se da prioridad a los hilos que hace menos tiempo que ingresaron al sistema.

Se pide:

(a) i. (15 pts) Realice un diagrama de planificación (tiempo vs hilos), comenzando en el tiempo t=0, indicando el estado de cada uno de los hilos, la cola de listos y el valor de la variable A en cada intervalo de tiempo.

ii. (3 pts) Calcule y defina el tiempo de espera de cada proceso.

(b) Sabiendo además que cada proceso requiere de 2050 KiB para almacenar su codigo, datos y heap, mientras que para la pila requiere de 8 KiB de espacio de memoria. El sistema operativo utiliza un gestor de memoria implementada con memoria virtual utilizando un modelo de paginación bajo demanda con las siguientes características:

- Direcciones virtuales de 48 bits.
- Tres niveles de paginación.
- Entradas de tabla de páginas de 16 bytes y páginas de 512 bytes.
- Las tablas de segundo y tercer nivel ocupan una página de memoria.

i. (5 pts) Determine como se distribuyen los bits cada dirección indicando para que se utilizan.

ii. (10 pts) ¿Cuántas páginas de 2do y 3er nivel son necesarias para almacenar cada proceso de usuario en memoria?

Solución: Diagrama tiempo vs hilos

P1-H1	E	E	E	E	L1	L1	E	B	B	L2	L1	E	E	L2	L1	E	E	E	E	T	
P1-H2				L1	E	E	L1	E	E	L1	E	B	B	L1	E	T	T	T	T	T	
P2-H1										E	B	B	L1	E	T	T	T	T	T	T	
Tiempo	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	
Variable				A=24			A=48	A=10		A'=11	A=11	A=10									

Referencia:

- P*n*-H*m* = Indica el hilo *m* del proceso *n*.
- E = El hilo se encuentra en estado ejecutando.
- L*i* = El hilo se encuentra en estado listo. El indice *i* indica la posición en la cola de procesos listos.
- B = El hilo se encuentra bloqueado a la espera de un evento de E/S.
- A = valor de la variable A en el proceso P1.
- A' = valor de la variable A en el proceso P2.

Tiempo de espera

Se acepta como correcto el cálculo de tiempo de espera a nivel de hilo y a nivel de proceso.

A nivel de hilo:

P1-H1 = $(30 - 20) + (55 - 45) + (75 - 65) = 30ms$

P1-H2 = $(20 - 15) + (35 - 30) + (50 - 45) + (70 - 65) = 20ms$

P2-H1 = $(65 - 60) = 5ms$

A nivel de proceso:

P1 = $50 - 45 + 70 - 65 = 10ms$

P2 = $65 - 60 = 5ms$

Tamaño de páginas

- Como las páginas son de 512 bytes (2^9 bytes), se necesita 9 bits para el desplazamiento.
- Además, como las entradas de las tablas de 2do y 3er nivel ocupan 16 bytes y la tabla ocupa 2^9 bytes, se tiene que hay $2^9/2^4 = 2^5$ entradas por tabla de 2do y 3er nivel. Es decir que se necesitan 5 bits para direccionar cada una de estas tablas.
- Finalmente, quedan $48 - 5 - 5 - 9 = 29$ bits para direccionar una entrada de primer nivel.

Es decir:

—— 29 bits 1er nivel ——— 5 bits 2do nivel ——— 5 bits 3er nivel ——— 9 bits offset ——

Uso de memoria

Nota:

- Se acepta como correcto si se calcula la suma de la cantidad de tablas de páginas de los procesos P1 y P2. En esta solución solamente se calcula la cantidad de páginas de un proceso, da igual cuál porque son idénticos.
- Se puede interpretar que un proceso requiere 8KiB de espacio de pila para todos sus hilos u 8 KiB para cada uno de sus hilos. Ambas soluciones se aceptan como correctas. Aquí se muestra la solución considerando que un proceso requiere en total 8 KiB de pila para todos sus hilos.

El mapeo de memoria que requiere cada proceso es de 2050KiB desde la dirección virtual 0 hacia las crecientes y 8KiB desde las más altas hacia abajo. Cada tabla de tercer nivel mapea 16KiB, por lo que se requiere:

2050KiB/16KiB= $(2048 + 2)/16 = (2^{11} + 2^1)/2^4 = 2^7 + 2^1/2^4 < 128 + 1$

En definitiva, para el código, datos y heap se requieren 128 tablas de tercer nivel completas y 1 tabla incompleta. Además se requiere de una tabla adicional para la pila. Entonces se requiere de 130 tablas de tercer nivel.

Por otro lado, cada tabla de 2do nivel es capaz de direccionar 2^5 tablas de 3er nivel. Entonces necesitamos $129/2^5 = 2^7/2^5 + 2^0/2^5 < 2^2 + 1 = 5$ tablas de 2do nivel para las 129 tablas de tercer nivel correspondientes al código, datos y heap. Además se precisa una tabla de 2do nivel adicional para la pila. En total 6 tablas de 2do nivel.

