Departamento de Arquitectura

Super parcial de Sistemas Operativos

15 de Julio 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 (20 pts)

(a) (4 pts) Nombre y describa brevemente dos métodos de asignación del espacio libre en disco.

(b) (4 pts) Defina los conceptos de soft link y hard link en Unix. Explique las diferencias entre ambos.

(c) (5 pts) Dado este código con mailboxes infinitos, send no bloqueante y receive bloqueante:

```
var a: mailbox of integer;

procedure vendedor
var item, pedido: integer;
begin
    while true
      item = receive(a);
      pedido = vender(item);
      send(a, pedido);
end

procedure cliente
var item, pedido: integer;
begin
  item = que_comprar();
  send(a, item);
  pedido = receive(a);
end

cobegin
  vendedor();
  cliente();
  cliente();
coend
```

Explique qué problemas podrían ocurrir durante su ejecución

Solución:

- Un primer problema que ocurre es el siguiente: Al utilizar el mismo mailbox, si el cliente mantiene la CPU durante toda la ejecución de su función, se ejecutará el send del cliente, y luego al ejecutar la línea `pedido = receive(a)`, no se bloqueará, sino que retirará el mismo pedido que él envió.
- Otro problema que puede ocurrir aún solucionando lo anterior, es el siguiente:

  El primer cliente ejecuta su send y antes de ejecutar el receive pierde la CPU. Luego el segundo cliente ejecuta su send y también pierde la CPU. Luego el vendedor ejecuta el receive, retira el pedido del primer cliente, lo procesa, y ejecuta el send con el pedido del primer cliente. Luego el segundo cliente recupera la CPU y ejecuta el receive. En esta situación, el segundo cliente estría realizando un pedido, pero estaría retirando el pedido del primer cliente.

(d) (4 pts) Describa qué es DAC (Discretionary Access Control)

(e) (3 pts) ¿Para qué sirve la protección de CPU? ¿Cómo se implementa?

Problema 2 (22 pts)

Sea un sistema de paginación en donde se utiliza paginación multinivel, direcciones virtuales de 32 bits y se cumplen los siguientes puntos:

- Utiliza paginación de memoria.
- Las páginas son de 4K Bytes
- Las tablas de páginas cuentan con 1K entradas y ocupan una página completa.
- Una memoria física de 512M Bytes

Se pide:

(a) (4 pts) Indique cuántos niveles de tabla de páginas son necesarios en este sistema. Indique el tamaño y formato de las direcciones lógicas.

Solución:

Las direcciones lógicas se separan en | # de página | desplazamiento |

Como las páginas son de 4 KB, el desplazamiento es de 12 bits (2^{12} = 4 K).

Como la dirección lógica es de 32 bits, y 12 se utilizan en el desplazamiento, se tienen 20 bits para direccionar las tablas de páginas.

Como se tienen 20 bits para direccionar las tablas de páginas, pero la de primer nivel solo admite 1 K entradas (2^{10}), es decir que se precisan 10 bits para indexarla. Se deduce que se precisa un segundo nivel de tabla de páginas, con la misma estructura, para direccionar los restantes 10 bits.

Por lo tanto la dirección lógica se divide como:

| índice en pág. 1er nivel (10 bits) | índice en pág. 2do nivel (10 bits) | desplazamiento (12 bits) |

(b) (4 pts) Indique el tamaño de las entradas en las tablas de páginas.

Solución: Existen 1 K (2^{10}) entradas en cada tabla de páginas y cada tabla ocupa completamente una página de tamaño 4 KB (2^{12}). Entonces se dispone de 2^{12} bytes/$2^{10} = 2^2$ bytes $= 4$ bytes por entrada en cada tabla de páginas.

(c) (3 pts) ¿Qué tamaño tiene el espacio virtual?

Solución: Como la dirección lógica tiene 32 bits, el tamaño del espacio virtual es 2^{32} bytes = 4 GB.

(d) (7 pts) Asumiendo que tenemos un proceso que requiere de 128 MB = 2^{27} bytes para almacenar su código, datos globales en memoria y datos dinámicos, y que la memoria requerida por su pila es de 16 MB (2^{24} bytes). Determinar:

i. ¿Cuántas tablas utiliza el proceso?

ii. ¿Qué índices se utilizan en cada tabla?.

