Departamento de Arquitectura

Examen de Sistemas Operativos

23 de febrero de 2023

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 **3 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.

Problema 1 (32 pts)

(a) Se tiene un sistema operativo multiprogramado para el cual se dispone de 2 procesadores. El planificador del sistema operativo utiliza una estrategia de planificación Round-Robin con quantum de 10 ms. Este sistema maneja hilos con modelo **Mx1**, utilizando una planificación FCFS (ante empates se ejecuta primero el hilo con identificador **mayor**).

Se tienen las siguientes rutinas:

R1	**R2**	**R3**
Ejecuta 10ms	Ejecuta 5ms	Ejecuta 15ms
Bloquea 5ms	Bloquea 10ms	Bloquea 10ms
Ejecuta 10ms	Ejecuta 20ms	Ejecuta 10ms
Bloquea 5ms		
Ejecuta 10ms		

Se tiene que **P1** cuenta con 2 hilos que ejecutan concurrentemente de forma que cada hilo ejecuta todo el código de la rutina **R1**. En cambio, **P2** y **P3** cuentan con un solo hilo de ejecución en los que ejecutan las rutinas **R2** y **R3** respectivamente. En el instante de tiempo inicial ($t = 0$) la cola de listos contiene a **P3**, **P1** y **P2** (en este orden).

Se pide:

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 y su posición en cada una de las colas.

II. (5 pts) Calcule y defina porcentaje de uso de cada CPU, tiempo de retorno y tiempo de espera de cada proceso.

Solución:

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
P1H1	L'1	L'1	L'1	E1	E1	B	L'2	L'1	L'1	L'1	E1	E1	B	L'1	L'1	E1	E1	T
P1H2	E2	E2	B	L'1	L'1	L'1	L'1	E2	E2	B	L'1	L'1	L'1	E1	E1	T		
P1	E2	E2	B	E1	E1	B	L1	E2	E2	B	E1	E1	B	E1	E1	E1	E1	T
P2	L1	L1	E2	B	B	E1	E1	E1	E1	T								
P3	E1	E1	E1	B	B	E2	E2	T										

Porcentaje de uso de CPU: El tiempo que cada CPU está siendo utilizada para cómputo sobre el tiempo total.
$\text{UsoCPU}_1 = 75/85 = 88\,\%$
$\text{UsoCPU}_2 = 35/85 = 41\,\%$

Tiempo de retorno: Es el intervalo de tiempo entre que un proceso es cargado hasta que termina de ejecutar.
$\text{Retorno}_1 = 85 - 0 = 85$
$\text{Retorno}_2 = 45 - 0 = 45$
$\text{Retorno}_3 = 35 - 0 = 35$

Tiempo de espera: Es el tiempo que un proceso pasa en la cola de listos.
$\text{Espera}_1 = 35 - 30 = 5\text{ms}$
$\text{Espera}_2 = 10 - 0 = 10\text{ms}$
$\text{Espera}_3 = 0\text{ms}$

(b) (4 pts) Describa tres métodos para la asignación de espacio de disco. Mencione una desventaja de cada uno de ellos.

(c) (4 pts) ¿Por qué en el PCB (Process Control Block) se reserva lugar para los registros de la CPU?

(d) (4 pts) Describa brevemente qué es un Hipervisor de Tipo 1.

Problema 2 (28 pts)

Se tiene un sistema operativo con una arquitectura de memoria que utiliza un modelo de paginación bajo demanda para la gestión de la memoria con las siguientes características:

- Las direcciones virtuales son de 52 bits.
- Se tiene un esquema con tablas de página multinivel de tres niveles.
- Todas las tablas tienen el mismo tamaño y ocupan una página completa cada una.
- Las entradas en las tablas de páginas son de 128 bits (16 bytes).
- Cada tabla de tercer nivel direcciona, cuando está completa, 256MiB (2^{28} bytes).

Se pide (justifique cada respuesta):

(a) (2 pts) ¿Cuál es el tamaño total en bytes del espacio de direccionamiento virtual?

(b) (6 pts) ¿Cuál es el tamaño de las páginas en el sistema?

(c) (3 pts) ¿Cuántos bits son necesarios para el desplazamiento?

(d) (4 pts) ¿Cuántas entradas tienen las tablas de páginas?

(e) (4 pts) ¿Cuántos bits de la dirección virtual son utilizados para determinar la entrada en cada tabla de páginas?

(f) (3 pts) Realice un diagrama que muestre como se realiza la traducción de una dirección virtual en el esquema propuesto.

(g) (6 pts) ¿Cuántas tablas de páginas en total son necesarias para direccionar un proceso de 8TiB repartidos como 512MiB de stack y el resto de código y datos?

Solución:

(a) 2^{52}

(b) La tabla de tercer nivel completa ocupa una página y direcciona 256MB. Sea P el tamaño de una página, como las entradas ocupan 16B, tiene $P/16$ entradas, cada apuntando a una página de P bytes. Entonces $(P/2^4) * P = 2^{28} => P = (2^{32})^{1/2} => P = 2^{16}$

(c) 16 bits.

