Departamento de Arquitectura

Segundo parcial de Sistemas Operativos

30 de junio de 2023

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

Formato:

- Indique su nombre completo y su 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 todas las hojas.
- Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva.

Dudas:

- Sólo se contestarán dudas de letra.
- No se contestarán 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. 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 (18 puntos)

(a) En relación con los sistemas de archivos:

I. [3 puntos] Describa qué es un hardlink, indique cómo se representa en sistemas de archivos indexados como ext2 para Linux y mencione al menos dos diferencias entre un hardlink y un softlink.

II. [4 puntos] Considere un sistema de archivos que soporta hasta 2 TiB de información y maneja bloques de 512 bytes. El sistema usa una estructura indexada multi-nivel. Sus inodos cuentan con 48 bits reservados para el campo tamaño y con 64 bytes de espacio reservado para almacenar una referencia a un bloque indirecto simple, una referencia a un bloque indirecto doble y múltiples referencias a bloques directos. Los bloques indirectos solo almacenan una secuencia de direcciones. ¿Cuál es el tamaño máximo de un archivo en este sistema?

Solución: Con la información planteada podemos determinar 2 límites para el tamaño máximo de un archivo:

1. La cantidad máxima de bloques que pueden ser asignados a un archivo
2. El valor máximo que puede representarse en el campo tamaño del archivo

El tamaño máximo de un archivo será el **mínimo** entre estos dos límites.

Cantidad máxima de bloques que pueden ser asignados a un archivo. El sistema de archivos soporta 2 TiB (2^{41} Bytes) de información y cada bloque maneja 512 bytes (2^9 bytes). Entonces, el campo para direccionar (o referenciar) bloques deberá contar hasta $2^{41}/2^9 = 2^{32}$ por lo que requerirá 32 bits (4 bytes) de tamaño.

La estructura de inodo cuenta con 64 bytes reservados para referencias. Una referencia indirecta simple (4 bytes) y una referencia indirecta doble (4 bytes). Los restantes 56 bytes se utilizan para referencias directas y son suficientes para $56/4 = 14$ referencias directas.
Cada bloque indirecto almacena solamente direcciones a bloques (por letra). Como cada bloque es de 512 bytes y cada dirección de bloque requiere 4 bytes, entonces cada bloque indirecto podrá contener hasta $2^9/2^2 = 2^7$ direcciones de bloques.
Entonces la cantidad máxima de bloques que pueden ser asignados son: 14 (directos) + 2^7 (indirectos simples) + $2^7 \times 2^7 = 2^{14}$ (indirectos dobles). Es decir que el tamaño máximo en bytes según este límite es $2^9 \times (14 + 2^7 + 2^{14})$ bytes.

Valor máximo que puede representarse en el campo tamaño del archivo. El valor máximo que puede representarse en el campo tamaño es 2^{48} bytes porque se cuenta con 48 bites para su representación.

(b) [4 puntos] Describa el/los método(s) libre(s) de busy waiting para realizar una E/S. Presente un pseudocódigo y explique el comportamiento esperado de una E/S a un dispositivo utilizando un método con busy waiting.

(c) [4 puntos] Exponga las condiciones de Coffman para la existencia de deadlocks. Explique la técnica de prevención de deadlocks por espera circular.

(d) [3 puntos] Explique la principal función de un hipervisor. Describa los hipervisores de tipo 2.

Problema 2 (23 puntos)

Considere un campo deportivo que cuenta con ocho canchas de variados deportes. Al llegar al campo deportivo, los jugadores deben buscar los implementos necesarios (pelotas, palos, raquetas, etc.) en la sala de utilería. En la sala de utilería pueden entrar hasta cinco jugadores a la vez, e ingresan en orden de llegada.

Existe un utilero que periódicamente debe entrar a la sala de utilería a ordenar los instrumentos. El utilero debe realizar su tarea de forma exclusiva y con prioridad sobre los jugadores.

Una vez que los jugadores tienen los implementos, se encuentran en una de las canchas para jugar. Al llegar a una cancha, cada jugador espera a que esté libre y acondicionada y luego espera que lleguen a la cancha los demás jugadores que participarán en el partido. Los jugadores esperan un máximo de 10 minutos desde el último compañero que llegó. Si en ese tiempo no llega ningún otro compañero, cancelan el partido y se van. Cuando el último jugador llega a la cancha, todos los jugadores entran a la cancha y juegan. Luego de que finaliza cada partido, la cancha debe ser acondicionada antes de permitir que se utilice nuevamente.