Solución: El proceso requiere 128 MB (2^{27} bytes) para código y memoria global (.bss y .data) y 16 MB (2^{24} bytes) para el stack. El stack se ubica en la región más alta de la memoria, y el código en la menor, por lo tanto no comparten páginas.

Cada página tiene capacidad para 2^{12} bytes, los primeros 128 MB requieren $2^{27}/2^{12} = 2^{15} = 32$K páginas para contenerlos. Como cada tabla de páginas de 2do nivel direcciona hasta 1K (2^{10}) páginas, se requieren $2^{15}/2^{10} = 2^5 = 32$ tablas de páginas completas de segundo nivel, y por lo tanto se utilizan 32 entradas de la tabla de páginas de 1er nivel.

Con un razonamiento análogo, los 16 MB de stack requieren $2^{24}/2^{12} = 2^{12} = 4K$ páginas, por lo tanto se utilizan $2^{12}/2^{10} = 2^2 = 4$ tablas de páginas completas de 2do nivel y 4 entradas en la tabla de páginas de 1er nivel.

Se utilizan en total 36 tablas de páginas de 2do nivel y la tabla de páginas de 1er nivel. Las tablas de páginas de 2do nivel se utilizan completamente. De la tabla de 1er nivel se utilizan los primeros 32 índices (0 a 31) para los segmentos de código y datos, y los últimos 4 índices para el segmento de stack ($2^{10} - 1$ a $2^{10} - 4$).

(e) (4 pts) Describa la estrategia de reemplazo Segunda oportunidad (Second chance).

Solución: Ver teórico.

Problema 3 (24 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 $10ms$ y un modelo de hilos $M \times 1$. 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
   pid = fork();
   if pid = 0 then
      Ejecuta 5ms
      Bloquea 10ms
      Ejecuta 5ms
   else
       create_thread(proc1)
       create_thread(proc1)
   end if
 end
```

