Departamento de Arquitectura

Primer parcial de Sistemas Operativos

7 de mayo de 2024

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

- **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**.
- Sólo se contestarán dudas de letra y no se aceptarán dudas en los últimos 30 minutos del parcial.
- El parcial es **SIN** material. En su banco solo puede tener las hojas del parcial, lápiz, goma y lapicera. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.
- El parcial dura **3 horas**. Al momento de finalizar el parcial **no se podrá escribir absolutamente nada en las hojas**. El estudiante debe ponerse de pie e ir a la fila de entrega. Se debe identificar cada una de las hojas con nombre, cédula y numeración antes de la hora de finalización del parcial.

Importante: Debe justificar todas las respuestas, siempre.

Problema 1 (15 puntos)

(a) (3 puntos) Explique los mecanismos que dispone una llamada al sistema para comunicar datos al sistema operativo.

> **Solución:**
> Existen tres mecanismos para que una llamada al sistema comunique datos al sistema operativo:
>
> - A través de un conjunto específico de registros.
> - Utilizando un bloque de memoria apuntado por un registro.
> - A través del stack del proceso que realiza el llamado.

(b) Considere un sistema operativo con micronúcleo.

I. (2 puntos) Describa cómo se construye y explique la principal funcionalidad del sistema.

> **Solución:**
> El sistema se construye mediante un núcleo que brinde un manejo mínimo de procesos y gestión de memoria, y que provea una capa de comunicación entre procesos. La capa de comunicación es la funcionalidad principal del sistema dado que los demás servicios se construyen como procesos separados al micronúcleo que ejecutan en modo usuario y acceden a través de pasaje de mensajes.

II. (2 puntos) Explique qué ventajas aporta sobre las otras estructuras para sistemas operativos vistas en el curso.

> **Solución:**
> Un sistema con micronúcleo aporta las siguientes ventajas:
>
> - Aumenta la portabilidad y la escalabilidad, al encapsular las características físicas del sistema.
> - No es necesario modificar el núcleo del sistema para incorporar un nuevo servicio.

- Es más seguro, dado que los servicios corren en modo usuario.
- El diseño simple y funcional permite construir sistemas más confiables.

(c) (4 puntos) Explique el funcionamiento de las instrucciones de sincronización provistas por hardware que implementan locks y presente un breve pseudocódigo que ejemplifique su uso.

Solución:

Las instrucciones provistas por hardware para implementar locks son TestAndSet y Swap. Ejecutan de forma atómica pero tienen la desventaja que necesitan de un busy-waiting para esperar que se cumpla la condición.

Un pseudocódigo para TestAndSet es:

```
lock = false;
procedure seccionCritica-TS;
begin
    while (TestAndSet(lock)); // busy waiting
    region_critica();
    lock = false;
    otras_tareas();
end;
```

Un pseudocódigo para Swap es:

```
lock = false;
procedure seccionCritica-S;
begin
    var = true;
    while (var) do
        Swap(lock,var); // busy waiting
    region_critica();
    lock = false;
    otras_tareas();
end;
```

(d) (4 puntos) Explique cómo se implementa la comunicación con dispositivos de entrada/salida, cómo se carga el software utilizado y los mecanismos utilizados para intercambiar información.

Solución:

La comunicación con dispositivos de entrada/salida se realiza mediante un software específico denominado device driver, que se incorpora al sistema operativo. Los device drivers son cargados de diferentes formas: i) son ensamblados estáticamente al núcleo del sistema, ii) se utiliza un archivo de configuración que indica cuáles cargar y iii) se cargan dinámicamente a demanda. Para intercambiar información (datos, control) las controladoras tienen un conjunto de registros que se acceden mediante dos mecanismos:

- Mapeo de memoria: los registros se mapean a un espacio reservado de direcciones de memoria principal. La comunicación se realiza leyendo y escribiendo sobre esas direcciones.
- Instrucciones directas: A cada registros se le asigna una dirección de puerto. La comunicación se realiza mediante instrucciones privilegiadas IN y OUT que permiten a los device drivers leer o escribir en los registros de la controladora.

