Sistemas Operativos

Departamento de Computación – FCEyN – UBA
Primer cuatrimestre de 2017

Nombre y apellido: _____

Nº orden: _____ L.U.: _____ Cant. hojas: _____

1	2	3	4	Nota

Segundo parcial – 13 de junio de 2017

ACLARACIONES: 1) **Numere** las hojas entregadas. Esta hoja se entrega y es la hoja cero. Complete en la primera hoja la cantidad total de hojas entregadas (sin contar el enunciado). 2) Realice cada ejercicio en **hojas separadas** y escriba **nombre, apellido y L.U. en cada una**. 3) Cada ejercicio se califica con **Bien**, **Regular** o **Mal**. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se califican globalmente. El parcial se aprueba con 2 ejercicios bien y a lo sumo 1 mal/incompleto. 4) El parcial **NO** es a libro abierto.

Justifique *adecuadamente* cada una de sus respuestas.

Ejercicio 1.

Un sistema operativo nuevo quiere implementar el sistema de archivos *FATino*, que combina algunos conceptos de FAT y otros de inodos. En él, los primeros tres bloques de un archivo se encuentran descriptos en una FAT, mientras que la cuarta entrada de la FAT, en caso de ser necesaria, apunta a un inodo tradicional con 1 nivel de acceso directo y el resto, indirectos. Suponga que el sistema cuenta con bloques de 8 KB, que la FAT tiene 128 MB y cada entrada de la tabla FAT tiene 16 bytes.

a) ¿Cuál es el tamaño máximo de archivo que admite este sistema? Justifique la respuesta mostrando los cálculos que la sustentan.

b) ¿Para archivos de hasta qué tamaño está especialmente diseñado *FATino*? Justifique la respuesta mostrando los cálculos que la sustentan.

c) Describa el pseudo código de una función que obtenga todos los bloques de un archivo en *FATino*.

Ejercicio 2.

Se desea implementar una API genérica que permita a los programas interfacear con drivers de dispositivos de cocina. En esta primera etapa se contemplan sólo los siguientes tipos de equipamiento con las funcionalidades mencionadas a continuación:

f1) hornos eléctricos (entregar X potencia durante Y minutos),

f2) licuadoras (encender motor),

f3) licuadoras (apagar motor),

f4) sensores de temperatura (la temperatura actual es X),

f5) heladeras inteligentes (se acabó el producto cuyo código de barras es X) y

f6) heladeras inteligentes (se pudrió el producto cuyo código de barras es Y).

Responda las siguientes preguntas (y justifique su respuesta):

a) ¿Qué mecanismo/s de E/S debería utilizar la API y por qué?

b) El kernel del SO provee la función `u8 inb(unsigned int direccion)` que lee un byte de un registro de manera bloqueante. ¿Para cuáles de las funciones se puede utilizar?

Ejercicio 3.

Considere la siguiente secuencia de referencias a páginas:

4, 7, 2, 5, 7, 8, 2, 2, 4, 1, 7, 9

a) Realice el seguimiento de cada uno de los algoritmos de reemplazo listados abajo considerando que el sistema cuenta con 4 frames (todos ellos incialmente libres). Además, indique el *hit rate* para cada algoritmo.

   i. FIFO
   ii. LRU
   iii. Segunda oportunidad

b) Para cada algoritmo de reemplazo de páginas del ítem anterior, exhiba un escenario donde su performance sea superior a alguno de los otros dos.

Ejercicio 4.

En cierto sistema los usuarios acumulan créditos y eso les permite ejecutar diversas funciones, que tienen un costo que va variando. El siguiente fragmento de código es el encargado de ejecutar la función `f()` si los créditos que le quedan al usuario superan el costo de la función. La variable `costo` contiene el costo que el sistema le asignó a la función `f()` y `creditos` es un puntero a la cantidad de créditos que tiene el usuario.

```
void seleccionar_funcion(unsigned int opcion) {
    func (*funciones)[1000] = [ &f1, &f2, ..., &f1000 ];
    int costos[1000] = [10, 100, 10000, 100000, ..., 10000000];
    if (opcion>=1000) { return; // error };

    usuario *usuario_actual = get_user();
    creditos = usuario_actual->creditos;
    return ejecutar_si_quedan_creditos(&creditos, costos[opcion-1], funciones[opcion-1]);
}

void ejecutar_si_quedan_creditos(int *creditos, int costo, func *f) {
    unsigned int saldo = (*creditos) - costo;
    if (saldo == 0) {
        return; // No tiene crédito.
    }

    // Sí tiene.
    (*f)(); // Se ejecuta f().
    *creditos = saldo; // Se actualiza el saldo.
}
```

El usuario controla la invocación a la función `seleccionar_funcion()`, a la que puede llamar todas las veces que desee, intentado ejecutar cualquiera de las opciones disponibles, todas las veces que quiera.

El código mostrado tiene un problema de seguridad con 2 consecuencias. Determine cuáles son. **Pista**: considere los tipos de las variables numéricas involucradas.