Departamento de Arquitectura

Simulacro Segundo parcial de Sistemas Operativos

30 de noviembre de 2019

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.

Problema 1 (20 pts)

(a) (2 pts) ¿Qué son y que brindan los hard links? ¿Con qué datos es necesario contar en la estructura de directorio para implementarlos? Justifique.

(b) (3 pts) Sobre la disposición de archivos en un dispositivo de almacenamiento secundario:

i. Describa brevemente los tres métodos de asignación vistos en el curso.

ii. Muestre las estructuras de datos vistas en el curso para su implementación.

(c) Se tiene un sistema de archivos híbrido. Por un lado, los directorios son almacenados usando una estrategia indexada simple con soporte para enlaces duros (hard links). Por otro lado, los archivos son almacenados usando una estructura FAT. Considere la siguiente estructura de datos:

```
1   const MAX_BLOQUES = 65536;
2   const MAX_INODOS = 8192;
3   type entrada_dir = Record
4     usado : boolean;                 // 1 bit
5     nombre : array [0..24] of char; // 25 bytes
6     tipo : (file, dir);              // 1 bit
7     inodo_num : int;                 // 2 bytes
8     fat_inicio : int;                // 2 bytes
9     tamano : int;                    // 2 bytes
10    reservado : array [0..5] of bit; // 6 bits
11  End; // 32 bytes
12
13  type inodo = Record
14    usado : boolean;                 // 1 bit
15    inodo_num : int;                 // 2 bytes
16    datos : array [0..7] of int;     // 16 bytes
17    tope : int;                      // 2 bytes
```

```
18      referencias : int;              // 2 bytes
19      reservado : array [0..6] of bit; // 7 bits
20    End; // 23 bytes
21
22    type bloque = array [0..1023] of byte; // 1024 bytes
23    type mapa_bits = array [0..MAX_BLOQUES-1] of bit;
24    type inodo_tabla = array [0..MAX_INODOS-1] of inodo;
25    type fat_tabla = array [0..MAX_BLOQUES-1] of int;
26
27    var
28      FAT: fat_table;
29      IT : inodos_tabla;
30      MB : mapa_bits;
```

Notas generales:

- Las variables FAT, IT, y MB son globales.
- En la estructura entrada_dir el campo inodo_num es utilizado únicamente cuando el tipo es *dir*. El atributo fat_inicio y tamaño son utilizados únicamente cuando el tipo es *file*.
- El directorio raíz es el inodo con índice 0 (cero).
- En FAT el valor −1 indica el fin de archivo.
- El campo referencias es utilizado para contar la cantidad de enlaces.
- En MB el valor 0 (cero) indica un bloque libre y 1 indica un bloque ocupado.

Se dispone de los siguientes procedimientos:

- `Procedure leerBlq(blq_num: int, Var buff: bloque) : boolean`
  Lee de disco el bloque blq_num, pasado como parámetro, y carga el contenido leído en el parámetro de salida buff. Retorna el éxito de la ejecución de la operación.
- `Procedure escrBlq(blq_num : int, buff : bloque) : boolean`
  Escribe en el bloque blq_num, pasado como parámetro, la información que se encuentra en el parametro buff. Retorna el éxito de la ejecución de la operación.
- `Procedure parteCamino(camino: array of char, var base: array of char, var resto: array of char);`
  Retorna la primera parte de la ruta en el parámetro base, y las restantes partes en resto. Por ejemplo: `parteCamino('/users/so/a.txt', 'users', '/so/a.txt')`
  `parteCamino('/a.txt', 'a.txt', '')`
  `parteCamino('/', '', '')`

Se pide:

i. (3 pts) Dada la estructura del sistema de archivos planteado, mencione y describa brevemente tres problemas de correctitud que pueden darse.

ii. (7 pts) Implementar una función que busque un directorio o archivo:
`Function search(cam: array of char): entrada_dir;`
Donde `cam` es la ruta completa al elemento del sistema de archivos buscado. Retorna una entrada de directorio que corresponde al elemento buscado y `null` en caso que la operación no se ejecute con éxito.

iii. (5 pts) Implementar una función que reemplace el contenido a un archivo:
`Function zeros(camArchivo: array of char):boolean;`
Donde `camArchivo` es el camino de un archivo cuyo contenido debe ser sobreescrito totalmente con ceros. Retorna `true` si la operación fue ejecutada con éxito. Esta función no crea archivos.

Problema 2 (20 pts)

(a) (3 pts) Describa las acciones del sistema operativo cuando ocurre un cambio de contexto

1. entre hilos.
2. entre procesos.

(b) (2 pts) ¿En qué momentos es invocado el planificador de corto plazo del sistema operativo?.

(c) Se tiene un sistema operativo multiprogramado en el cual se dispone de 2 procesadores. El planificador del sistema operativo utiliza una estrategia de planificación Round-Robin con quantum de 10 ms. Este sistema maneja hilos con modelo Mx1, utilizando una planificación Round-Robin con quantum de 5 ms a nivel de usuario.

Se tienen los siguientes procesos:

P1	**P2**	**P3**
Ejecuta 10ms	Ejecuta 5ms	Ejecuta 10ms
Bloquea 5ms	Bloquea 10ms	Bloquea 10ms
Ejecuta 10ms	Ejecuta 20ms	Ejecuta 10ms

Se tiene que **P1** cuenta con 2 hilos que ejecutan concurrentemente de forma que cada hilo ejecuta todo el código del proceso **P1**. En cambio, **P2** y **P3** cuentan con un solo hilo de ejecución. En el instante de tiempo inicial ($t = 0$) la cola de listos contiene a **P1**, **P2** y **P3** (en este orden).

Se pide:

i. (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.

ii. (10 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.

iii. (3 pts) Calcule y defina porcentaje de uso de CPU, tiempo de retorno y tiempo de espera de cada hilo.

Problema 3 (20 pts)

(a) (3 pts) Explique qué desventaja particular tienen los hilos implementados a nivel de usuario en un sistema con paginación bajo demanda pura.

(b) (3 pts) En un sistema con memoria virtual, describa el algoritmo de reemplazo de segunda chance (second-chance algorithm).

(c) Se tiene un sistema donde cada proceso requiere de 2050 KiB para almacenar su codigo, datos y heap, mientras que para la pila requiere de 8 KiB de espacio de memoria. El sistema operativo utiliza un gestor de memoria implementada con memoria virtual utilizando un modelo de paginación bajo demanda con las siguientes características:

- Direcciones virtuales de 48 bits.
- Tres niveles de paginación.
- Entradas de tabla de páginas de 16 bytes y páginas de 512 bytes.
- Las tablas de segundo y tercer nivel ocupan una página de memoria.

i. (4 pts) Determine como se distribuyen los bits cada dirección indicando para que se utilizan.

ii. (10 pts) ¿Cuántas páginas de 2do y 3er nivel son necesarias para almacenar cada proceso de usuario en memoria?