Departamento de Arquitectura

Examen de Sistemas Operativos

20 de diciembre 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 **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)

Conteste justificando brevemente cada una de sus respuestas.

(a) (4 pts) Describa brevemente: sistemas batch, sistemas de tiempo compartido, sistemas de tiempo real y sistemas distribuidos.

(b) I. (2 pts) Explique brevemente la estructura de una tabla de páginas jerárquica.

II. (2 pts) ¿Cuál es la motivación para utilizar dicha estructura?

(c) I. (2 pts) Describa como funciona la TLB (Translation Lookaside Buffer) e indique por que es indispensable para el uso eficiente de la memoria principal.

II. (2 pts) ¿Qué ventajas agrega el uso de ASID – Address Space Identifier en un TLB?

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

(e) (4 pts) Defina sistema de archivos virtual y describa brevemente sus funcionalidades más importantes.

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

(g) (4 pts) Describa 3 métodos para la interacción entre la controladora y un dispositivo durante una operación de Entrada/Salida.

(h) (4 pts) Describa y compare hipervisores tipo I y II.

Problema 2 (33 pts)

Se tiene un sistema operativo multiprogramado en el cual se dispone de un único procesador y un modelo de hilos **Mx1**.

Cuenta con un planificador por **prioridades** que sigue las siguientes reglas:

- Cuando un proceso ingresa a la cola de listos, su prioridad será el largo del próximo CPU-Burst (**a nivel de proceso**)
- Cada 5ms, los procesos en la cola de listos suman 5 a su prioridad
- El planificador es **expropiativo**

La planificación a nivel de hilos es **round robin** con un cuanto de 5ms.

Existen dos procesos P1 y P2. P1 a su vez cuenta con dos hilos, P1A y P1B, que ejecutan concurrentemente:

P1		P2
P1A	P1B	
Ejecuta 10ms	Ejecuta 15ms	Ejecuta 10ms
Bloquea 10ms	Bloquea 5ms	Bloquea 10ms
Ejecuta 5ms	Ejecuta 5ms	Ejecuta 10ms

Se asume que la operación de bloqueo lleva un tiempo despreciable, esto podría interpretarse como que la última instrucción del bloque de ejecución se encarga de bloquear.

Inicialmente en el sistema no se tiene ningún proceso de usuario ejecutando y en tiempo t=0 se lanza la ejecución de P1 (P1A y P1B) y P2.

Se pide:

(a) (2 pts) Realice un diagrama que muestre los estados y transiciones que tiene un proceso cualquiera en un sistema operativo con esas características. Describa brevemente cada componente del diagrama.

(b) (27 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. Se recomienda utilizar la siguiente notación: $\text{Estado}_{\text{prioridad}}$ (un proceso en la cola de listos con prioridad 10 L_{10}, un proceso ejecutando E, un hilo en la primera posición de la cola de listos L_1, etc.)

(c) (2 pts) Defina tiempo de espera y tiempo de retorno.

(d) (2 pts) Calcule el tiempo de espera y el tiempo de retorno de cada proceso.

Solución:

		5	10	15	20	25	30	35	40	45	50	55	60	65
Nivel de Usuario	P1A	E	L'1	L'1	L'1	E	B	B	L'1	E	T	T	T	T
	P1B	L'1	E	L'2	L'2	L'1	L'1	L'1	E	L'1	E	B	E	T
Nivel de Kernel	P1	E15	E15	L5	L10	E15	B	B	E15	E15	E15	B	E5	T
	P2	L10	L15	E20	E20	B	B	E10	L5	L10	L15	E20	T	T

Tiempo de espera P1: 10ms
Tiempo de espera P2: 25ms
Tiempo de retorno P1:60ms
Tiempo de retorno P2: 55ms

Problema 3 (35 pts)

Se desea modelar en ADA una heladería. En la heladería hay dos vendedores y cinco empleados que arman los helados. Los cucuruchos más grandes solo pueden ser armados por los empleados más experientes por lo que, dependiendo del helado solicitado por el cliente, no necesariamente podrá ser atendido por cualquier empleado.

Cuando llega un cliente le hace el pedido al vendedor y este consulta **simultáneamente** a los cinco empleados para ver si alguno está libre y es capaz de armar el helado solicitado (asumiremos que se prepara un helado por cliente). El primer empleado que conteste afirmativamente será el encargado de armar el helado solicitado y entregárselo al cliente quien esperará hasta que el pedido esté pronto. A los otros empleados disponibles (en caso que los haya) se les indicará que el pedido ya fue atendido por otro de ellos. En caso de que todos estén ocupados se le indica al cliente que no se le puede vender en este momento y se retira.