Pregunta 3 (35 pts)

Se desea modelar el funcionamiento de una mesa de votación y su respectivo cuarto secreto para las elecciones nacionales.

Cuando un votante llega a la mesa deberá presentar su credencial la cual será validada por la mesa. Luego de ser autorizado el votante tomará un sobre de la pila y se dirigirá al cuarto secreto. Si el cuarto está ocupado deberá esperar a que se libere.

Una vez en el cuarto secreto, el votante pondrá la hoja de votación en el sobre. Cuando esta tarea finalice retornará a la mesa para depositar el voto. La mesa puede atender más votantes mientras el cuarto secreto esté ocupado. En el momento que el votante salga del cuarto secreto a depositar su voto puede tener que esperar a que la mesa termine la atención de otro votante que se encuentra en la mesa haciendo los trámites previos a la entrada al cuarto secreto, pero tendrá prioridad frente a los demás votantes y delegados esperando que la mesa los atienda.

Cada cierto tiempo, los delegados de los diferentes partidos llegan a la mesa de votación y solicitan acceder al cuarto secreto para reponer listas. Un delegado partidario debe presentar a la mesa su identificación de delegado. Los delegados harán cola para ser atendidos por la mesa junto a los votantes esperando para presentar su credencial. Ambos serán atendidos por orden de llegada.

En ningún momento podrán haber colados en las filas de la mesa de votación y del cuarto secreto. Se puede asumir que nunca se acaban las listas en el cuarto secreto. Si el documento que presenta no es válido la persona se retira inmediatamente.

Se pide: modelar utilizando mailboxes, los procesos Mesa, Votante y Delegado.

Se dispone de los siguientes procedimientos auxiliares:

`obtener_documento():Documento`
: Ejecutado por los votantes y delegados para obtener su documento de identificación

`validar_documento(Documento)`**:boolean**
: Ejecutado por la mesa para validar la credencial del votante o la identificación del delegado

`tomar_sobre():Sobre`
: Ejecutado por los votantes para obtener un sobre

`poner_lista(Sobre):Sobre`
: Ejecutado por los votantes para poner una lista en el sobre

`votar(Sobre)`
: Ejecutado por los votantes para poner el sobre en la urna

`recargar()`
: Ejecutado por los delegados para recargar las listas de votación

Solución: Se usarán mailboxes infinitos (send no bloqueante) con receive bloqueante.

```
Mailbox mb_atencion of NIL;
Mailbox mb_espero_mesa of NIL;
Mailbox mb_cuarto_secreto of NIL;
Mailbox mb_espero_votar of NIL;
Mailbox mb_fin_votar of NIL;
Mailbox mb_mutex of NIL;
Mailbox mb_espero_validacion of boolean;
Mailbox mb_fila_prioridad of integer;
Mailbox mb_identificador_mesa of Documento;
```

```
procedure Votante()
var ok: boolean;
    sobre: Sobre;
    esperando: integer;
    documento: Documento;

begin
    documento = obtener_documento();
    mb_atencion.send(NIL);
    mb_espero_mesa.receive(); // cola de espera

    mb_mutex.receive();
    mb_identificador_mesa.send(documento);
    mb_espero_validacion.receive(ok)
    mb_mutex.send(NIL);

    if ok then
        sobre := tomar_sobre();

        mb_cuarto_secreto.receive(); // cola de espera
        sobre := poner_lista(sobre);
        mb_cuarto_secreto.send(NIL);

        mb_fila_prioridad.receive(esperando);
        mb_fila_prioridad.send(esperando + 1);

        mb_atencion.send(NIL);
        mb_espero_votar.receive(); // cola de espera
        votar(sobre);
        mb_fin_votar.send();
    end if
end


procedure Delegado()
var ok: boolean;
    documento: Documento;

begin
    while(true) do
        documento = obtener_documento();
        mb_atencion.send(NIL);
        mb_espero_mesa.receive(); // cola de espera

        mb_mutex.receive();
        mb_identificador_mesa.send(documento);
        mb_espero_validacion.receive(ok)
        mb_mutex.send(NIL);

        if ok then
            mb_cuarto_secreto.receive(); // cola de espera
```

```
            recargar();
            mb_cuarto_secreto.send(NIL);
        end if
    end while
end


procedure Mesa()
var esperando: integer;
    documento: Documento;

begin
    while(true) do
        mb_atencion.receive();
        mb_fila_prioridad.receive(esperando);
        if esperando = 0 then
            mb_fila_prioridad.send(esperando);
            mb_espero_mesa.send();
            mb_identificador_mesa.receive(documento);
            mb_espero_validacion.send(validar_documento(documento));
        else
            mb_fila_prioridad.send(esperando - 1);
            mb_espero_votar.send();
            mb_fin_votar.receive();
        end if
    end while
end


begin
    mb_cuarto_secreto.send(NIL);
    mb_mutex.send(NIL);

    cobegin
        Votante();
        ...
        Votante();
        Delegado();
        ...
        Delegado();
        Mesa();
    coend
end
```

Nota: se acepta como correcto que el votante mantenga la mesa bloqueada mientras ejecuta la función tomar_sobre().