Modelar en ADA las tareas Jugador, Utilero, Sala y Cancha. No se permite usar tareas auxiliares. Se dispone de las siguientes funciones auxiliares:

- `ordenar_implementos()`: Ejecutada por el utilero una vez que está dentro de la sala de utilería.
- `obtener_implementos()`: Ejecutada por los jugadores dentro de la sala de utilería pera obtener los implementos.
- `otras_tareas()`: Ejecutada por el utilero cuando no está ordenando la sala.
- `acondicionar()`: Ejecutada por la cancha luego de que los jugadores terminaron de jugar un partido y se retiraron de la cancha.
- `que_cancha()`: Ejecutada por ej jugador para saber en qué cancha debe jugar. Retorna un entero (el identificador de la cancha).
- `cuantos_jugadores()`: Ejecutada por la cancha para saber cuántos jugadores son necesarios para un partido. Retorna un entero (el número de jugadores necesarios para un partido).
- `jugar()`: Ejecutada por los jugadores para jugar.

Solución:

```
program seg_parcial_2023 is

task SALA is
    entry entra_jugador();
    entry sale_jugador();
    entry entra_utilero();
    entry sale_utilero();
end SALA;
task body SALA is
    begin
        cant_jugadores := 0;
        hay_utilero := False;
        loop
            select
                when cant_jugadores = 0 =>
                    accept entra_utilero();
                    hay_utilero := true;
            or
                    accept sale_utilero();
                    hay_utilero := False;
            or
                when entra_utilero'Count = 0 and not hay_utilero and
                        cant_jugadores < 5 =>
                    accept llega_jugador();
                    cant_jugadores := cant_jugadores + 1;
            or
                    accept sale_jugador();
                    cant_jugadores := cant_jugadores - 1;
            end select;
        end loop;
    end
end SALA

task UTILERO;
task body UTILERO is
    begin
        loop
            sala.entra_utilero();
            ordenar_implementos();
            sala.sale_utilero();
            otras_tareas();
        end loop
    end
end UTILERO;


task type JUGADOR;
task body JUGADOR is
    var cancha: integer
        juega: boolean
    begin
        sala.entra_jugador();
        obtener_implementos();
        sala.sale_jugador();
```

```
        cancha := que_cancha();
        canchas[cancha].llega_jugador();
        juega := canchas[cancha].entra_jugador();
        if juega then
            jugar();
            canchas[cancha].termina_jugador();
        end
    end
end JUGADOR;

task type CANCHA;
    entry llega_jugador();
    entry entra_jugador();
    entry termina_jugador();
end CANCHA
task body CANCHA is
    begin
        cant_jugadores := cuantos_jugadores();
        loop
            cancelado := false;
            -- ingresa el primer jugador
            accept llega_jugador();
            jugadores := 1;
            -- los siguientes jugadores llegan en
            -- menos de 10 minutos o se cancela
            while ((jugadores < cant_jugadores) and (not cancelado)) loop
                select
                    accept llega_jugador();
                    jugadores++;
                or
                    delay 10
                    cancelado := true;
                end select
            end loop
            -- el partido está aprobado o cancelado
            for i := 1..jugadores loop
                accept entra_jugador()
                    return not cancelado;
                end
            end loop
            if not cancelado then
                for i := 1..jugadores loop
                    accept termina_jugador();
                end loop
                acondicionar();
            end
        end loop
end CANCHA;

var canchas: array[8] of CANCHA;

end program.
```

Problema 3 (19 puntos)

Considere el sistema de organización de las tablas de páginas utilizado en Linux, que utiliza un árbol jerárquico de tres niveles:

- Page Global Directory (PGD), el nivel de la tabla de páginas externo que tiene un único nodo,
- Page Middle Directory (PMD), el nivel de página intermedio, que contiene varios nodos y
- Page Table Entries (PTE), el nivel interno, que contiene varios nodos con punteros a las direcciones de los frames en memoria principal.