Cada cierto tiempo llega un supervisor de bromatología el cual verifica que los helados no estén contaminados. Mientras el supervisor trabaja no se podrá armar ningún helado (los empleados pueden terminar de armar los que ya habían empezado antes de que llegue el supervisor). El supervisor tiene prioridad sobre los clientes. Es válido rechazar clientes nuevos mientras el inspector está trabajando.

Se dispone de los siguientes procedimientos auxiliares:

- `elegir_vendedor(): {1,2}` que ejecutado por un cliente le indica el número de vendedor con quien comunicarse.
- `que_helado(): pedido` que ejecutado por un cliente retorna un pedido con el helado que desea el cliente
- `comer_helado(helado)` ejecutado por el cliente para comer el helado
- `puedo_armar_helado(pedido): boolean` ejecutado por el empleado para ver si puede armar el helado
- `armar_helado(pedido): helado` que ejecutado por un empleado arma el helado solicitado
- `verificar_helados()` ejecutado por el inspector para verificar el estado de los helados en el local
- `otras_tareas_inspector()` ejecutado por el inspector cuando no esta inspeccionando

Se pide:

(a) (2 pts) Dado el siguiente código:

```
select
        a.encuentro()
        Bloque A
else
        Bloque B
```

¿Qué sucede si la tarea a no está aceptando el encuentro?

(b) (33 pts) Implementar en ADA las tareas vendedor, empleado, cliente e inspector. Se puede usar hasta una tarea auxiliar.

Solución:

a) Se ejecuta el Bloque B

b)

```
task type cliente is
end cliente;

task body cliente is
var
    empleado: integer;
    h: helado;
    ok: boolean;

begin
    vendedores[elegir_vendedor()].pedir_helado(que_helado(), empleado, ok);
    if ok then
        empleados[empleado].entregar_helado(h);
        comer_helado(h);
    end if
end

task type vendedor is
    entry pedir_helado(p: IN pedido, empleado: OUT integer, ok: OUT boolean);
    entry respuesta_empleado(resp: IN boolean, id: IN integer, ok: OUT boolean);
    entry obtener_id(id: IN integer);
end

task body vendendor is
var
    mi_id, cant, i, id_c: integer;
    resp, primero: boolean;

begin
    accept obtener_id(id: IN integer) do
        mi_id := id;
    end accept

    loop
        cant := 5;
        accept pedir_helado(p: IN pedido, empleado: OUT integer, ok: OUT boolean)
        do
            for i:= 1 to 5 do
                select
                    cocineros[i].puede_preparar(mi_id, p);
                else
                    cant := cant - 1;
                end select
            end for

            ok := false;
            if cant > 0 then
                while (cant > 0) and (not ok) do
                    accept respuesta_empleado(resp: IN boolean, id_c: IN integer,
                            ok_empleado: OUT boolean) do
                        if resp then
                            ok_empleado := true;
                            ok := true;
                            empleado := id_c;
                        else
                            ok_empleado := false;
```

```
                    end
                end accept
                cant := cant - 1;
            end while
        end if
    end accept
    -- acepto los empleados que quedaban pendientes
    while (cant > 0) do
        accept respuesta_empleado(resp: IN boolean, id_c: IN integer,
                ok_empleado: OUT boolean) do
             ok_empleado := false;
         end accept
         cant := cant - 1;
     end while
end loop
end

task type empleado is
    entry puede_preparar(vendedor: IN integer, p: IN pedido);
    entry entregar_helado(hel: OUT helado);
    entry llega_inspector();
    entry sale_inspector();
    entry obtener_id(id: IN integer);
end

task body empleado is
var
    vend, mi_id: integer;
    ok: boolean;
    p: pedido;
    h: helado;

begin
    accept obtener_id(id: IN integer) do
        mi_id := id;
    end accept

    loop
        select
            when llega_inspector'count = 0 =>
            accept puede_preparar(vendedor: IN integer, ped: IN pedido) do
                vend := vendedor;
                p := ped;
            end accept;
            vendedores[vend].respuesta_empleado(puedo_armar_helado(p), mi_id, ok);
            if ok then
                h = armar_helado(p);
                accept entregar_helado(hel: OUT helado) do
                    hel := h;
                end accept;
            end if;
        or
            accept llega_inspector();
            accept sale_inspector();
        end select
    end loop
```

```
end

task inspector is
end

task body inspector is
begin
    loop
        otras_tareas_inspector();
        for i := 1 to 5 do
            empleados[i].llega_inspector();
        end for
        verificar_helados();
        for i := 1 to 5 do
            empleados[i].sale_inspector();
        end for
    end loop
end

empleados: array[1..5] of empleado;
vendedores: array[1..2] of vendedor;

// programa principal
var i: integer;

begin
    vendedores[1].obtener_id(1);
    vendedores[2].obtener_id(2);

    for i := 1 to 5 do
        empleados[i].obtener_id(i);
    end for
end
```