Departamento de Arquitectura

Examen de Sistemas Operativos

2 de agosto de 2024

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

- **Indique su nombre completo, número de examen 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 primer hoja.
- **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 versión.
- Sólo se contestarán dudas de letra y no se aceptarán dudas en los últimos 30 minutos del examen.
- El examen es **SIN material**. En su banco solo puede tener las hojas del examen, lápiz, goma y lapicera. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.
- Para aprobar el examen es necesario un mínimo de **60 puntos**.
- El examen dura **3 horas**. Al momento de finalizar el examen **no se podrá escribir absolutamente nada en las hojas**. El estudiante deberá dirigirse a la fila de entrega. Identificar cada una de las hojas con nombre, número de examen, cédula y numeración debe realizarse dentro de la duración del examen.

Problema 1 (33 pts)

(a) Sobre gestión de memoria:

I. (5 pts) Explique los conceptos de enlazado estático y enlazado dinámico de bibliotecas y sus diferencias. Mencione una ventaja o desventaja de cada uno y explique en qué etapa de la preparación de un programa para su ejecución se cargan las bibliotecas.

> **Solución:** El enlazado estático y enlazado dinámico son dos formas para que los procesos puedan utilizar bibliotecas. En el caso del enlazado estático, en el momento de la compilación se incorpora el código de las bibliotecas junto al código del programa. Tiene la desventaja de que si muchos procesos enlazados estáticamente utilizan la misma biblioteca, cada uno tendrá una copia propia y la memoria se utiliza de manera redundante. El enlazado dinámico consiste en que las bibliotecas compartidas se guardan en memoria compartida y los ejecutables guardan una referencia a la ubicación de las bibliotecas en memoria. Tiene la ventaja de utilizar de manera más eficiente la memoria, pero requiere soporte para enlazado dinámico por parte del sistema operativo y puede producir fallos de página, que implica cierto overhead.

II. (5 pts) ¿Cuándo ocurre un fallo de página? Enumere las acciones que debe tomar el sistema operativo ante un fallo de página.

> **Solución:** Un fallo de página ocurre cuando un proceso intenta acceder a una dirección de memoria ubicada en una página que está en memoria no residente. Al ocurrir un fallo de página, se genera una interrupción que activa una rutina de atención en el sistema operativo para cargar la página en memoria principal. La rutina de atención debe realizar las siguientes acciones:
>
> 1. Buscar un frame libre en memoria principal. Si no se encuentra, se debe enviar un frame víctima, seleccionado mediante un algoritmo de reemplazo, al backing store
> 2. Cargar la página buscada desde el disco y colocarla en memoria principal.
> 3. Actualizar la tabla de páginas y la información en el PCB para indicar que la página está disponible en memoria principal.
> 4. Retornar el control a la instrucción interrumpida por el fallo de página.

(b) Respecto a planificación de procesos.

I. (4 pts) Describa los tres modelos de hilos que proveen los sistemas operativos.

II. (3 pts) ¿Cómo se relacionan los modelos de hilos con los algoritmos de planificación?

III. (3 pts) En un sistema multi procesador, ¿qué tipo de modelo de hilos sería más beneficioso?

Solución:

I. M×1 (Many-To-One): Varios threads de a nivel de usuario a un único thread a nivel de sistema. 1×1 (One-to-One): Cada thread de usuario se corresponde con un thread a nivel del núcleo (kernel thread). M×N(Many-To-Many): Varios threads a nivel de usuario se corresponde con varios threads a nivel del núcleo.

II. Los algoritmos de planificación planifican los hilos a nivel de kernel y determinan el próximo hilo a ejecutar.

III. Depende de la aplicación. Para una aplicación que puede aprovechar el paralelismo mediante la ejecución de múltiples hilos independientes, un modelo 1×1 es el más adecuado, ya que permite planificar los distintos hilos a distintos procesadores. Como ventaja adicional, se puede mencionar que el bloqueo de un hilo no afecta al proceso entero, sino solo al hilo que se bloqueó.

(c) (8 pts) En un servidor de tipo Unix se encuentra el archivo /home/sistoper/solucionExamen.tex, con permisos 660 y de propiedad del usuario sistoper. Suponga que tiene acceso al servidor vía ssh, pero con el usuario estudiante, que no tiene permiso para ejecutar sudo, pero sí puede ejecutar contenedores. La configuración del servidor establece que los contenedores ejecutan como root. Si ejecuta un contenedor montando únicamente el directorio /home/estudiante del host, dentro del contenedor en la ruta /mnt, ¿existe alguna forma que el usuario estudiante lea el archivo solucionExamen.tex? Justifique su respuesta.