```
procedure proc1()
begin

   Ejecuta 5ms
   Bloquea 5ms
   Ejecuta 5ms
end
```

Se sabe que la operación `fork()` es estilo Unix. 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 $5ms$ (`fork`, `create_thread`,

condición del if, etc). La asignación en pid = fork() se considera instantánea (es decir, solo cuenta el tiempo de ejecución del `fork`). La planificación definida para los hilos a nivel de usuario es FCFS.

Se pide:

(a) (2 pts) Explique el modelo de hilos $M \times 1$.

(b) (2 pts) Explique por qué la función `fork` retorna 0 al proceso hijo y el pid del proceso hijo al proceso padre

> **Solución:** Un proceso puede tener muchos hijos, pero cada proceso tiene únicamente un padre. Por lo tanto, tiene sentido que el `proceso padre` reciba el PID del proceso hijo, puesto que el proceso hijo puede solicitar el identificador de su proceso padre a través de una system call. Además, devolver id = 0 al hijo permite realizar bifurcaciones simples en el código para separar el código que ejecuta el hijo y el que ejecuta el padre.

(c) (18 pts) Realice un diagrama de planificación (tiempo vs hilos), comenzando en el tiempo $t = 0$, indicando el estado de cada uno de los hilos.

> **Solución:**

	0	5	10	15	20	25	30	35	40	45	50	55	60	65
P1	E	E	L	L	E	E	L	E	B	E	B	E	E	T
P2		L	E	E	B	B	E	T						
$P1_1$	E	E	L	L	E	E	T							
$P1_2$						L	L	E	B	L	B(L)	E	T	
$P1_3$							L	L	B(L)	E	B	L	E	T

(d) (2 pts) ¿Cuál es el proceso con mayor tiempo de espera? ¿Cuál es el proceso con menor tiempo de retorno? Justifique sus respuestas.

> **Solución:**
> El proceso con mayor tiempo de espera es el proceso 1 (15ms). El proceso con menor tiempo de retorno es el proceso 2 (30ms).

Problema 4 (34 pts)

(a) (6 pts) Dadas las siguientes tablas de recursos asociados a procesos y de máxima cantidad de recursos requeridos por cada proceso indique si el sistema se encuentra en un **estado seguro** sabiendo que hay un máximo de 4 instancias de cada recurso.

	R1	R2	R3	R4
P1	0	1	1	1
P2	0	0	0	0
P3	2	2	0	0
P4	0	1	3	0

Recursos asignados

	R1	R2	R3	R4
P1	0	1	1	1
P2	4	2	0	0
P3	2	4	0	3
P4	3	2	4	0

Recursos máximos requeridos

> **Solución:** Para ver si el sistema está en un estado seguro es necesario encontrar un orden en que se puedan satisfacer los pedidos de todos los procesos. En el estado que se indica los recursos libres son:

R1	R2	R3	R4
2	0	0	3

y sabemos que P1 ya tiene todos los recursos que necesita por lo que podemos asumir que en algún momento va a terminar y liberarlos. En ese estado los recursos libres serán:

R1	R2	R3	R4
2	1	1	4

y tenemos que ver los recursos que precisan los otros procesos. Restando las matrices tenemos que los recursos que necesitan son:

	R1	R2	R3	R4
P2	4	2	0	0
P3	0	2	0	3
P4	3	1	1	0

Por lo que podemos ver que con los recursos libres no se puede satisfacer a ninguno de los tres procesos restantes por lo que el sistema no está en un estado seguro.

(b) (28 pts) Se quiere definir un sistema que permita modelar el ingreso de docentes y estudiantes al local de una muestra de parciales. Por la situación sanitaria no pueden ingresar más de 20 personas al local. Las personas entrarán en orden de llegada al local. Pero:

i para que ingresen estudiantes tiene que haber al menos un docente dentro del local

ii en caso de tener personas esperando fuera tendrán prioridad para el ingreso a la sala los docentes

iii en caso que un docente se quiera retirar debe considerar que mientas haya estudiantes en el salón debe quedar al menos un docente.

Implemente con monitores la realidad anterior. Dispone de las funciones:

Muestro_Parciales()
Procedimiento de los docentes que modela la tarea de mostrar parciales. Una vez concluida la ejecución de la función el docente se retira del salón.

Reviso_Parcial()
Procedimiento de los estudiantes, para revisar el parcial. Una vez concluida la ejecución de la función el estudiante se retira del salón.

Nota: no es necesario modelar las interacciones docentes-estudiante durante la muestra.

Solución:

```
procedure docente ()
begin
    salon.entra_docente();
    muestro_parciales();
    salon.sale_docente();
end

procedure estudiante ()
begin
    salon.entra_estudiante();
    reviso_parcial();
    salon.sale_estudiante();
end
```

```
Monitor salon
begin
    var estudiantes, cant_e, cant_d, cant, doc_esperando,
    esperando_doc:integer;

    var docente, docente_salida, estudiante,
    estudiante_docente: condition;

    procedure entra_docente()
    begin
        if (cant == 20) then
            doc_esperando++;
            docente.wait();
            doc_esperando--;
        end
        cant_d ++;
        cant++;

        docente_salida.signal();
        estudiante_docente.signal();
    end

    procedure sale_docente()
    begin
        if (estudiantes > 0 && cant_d == 1) then
            docente_salida.wait();
        end
        cant_d--;
        cant--;
        if (doc_esperando > 0) then
            docente.signal();
        else
            estudiante.signal();
        end
    end

    procedure entra_estudiante()
    begin
        estudiantes++;
        if (cant_doc == 0) then
            estudiante_docente.wait();
            estudiante_docente.signal();
        end
        if (cant == 20) then
            estudiante.wait();
        end
        cant ++;
    end
```

```
    procedure sale_estudiante()
    begin
        estudiantes--;
        cant--;
        if (doc_esperando > 0) then
            docente.signal();
        else if (estudiantes == 0) then
            docente_salida.signal();
        else
            estudiante.signal();
        end
    end


    begin
        estudiantes = 0;
        cant_e = 0;
        cant_d = 0;
        cant = 0;
        doc_esperando = 0;
        esperando_doc = 0;
    end
end

cobegin
    docente();
    ...
    docente();
    estudiante();
    ...
    estudiante();
coend
```