(d) El tamaño de la página dividido el tamaño de una entrada: $2^{16}/2^4 = 2^{12}$

(e) 12 bits.

(f) Ver teórico.

(g) Dado que cada tabla de tercer nivel direcciona 2^{28} bytes y las tablas cuentan con 2^{12} entradas una tabla de segundo nivel direcciona 2^{40} B = 1 TiB.
Para direccionar el stack (2^{29} bytes) serán necesarias $2^{29}/2^{28} = 2$ tablas de tercer nivel.
Finalmente se requiere direccionar el código y los datos (8TiB - 512MiB). Los primeros 7TiB requerirán 7 tablas de segundo nivel (con todas sus tablas de tercer nivel llenas). El último TiB también requiere una tabla de segundo nivel, donde todas menos las últimas 2 entradas estarán en uso. Es decir, se usarán $2^{12}-2$ tablas de 3er nivel. Como el stack y el código comienzan de extremos opuestos de la memoria las tablas de tercer nivel sobrantes no podrán agruparse y serán necesarias dos tablas extra de segundo nivel. Sumando, obtenemos que se requiere una tabla de primer nivel, 9 tablas de segundo nivel y $8 * 2^{12}$ tablas de tercer nivel.

Problema 3 (40 pts)

Se desea modelar una empresa de logística que dispone de un depósito de almacenamiento que alquila a clientes por metro cuadrado. El depósito cuenta con 1000 m^2 pero por razones administrativas la empresa no puede manejar más de 50 clientes en total.

Cuando llega un cliente el mismo indicará los m^2 que necesita y se le dirá si puede guardar la mercadería. Si no hay espacio en el depósito o se excedió el máximo de clientes totales el cliente será rechazado. En caso de que haya espacio los clientes guardan una única vez (no agregan más m^2 en el futuro).

Los clientes pueden retirar la mercadería del depósito indicando los m^2 que quedarán disponibles para otros clientes. Se asume que siempre retiran exactamente lo mismo que guardaron (no hay que controlar esto). Al retirar todo ya no se consideran como clientes de la empresa. El retiro de mercadería tiene prioridad sobre el ingreso de clientes nuevos. No puede haber más de un cliente a la vez en el depósito.

En el depósito trabaja una cuadrilla que lo limpia periódicamente. Cuando la cuadrilla está limpiando el depósito el mismo no podrá ser usado para agregar o sacar mercadería. La limpieza tiene menos prioridad que los pedidos pendientes de agregar o sacar cosas del depósito.

Se pide: modelar en ADA los clientes y la cuadrilla de limpieza. Se pueden usar tareas auxiliares.

Nota: Las únicas razones válidas para rechazar clientes son que no haya espacio o que no se puedan tomar nuevos clientes. En cualquier otro caso el mismo deberá esperar hasta poder ser aceptado.

Se cuenta con las siguientes funciones auxiliares:

- `que_hacer(tipo: out boolean, m2: out integer)`: Usada por los clientes saber qué quieren hacer, cuando tipo es verdadero quieren guardar y cuando es falso quieren retirar
- `guardar(m2: integer)`: Usada por los clientes para guardar la mercadería
- `retirar(m2: integer)`: Usada por los clientes para retirar la mercadería
- `limpiar()`: Usada por la cuadrilla para limpiar el depósito
- `otras_tareas()`: Usada por la cuadrilla luego de terminar de limpiar

Solución:

```
task deposito is
    entry guardar(m2: in integer, result: out boolean)
    entry retirar(m2: in integer)
    entry limpiar()
    entry terminado()
end deposito

task body deposito is
var clientes: integer;
    en_uso, ok: boolean;
    deposito, metros: integer;
begin
    clientes := 0;
    en_uso := false;
    deposito := 1000;
    loop
        select
            when (retirar'count = 0) and (not en_uso) =>
            accept guardar(m2: in integer, result: out integer)
                result := (clientes < 50) and (m2 <= deposito);
```

```
                ok := result;
                metros := m2
            end accept
            if ok then
                deposito := deposito - metros;
                clientes := clientes + 1;
                en_uso := true;
            end if
        or
            when not en_uso =>
            accept retirar (m2: in integer)
                metros := m2;
            end accept
            deposito := deposito - metros;
            clientes := clientes - 1;
            en_uso:= true;
        or
            when (retirar'count = 0) and (guardar'count = 0) and (not en_uso) =>
            accept limpiar;
            en_uso := true;
        or
            accept terminado;
            en_uso := false;
        end select
    end loop
end deposito

task type cliente is
end cliente

task body cliente is
var guardar, resultado: boolean
    m2: integer
begin
    que_hacer(guardar, m2);
    if guardar then
        deposito.guardar(m2, result);
        if resultado then
            guardar(m2);
            deposito.terminado();
        end if
    else
        deposito.retirar(m2);
        retirar(m2);
        deposito.terminado();
    end if
end cliente

task cuadrilla is
end cuadrilla

task body cuadrilla is
    loop
        deposito.limpiar();
        limpiar();
        deposito.terminado();
```

```
        otras_tareas();
    end loop
end cuadrilla
```