Asuma que dentro del contenedor posee los siguientes binarios que son compatibles con el host:

- *bash*: CLI (Command Line Interface).
- *cat*: permite leer el contenido de un archivo.
- *chmod*: permite modificar los permisos de un archivo o directorio.
- *chown*: permite modificar el usuario y/o grupo propietario de un archivo o directorio.

Solución: Es posible, ya que dentro del contenedor se es *root*, lo que permite cambiar propietarios y permisos de los archivos. Mediante *chown* se establece como propietario del binario *cat* a *root*, además de otorgar el permiso especial SUID vía *chmod*. Posteriormente, este ejecutable se copia en la ruta */mnt*, para poder acceder a él desde el host en */home/estudiante/cat*. Esto es posible ya que el sistema de archivos del host fue montado en el contenedor (montaje de directorios). Cuando el usuario *estudiante* ejecute */home/estudiante/cat /home/sistoper/solucion_examen.tex*, el proceso ejecutará como *root* gracias al bit SUID, permitiéndole leer las soluciones, dado que las restricciones de permisos no son aplicadas al superusuario. De igual forma, se pueden realizar los mismos pasos sobre *bash* e iniciar una nueva shell en el host ejecutando */home/estudiante/bash -p* y tomar control total del sistema.

(d) (5 pts) Dado un sistema con con cuatro procesos y cuatro recursos (A, B, C y D) con las siguientes asignaciones en un momento dado:

.

	Asignado			
	A	B	C	D
P_0	0	0	0	1
P_1	2	0	0	0
P_2	0	1	2	0
P_3	2	1	2	0

	Máximo			
	A	B	C	D
P_0	4	5	3	2
P_1	3	2	2	0
P_2	4	4	5	7
P_3	2	2	4	0

Disponible			
A	B	C	D
0	3	2	6

Determine si el sistema se encuentra en un estado seguro. Justifique su respuesta.

Solución: Para determinar si el sistema está en un estado seguro se debe encontrar una secuencia segura. Como primer paso, se determinan los recursos que le faltan pedir a cada proceso, restando asignado a máximo:

	A	B	C	D
P_0	4	5	3	1
P_1	1	2	2	0
P_2	4	3	3	7
P_3	0	1	2	0

Dados los recursos disponibles es posible satisfacer al proceso 3 y retornar sus recursos. El vector de disponibles pasará a ser:

A	B	C	D
2	4	4	6

Con los recursos disponibles es posible satisfacer las demandas del proceso 1 y retornar sus recursos. El vector de disponibles pasará a ser:

A	B	C	D
4	4	4	6

En este punto, se comprueba que con los recursos disponibles no es posible satisfacer los recursos de ninguno de los procesos restantes, por lo que no se puede encontrar una secuencia segura luego de ejecutar P_3, P_1. Entonces, el sistema no se encuentra en un estado seguro.

Problema 2 (32 pts)

(a) Considere un sistema con direcciones virtuales de 36 bits. El sistema utiliza paginación, con un tamaño de página de 8 KiB y entradas en la(s) tabla(s) de página de tamaño 4 B, de los cuales 5 son bits de control.

I. (2 pts) ¿Cuántas páginas tiene el espacio de direccionamiento virtual del sistema ?

Solución: El espacio de direcciones virtuales tiene 64 GiB = 2^{36}. El tamaño de página es de 8 KiB (2^{13}). El número de páginas es $2^{36}/2^{13} = 2^{23}$.

II. (2 pts) ¿Cuál es el tamaño máximo de memoria física direccionable en el sistema?

Solución: Cada entrada de la tabla de página tiene 4 B, pero 5 bits se utilizan para control. Restan 27 bits para direccionamiento, que permiten referenciar 2^{27} páginas. Cada página tiene tamaño 2^{13}, entonces se direccionan $2^{27} \times 2^{13} = 2^{40}$ = 1 TiB de memoria física.

(b) (8 pts) Considere un proceso que utiliza 8 GiB de memoria contigua no estática. En términos de espacio requerido para almacenar sus tablas de páginas, ¿es más eficiente utilizar una estructura de un nivel o una estructura jerárquica de dos niveles que utilice una división lo más equitativa posible de bits para el direccionamiento?

Solución: Con una estructura uninivel necesita referenciar $2^{36}/2^{13} = 2^{23}$ páginas. Cada entrada de la tabla de páginas es de 4 B, por lo cual necesita $2^{23} \times 2^2 = 2^{25}$ B = 32 MiB.