Nota: Se aceptó como correcta la solución que explica los métodos de implementar una comunicación de Entrada/Salida, siempre que incluyera una referencia al método de acceso directo a memoria (DMA).

Problema 2 (15 puntos)

Sea un sistema operativo multiprogramado con un único procesador que utiliza un planificador con dos colas de prioridad (alta y media) con retroalimentación. La cola de alta prioridad tiene una planificación Round-Robin con un cuanto de 2ms y la cola de media prioridad tiene una planificación Shortest Job First expropiativa. Un proceso en la cola de alta prioridad que utiliza todo su cuanto sin bloquearse, se mueve a la cola de media prioridad. Un proceso en la cola de media prioridad sube a la cola de alta prioridad si se bloquea. Se considera que la última instrucción de un proceso en un bloque "Ejecuta" previo a un bloque "Bloquea" es la que bloquea al proceso. Cualquier caso de empate se define con una prioridad dada por el índice del proceso: a menor índice, mayor prioridad (prioridad(P1) >prioridad(P2) >... >prioridad(Pn)).

En el instante t=0 se lanza la ejecución de tres procesos (P1, P2 y P3) en la cola de alta prioridad con los siguientes tiempos de ejecución y bloqueo:

P1	P2	P3
Ejecuta 4ms	Ejecuta 2ms	Ejecuta 3ms
Bloquea 2ms	Bloquea 3ms	Bloquea 2ms
Ejecuta 4ms	Ejecuta 4ms	Ejecuta 1ms
Bloquea 2ms	Bloquea 1ms	Bloquea 2ms
Ejecuta 2ms	Ejecuta 2ms	Ejecuta 3ms
Termina	Termina	Termina

(a) (12 puntos) Realice el diagrama de planificación (Tiempo vs.Procesos) que indique el estado de cada proceso (listo/ejecutando/bloqueado/terminado) en cada intervalo de tiempo. Además, se debe indicar en cada caso el nivel de prioridad en el que se encuentra el proceso (en qué lugar de la cola de prioridad que corresponda).

Justifique brevemente en cada caso por qué a un proceso se le asigna el recurso CPU y por que un proceso sube o baja su prioridad.

Se recomienda seguir la siguiente nomenclatura para el diagrama de planificación:

- E: ejecutando
- LA1: listo en alta prioridad en el primer lugar de la cola
- LM3: listo en media prioridad en el tercer lugar de la cola
- B: bloqueado
- T: terminado

Se recomienda seguir el siguiente formato para el diagrama de planificación:

Tiempo	P1	P2	P3	Justificación
1				
2				

Solución:

Tiempo	P1		P2		P3		Cola de alta prioridad	Cola de media prioridad	
	Instrucciones	Estado	Instrucciones	Estado	Instrucciones	Estado			Justificación
1	Ejecuta 4ms	E	Ejecuta 2ms	LA1	Ejecuta 3ms	LA2	[P2, P3]	[]	Todos comienzan el la cola de alta prioridad que es RR, el mecanismo de desempate es: prioridad(P1) >prioridad(P2) >prioridad(P3).
2		E		LA1		LA2	[P2, P3]	[]	P1 agota el quantum sin bloquearse, baja a la cola de prioridad media y ejecuta el siguiente proceso de la cola de prioridad alta.
3		LM1		E		LA1	[P3]	[P1 cpu_b=2]	Pasa a ejecutar P2, al tratarse del siguiente proceso existente en la cola de prioridad alta.
4		LM1		E		LA1	[P3]	[P1 cpu_b=2]	Dado que "Un proceso en la cola de alta prioridad que utiliza todo su cuanto sin bloquearse, se mueve a la cola de media prioridad" y como "Se considera que la última instrucción de un proceso en un bloque "Ejecuta" previo a un bloque "Bloquea" es la que bloquea al proceso", P2 seguirá en la cola de prioridad alta ya que no cumple con la primer sentencia, porque se está bloqueando al finalizar la ejecución en el tiempo 4, como indica la segunda sentencia.
5		LM1	Bloquea 3ms	B		E	[]	[P1 cpu_b=2]	Pasa a ejecutar P3, al tratarse del siguiente proceso existente en la cola de prioridad alta.
6		LM1		B		E	[]	[P1 cpu_b=2]	Como P3 agota el quantum sin bloquearse, baja a la cola de prioridad media
7		LM1		B		E	[]	[P1 cpu_b=2]	P3 sigue ejecutando ya que su CPU-brust es 1 y es menor al de P1.
8		LM1	Ejecuta 4ms	E	Bloquea 2ms	B	[]	[P1 cpu_b=2]	P2 al desbloquearse pasa a la cola de prioridad alta y por eso ejecuta antes de P1 que se encuentra en la cola de prioridad media.
9		LM1		E		B	[]	[P1 cpu_b=2]	P3 al desbloquearse pasa a la cola de prioridad alta y por eso ejecutará antes de P1 y P2 que se encuentra en la cola de prioridad media.
10		LM1		LM2	Ejecuta 1ms	E	[]	[P1 cpu_b=2, P2 cpu_b=2]	El mecanismo de desempate para la cola de prioridad media es: prioridad(P1) >prioridad(P2) >prioridad(P3).
11		E		LM1	Bloquea 2ms	B	[]	[P2 cpu_b=2]	Solo hay procesos en la cola de prioridad media, se toma el siguiente de allí.
12		E		LM1		B	[]	[P2 cpu_b=2]	P2 al desbloquearse pasa a la cola de prioridad alta y por eso ejecutará antes que los que se encuentra en la cola de prioridad media.
13	Bloquea 2ms	B		LM1		E	[]	[P2 cpu_b=2]	Ejecuta el proceso de prioridad alta
14		B		LM1	Ejecuta 3ms	E	[]	[P2 cpu_b=2]	P3 agota el quantum sin bloquearse, baja a la cola de prioridad media y ejecuta el siguiente proceso de la cola de prioridad alta.
15	Ejecuta 4ms	E		LM2		LM1	[]	[P3 cpu_b=1, P2 cpu_b=2]	Como P3 tiene menor CPU-brust su prioridad es mayor que la de P2.

Tiempo	P1		P2		P3		Cola de alta prioridad	Cola de media prioridad	
	Instrucciones	Estado	Instrucciones	Estado	Instrucciones	Estado			Justificación
16		E		LM2		LM1	[]	[P3 cpu_b=1, P2 cpu_b=2]	Como P1 agota el quantum sin bloquearse, baja a la cola de prioridad media
17		LM1		LM2		E	[]	[P1 cpu_b=2, P2 cpu_b=2]	El mecanismo de desempate para la cola de prioridad media es: prioridad(P1) >prioridad(P2) >prioridad(P3).
18		E		LM1	Termina	T	[]	[P2 cpu_b=2]	Ejecuta P1 ya que es el proceso de mayor prioridad (por desempate)
19		E		LM1			[]	[P2 cpu_b=2]	
20	Bloquea 2ms	B		E			[]	[]	Ejecuta el único proceso listo
21		B		E			[]	[]	
22	Ejecuta 2ms	E	Bloquea 1ms	B			[]	[]	P2 al desbloquearse pasa a la cola de prioridad alta y por eso ejecutará antes que los que se encuentra en la cola de prioridad media.
23		E		LA1			[P2]	[]	
24	Termina	T	Ejecuta 2ms	E			[]	[]	
25				E			[]	[]	
26			Termina	T			[]	[]	

Nota: Se aceptó como solución correcta que P2 baje a la cola de media prioridad en el tiempo 8 (luego de desbloquearse) al poder considerar que se consume todo su cuanto.

(b) (3 puntos) Calcule el tiempo de espera y de retorno de cada proceso.

Solución:

Espera(P1) = 9ms, Retorno(P1) = 23ms
Espera(P2) = 13ms, Retorno(P2) = 25ms
Espera(P3) = 6ms, Retorno(P3) = 17ms

Problema 3 (20 puntos)

Se desea modelar una atracción de un parque de diversiones. Para acceder a la atracción los clientes esperan para subir sin importar el orden de llegada, pero dando prioridad en el ingreso a los menores. La atracción tiene una capacidad de 3 personas y solamente comienza a funcionar cuando está ocupada totalmente.