Cada nodo en la estructura de árbol ocupa un frame en memoria. La memoria física es direccionable con 64 bits, el tamaño de los frames es 4 KiB. En la tabla de páginas no se utilizan bits de control. El nivel PGD utiliza la mínima cantidad de bits posible para direccionar.

(a) [6 puntos] Se ejecuta un proceso que direcciona 4 GiB de memoria. Explique cómo se interpreta una dirección virtual y determine cuántos nodos ocupan los niveles PMD y PTE cuando el proceso usa los 4 GiB de memoria.

Solución: El proceso direcciona 4 GiB de memoria, **la dirección virtual es de 32 bits**.
Cada frame es de 4 KiB = 2^{12} bytes, **se requieren 12 bits para direccionamiento**, el campo **offset de la dirección virtual es de 12 bits**. El tamaño de las páginas es también 2^{12} bytes.
Las direcciones físicas son de 64 bits, **cada entrada en la tabla de páginas es de 8 bytes**.
Las entradas en la tabla PTE son (tamaño de página/tamaño de entrada) $2^{12}/2^3 = 2^9$, **se requieren 9 bits para direccionamiento**.
Las entradas en la tabla PMD son (tamaño de página/tamaño de entrada) $2^{12}/2^3 = 2^9$, **se requieren 9 bits para direccionamiento**.
Restan dos bits para direccionar las entradas en la tabla PGD (el mínimo número posible).
La dirección virtual es: **[2 bits PGD] [9 bits PMD] [9 bits PTE] [12 bits OFFSET]**
El número de nodos en el nivel 1 (PGD) es **uno** (lo indica la letra)
El número de nodos en el nivel 2 (PMD) es **cuatro** (las cuatro entradas direccionables con los 2 bits del PGD)
El número de nodos en el nivel 3 (PTE) es **2048** (4 nodos PMD, cada uno con $2^9 = 512$ entradas, direccionables con los 9 bits del PMD).

(b) [4 puntos] Explique cuántos niveles se necesitarían en la tabla de páginas para extender el esquema jerárquico para que un proceso pueda direccionar direccionar 2^{64} bytes de memoria. ¿Cuántas entradas tendría el nivel más externo de la jerarquía (PGD)?

Solución: El proceso requiere 64 bits para direccionar. Se necesitan 12 bits para el offset y por el mismo argumento de la parte (a), **cada nivel interno requiere 9 bits para el direccionamiento**. Se requieren floor((64 − 12) div 9) = 5 niveles internos, más el nivel PGD (7 bits restantes), en total **6 niveles**.
El nivel más externo (PGD) tiene $2^7 = 128$ entradas.

(c) [5 puntos] Para el caso del proceso que que es capaz de direccionar 2^{64} bytes de memoria, en lugar de utilizar una estructura jerárquica se utiliza una tabla de hash con 2^{12} entradas. Explique cuál es el mínimo y máximo número de frames necesarios para almacenar la tabla de hash.

Solución: La tabla de páginas hasheada está formada por **punteros a listas encadenadas**.
El mínimo tamaño necesario es cuando la tabla de hash está vacía. Los punteros son direcciones de memoria de 64 bits, **se requieren 8 bytes por entrada de la tabla de hash**. Hay 2^{12} entradas, el tamaño requerido es $2^{12} \times 2^3 = 2^{15} = 32$KiB, que corresponden a **8 frames**.

El máximo tamaño es cuando la tabla de hash está llena (contiene la información de todas las páginas). En total, existirán 2^{52} elementos en las listas encadenadas. **Cada elemento de la lista contiene el número de página (52 bits), la dirección de inicio del frame (52 bits) y un puntero al siguiente (64 bits)**. En total se requiere $2^{52} \times (7+7+8) = 22 \times 2^{52}$ bytes, que corresponden a 22×2^{40} frames. Se admite como respuesta correcta la aproximación 24×2^{40} frames.

(d) [4 puntos] Explique cuál es la entrada correspondiente en la tabla de hash para la dirección virtual 0x0000000020004003 si se utiliza la función de hash mod.

Solución: La dirección es 0x0000000020004003.
Los 52 bits para determinar la dirección en la tabla de hash son 0x0000000020004.
Existen 2^{12} (0x1000) cubetas. La entrada correspondiente es **0x20004 mod 0x1000 = 4**.