Para una estructura jerárquica de dos niveles, se necesitan 13 bits para el offset y los 23 restantes se dividen (casi) equitativamente para tablas de primer (11) y segundo nivel (12). El proceso accede a 2^{20} páginas. El segundo nivel permite referenciar 2^{12} páginas, por lo cual se necesitan 2^8 tablas de segundo nivel. El tamaño de la tabla de primer nivel es $2^{11} \times 4$. El espacio requerido es $(2^{11} \times 4) + (2^{12} \times 2^8 \times 4) = 2^{13} + 2^{22} \approx 4$ MiB.

En términos de espacio requerido, es más eficiente usar una estructura de dos niveles.

(c) (5 pts) Para el proceso de la parte previa, cuánto mejora sobre las dos estructuras ya estudiadas (uno y dos niveles), en términos de espacio requerido para almacenar sus tablas de páginas, usar una estructura jerárquica de tres niveles que utilice una división lo más equitativa posible de bits para el direccionamiento?

Solución: Por un argumento análogo al de la parte b), los 23 bits restantes de la dirección de memoria se dividen en 8 para el tercer nivel, 8 para el segundo nivel y 7 para el primer nivel. El proceso accede a 2^{20} páginas. El tercer nivel permite referenciar 2^8 páginas, por lo cual se necesitan 2^{12} tablas de tercer nivel. Para referenciar las tablas de tercer nivel se requieren $2^{12}/2^8 = 2^4$ tablas de segundo nivel y también se requiere la tabla de primer nivel. Se necesita $(2^7 \times 4) + (2^4 \times 2^8 \times 4) + (2^{12} \times 2^8 \times 4) = 2^9 + 2^{14} + 2^{22} \approx 4$ MiB. Como la asignación se realiza de a páginas, un número importante de las entradas de la primer tabla de primer nivel quedan sin utilizar.

En términos de espacio requerido, no se mejora por utilizar una estructura jerárquica de tres niveles.

(d) (3 pts) ¿En general, qué aporta a la administración de memoria usar un esquema jerárquico con una mayor cantidad de niveles?

Solución: El esquema jerárquico con una mayor cantidad de niveles tiene dos aportes fundamentales: i) reducir el espacio de almacenamiento requerido para la tabla de páginas, dependiendo de la cantidad de memoria que utilicen los procesos y ii) permitir una gestión más eficiente de la memoria, organizando la estructura en forma de árbol y aprovechando las capacidades de utilizar almacenamiento no contiguo y eventualmente haciendo swap de páginas no utilizadas de las tablas de páginas.

(e) (3 pts) ¿Qué sucede, para los tres esquemas considerados (uno, dos y tres niveles), si el proceso de la parte b) requiere 1 MiB adicional de memoria contigua a la ya utilizada?

Solución: Si el proceso requiere 1 MiB (2^{20}) adicional de memoria contigua se necesitarían $2^{20}/2^{13} = 2^7$ páginas adicionales. Usando dos niveles se necesitaría una tabla adicional de nivel 2. Usando tres niveles se necesitaría una tabla adicional de nivel 3 y una tabla adicional de nivel 2.

(f) (3 pts) ¿Qué sucede, para los tres esquemas considerados (uno, dos y tres niveles), si el proceso de la parte b) requiere 1 MiB adicional de memoria estática?

Solución: Si el proceso requiere 1 MiB (2^{20}) adicional de memoria estática, se debe considerar que la memoria irá al stack. Se necesitarían $2^{20}/2^{13} = 2^7$ páginas adicionales (desde las direcciones más altas de memoria). Usando dos niveles se necesitaría una tabla adicional de

nivel 2 y ninguna adicional de nivel 1. Usando tres niveles se necesitaría una tabla adicional de nivel 3, una tabla adicional de nivel 2 y ninguna adicional de nivel 1 (se utilizará la última entrada de la primer página).

(g) Para la estructura jerárquica que utiliza dos niveles se introduce una TLB cuyo tiempo de búsqueda es de 12 ns (12×10^{-9} s), mientras que un acceso a memoria demanda 200 ns.

I. (2 pts) Qué factor de mejora en términos de tiempo de acceso se obtiene al usar la TLB si el proceso cuenta con un hit rate de 70 %?

Solución: La mejora en tiempos de acceso se calcula mediante la métrica EAT = tiempo de búsqueda en TLB + (hit ratio $\times$ tiempo de acceso a memoria) + (1 - hit ratio) $\times$ (3 $\times$ tiempo de acceso a memoria). El factor 3 en el último sumando se debe a que al utilizar un esquema de paginación en dos niveles se debe acceder a tres niveles de direccionamiento (tabla de primer nivel y tabla de segundo nivel, ambas almacenadas en memoria, y dirección efectiva en memoria principal).
Si no se utiliza TLB, el tiempo de acceso es 3 $\times$ tiempo de acceso a memoria = 3 $\times$ 200 ns = 600 ns. Cuando se utiliza TLB el tiempo de acceso es 12 ns + (0.7 $\times$ 200 ns) + (1 - hit ratio) $\times$ (3 $\times$ 200 ns) = 12 ns + 140 ns + 180 ns = 332 ns.
El factor de mejora es 600 ns/332 ns = 1.81.