Para garantizar el orden y la seguridad en la atracción, los clientes deben subir uno por vez y bajar en el mismo orden que subieron. Al igual que ocurre al subir, los clientes deben bajar de a uno.

Se dispone de las siguientes funciones auxiliares:

- `soy_menor(): boolean`, ejecutada por el cliente para saber si es menor.
- `iniciar()`, inicia el funcionamiento de la atracción y se bloquea durante su tiempo de operación.
- `subir()`, ejecutada por el cliente para subir al juego.
- `salir()`, ejecutada por el cliente para salir del juego.

(a) (8 puntos) Implementar los procesos cliente y atracción usando monitores

Solución:

```
monitor parque
    var menores, mayores, adentro: integer
    var entra_menor, entra_mayor, inicio, espera: condition

    procedure entra(menor: boolean)
    begin
```

```
            if adentro >= 3 then
                if menor then
                    menores++;
                    wait(entra_menor);
                else
                    mayores++;
                    wait(entra_mayor);
                end
            end
            adentro++;
            subir();
            if adentro = 3 then
                signal(inicio);
            end
            wait(espera);
            bajar();
            adentro--;
        end

        procedure deja_entrar()
        begin
            for i:= 1 to (3 - adentro) do
                if menores > 0 then
                    menores--;
                    signal(entra_menor);
                else if mayores > 0 then
                    mayores--;
                    signal(entra_mayor);
                end
            end
        end

        procedure espera_lleno()
            if adentro < 3 then
                wait(inicio);
            end
        end

        procedure fin()
        begin
            for i:= 1 to 3 do
                signal(espera);
            end
        end

        begin
            mayores := 0;
            menores := 0;
            adentro := 0;
        end
    end monitor

    procedure cliente
    begin
        parque.entra(soy_menor());
    end
```

```
procedure atraccion
begin
    while true do
        parque.deja_entrar();
        parque.espera_lleno();
        iniciar();
        parque.fin();
    end
end

begin
    cobegin
        cliente();
        ...
        cliente();
        atraccion();
    coend
end
```

(b) (12 puntos) Implementar los procesos cliente y atracción usando semáforos

Solución:

```
var mayores, menores, id: integer
var mutex_menor, mutex_mayor, mutex_id, entra_menor, entra_mayor:semaphore
var llega_cliente, sync_entrada, sync_salida: semaphore
var fin: array[0..2] of semaphore

begin
    mayores := 0;
    menores := 0;
    init(mutex_menor, 1);
    init(mutex_mayor, 1);
    init(mutex_id, 1);
    init(entra_menor, 0);
    init(entra_mayor, 0);
    for i:= 1 to 3 do
        init(fin[i], 0);
    end
    init(llega_cliente, 0);
    init(sync_entrada, 0);
    init(sync_salida, 0);
    cobegin
        cliente();
        ...
        cliente();
        atraccion();
    coend
end

procedure cliente
var pos: integer
begin
    if soy_menor() then
```

```
            P(mutex_menor);
            menores++;
            V(mutex_menor);
            V(llega_cliente);
            P(entra_menor);
        else
            P(mutex_mayor);
            mayores++;
            V(mutex_mayor);
            V(llega_cliente);
            P(entra_mayor);
        end
        P(mutex_id);
        pos := id;
        id++;
        subir();
        V(mutex_id);
        V(sync_entrada);
        P(fin[id]);
        bajar();
        V(sync_salida);
    end

    procedure atraccion
    begin
        while true do
            P(mutex_id);
            id := 0;
            V(mutex_id);
            for int i:=1 to 3 do
                P(llega_cliente);
                P(mutex_menor);
                P(mutex_mayor);
                if menores > 0 do
                    V(entra_menor);
                    menores--;
                else
                    V(entra_mayor);
                    mayores--;
                end
                V(mutex_menor);
                V(mutex_mayor);
            end
            for i:= 1 to 3 do
                P(sync_entrada);
            end
            iniciar_juego();
            for i:= 1 to 3 do
                V(fin[i]);
                P(sync_salida);
            end
        end
    end
```