II. (4 pts) Qué mejora se obtiene al acceder secuencialmente a todos los elementos de un arreglo de 20 000 caracteres, almacenado a partir de la dirección de memoria 0x000003000 ?

Solución: La dirección de memoria 0x000003000 corresponde a la segunda página de memoria, dado que el tamaño de página es 8KiB = 0x000002000.
Un char se almacena en un byte, por lo que se debe acceder a 20000 B contiguos, que serán almacenados en la página #2 (direcciones 0x000003000 a 0x000003FFF, 4 096 elementos), en la página #3 (direcciones 0x000004000 a 0x000005FFF, 8 192 elementos), en la página #4 (direcciones 0x000006000 a 0x000007FFF).
Al utilizar una TLB se da un miss al acceder al primer elemento, luego se dan 4 095 hits, luego un miss al cambiar la página, luego 8 191 hits, un nuevo miss al cambiar la página, y finalmente 8 191 hits.
El tiempo sin TLB es 3 $\times$ tiempo de acceso a memoria $\times$ 20 000 = 3 $\times$ 200 ns $\times$ 20000 = 12 000 000 ns = 12 ms.
Cuando se utiliza TLB el tiempo de acceso es 20 000 $\times$ 12 ns + (19 997 $\times$ 200 ns) + 3 $\times$ (3 $\times$ 200 ns) = 240 000 ns + 3 999 400 ns + 1800 ns = 4 240 800 ns.
El factor de mejora es 12 000 000 ns/4 241 200 ns = 2.83.

Problema 3 (35 pts)

Se desea implementar una función del sistema operativo para controlar el acceso a memoria por parte de los procesos. Los procesos solicitan el acceso a un determinado marco de memoria física, lo usan y luego lo devuelven. Los marcos se acceden para lectura y escritura, por lo que si un marco está siendo usado por un proceso no podrá ser usado por otro. Por limitaciones de hardware no puede haber más de 50 procesos accediendo a la memoria a la vez y cada uno de ellos no puede usar más de una página al mismo tiempo. La cantidad total de marcos disponibles no es conocida, ya que depende del hardware particular donde se ejecuta el sistema operativo. Si hay varios procesos esperando por el mismo marco, deben ser atendidos en orden de llegada.

Implemente utilizando mailboxes el código de un proceso de usuario y el del servicio del sistema operativo. No se pueden usar tareas auxiliares. Se pueden usar tipos de datos estándar (lista, cola, diccionario, pila) sin implementarlos, pero debe explicarse como funcionan los procedimientos usados.

Se dispone de las siguientes funciones auxiliares:

- `seleccionarMarco(): integer`, ejecutada por el proceso para obtener el el marco a ser accedido.
- `usarMarco()`, ejecutada por el proceso para usar el marco.

Solución:

```
var lugares: mailbox of integer
    pedidos: mailbox of [string, integer, integer]
    espera: array [1..50] of mailbox of integer

procedure cliente
    var pos, marco: integer;

    marco := seleccionar_marco();
    pos := receive(lugares);
    send(pedidos, ['entrar', marco, pos]);
    receive(espera[pos]);
    usar_marco();
    send(pedidos, ['salir', marco, pos]);
    send(lugares, pos);
end

procedure so
    var espera: map of [integer, queue of integer]
        marco, pos: integer;
        tipo: string;
        cola: queue of integer;

    for i in 1..50 do
        send(lugares, i);
    end

    while true do
        [tipo, marco, pos] = receive(pedidos);
        if tipo = 'entrar' then
            if espera.contains_key(marco) then
                espera.get(marco).insert(pos);
            else
                cola.new(integer)
                espera.add(marco, cola);
                send(espera[pos], nil);
            end
        else { salir }
            if not espera.get(marco).is_empty() then
                pos = espera.get(marco).delete();
                send(espera[pos], nil);
            else
                espera.remove(marco);
            end
        end
    end while
end

cobegin
  so();
```

```
  cliente();
  ...
  cliente();
coend
```

Se usa el tipo de datos mapa(diccionario) con procedimientos **add** para agregar un mapeo, **get** para obtener el valor asociado a partir de una clave, **remove** para borrar un mapeo y **contains_key** que dada una clave indica si existe un valor asociado a la misma.

Además se usa una cola LIFO con procedimientos **new** que crea una nueva cola, **insert** que agrega un elemento a una cola y **delete** que saca al primer elemento de una cola y lo retorna.