Examen parcial de Sistemas Operativos 1

Evaluación parcial 2

15 de junio del 2022

La prueba parcial tiene una duración de 2 horas (15:00-17:00). El parcial tiene las siguientes condiciones y restricciones. A partir de la página siguiente se muestra el enunciado de la prueba.

- **El enunciado de la prueba se publicará en el campus virtual el mismo día de la prueba a las 8:00,** dónde el alumnado podrá prepararse la prueba con el material del campus virtual, Internet, etc.
- La prueba parcial se debe responder de forma **individual**.
- **Las respuestas de la prueba pueden ser realizadas tanto en castellano como en catalán**.
- **NO se podrá traer apuntes a clase**.
- **A cada una de las páginas entregadas se debe indicar el nombre completo y NIUB**.
- Se utilizarán **4 páginas como máximo** (entregadas por el profesorado) para responder las preguntas de la prueba parcial. Es importante comentar y/o razonar las respuestas a las preguntas. Es suficiente responder de forma breve a cada una de las preguntas (con 3 o 4 frases) para que se dé la respuesta como válida. No es necesario extenderse.
- A la hora de evaluar se valorará el hecho de que el estudiante haya entendido lo que se dé como respuesta. Se recomienda pues responder a las preguntas con información que el estudiante comprenda. El profesorado podrá requerir al estudiante que explique alguna de sus respuestas posteriormente a la prueba.
- En caso necesario, el profesorado puede proporcionar un justificante de asistencia a la prueba.
- **Cualquier intento de copia comportará la aplicación de la normativa académica general de la Universidad de Barcelona y el inicio de un proceso disciplinario.**

Problema 1 (4 puntos en total, 0.5 puntos por pregunta)

A continuación se muestra un código en C que compila y funciona correctamente.

```
1  int main(int argc, char **argv)
2  {
3    struct stat st;
4
5    int i, fd, len, countA, countB;
6    char *file_memory;
7
8    stat(argv[1], &st);
9    len = st.st_size;
10   countA = 0;
11   countB = 0;
12
13   fd = open(argv[1], O_RDWR, S_IRUSR | S_IWUSR);
14
15   file_memory = mmap(0, len, PROT_READ | PROT_WRITE,
16               MAP_SHARED, fd, 0);
17
18   close(fd);
19
20   for(i = 0; i < len; i++)
21     if (file_memory[i] == 'a' || file_memory[i] == 'A')
22       countA++;
23
24   printf("Number of vowels (a/A) in the file: %d \n", countA);
25
26 // munmap(file_memory, len);
27
28   for (i = 0; i < len; i++)
29     if (file_memory[i] == 'b' || file_memory[i] == 'B')
30       countB++;
31
32   printf ("Number of vowels (b/B) in the file: %d \n", countB);
33
34  munmap(file_memory, len);
35   return 0;
36 }
```

Al ejecutar el código anterior (./mmap file.txt) se muestran los siguientes mensajes por pantalla.

```
Number of vowels (a/A) in the file: 520
Number of vowels (b/B) in the file: 90
```

En los siguientes apartados se analizará paso por paso lo que hace el código. Se recomienda leer primero todo el enunciado antes de comenzar a responder las preguntas.

a) Comenta brevemete qué es lo que hace la línea 15 del código.

b) Al realizar el `mmap`, ¿se carga todo el fichero a memória? Razona tu respuesta.

c) ¿Cuáles son las ventajas de `mmap` en el código con respecto a utilizar las llamadas a sistemas `read/write`?

d) ¿Cómo es posible que podamos acceder al contenido del fichero abierto con identificador fd (línea 13) después de realizar un `close` del mismo (línea 18)? Razona tu respuesta.

e) En las líneas 20-22 y 28-30 se contabiliza el número de ciertas vocales de un fichero. Si se realizaran cambios en el código y se reemplazaran unas vocales por otras (As por Bs), ¿cada vez que se realiza un cambio en una vocal se guarda dicha modificación a disco? Razona tu respuesta.

f) ¿Qué es lo que hace la línea 34 del código (`munmap`)? ¿Por qué es necesario ejecutar esta línea? Es decir, ¿cuál es el objetivo de ejecutar esta línea?

g) Si descomentamos la línea 26, ¿será posible acceder al fichero en las líneas 28-30 para contar las vocales b/B? Razona tu respuesta.

h) Si modificamos el código y reservamos memoria dinámica con un `malloc`, ¿seria posible escribir en esta parte de la memoria después de realizar un `free`? Razona tu respuesta.

Problema 2 (2.25 puntos en total, 0.75 puntos por pregunta) Supongamos que un sistema operativo con un solo procesador utiliza un algoritmo de planificación multi-level feedback queue (MFQ). En un instante determinado hay múltiples procesos en la lista de "ready" y sólo uno ejecutándose. El usuario admin ejecuta un nuevo proceso y el sistema operativo debe decidir cómo realizar su planificación.

a) El MFQ tiene diversas colas con rebanadas de tiempo de 10ms, 20ms, 40ms y 80ms, cada una con prioridades diferentes, que corresponden a colas round robin. Describe en que consiste la planificación round robin.

b) Al comenzar a ejecutarse el proceso este empieza a leer datos de disco (que no están en el buffer interno del sistema operativo). ¿En qué nivel de la cola situará el sistema operativo al proceso? ¿En uno con una rebanada de tiempo baja (10ms) o alta (80ms)? Razona la respuesta.

c) Una vez el proceso ha cargado los datos de disco comienza a realizar operaciones matemáticas sobre ellas sin acceder a disco. ¿El sistema operativo moverá este proceso a otro nivel de la cola? Razonar la respuesta indicando, en su caso, a qué cola lo moverá el sistema operativo.

Problema 3 (3.75 puntos en total, 0.75 puntos por pregunta) Comentar las siguientes afirmaciones de los sistemas operativos. En caso de que la frase sea incorrecta, indicar cómo funciona realmente y razonar la respuesta poniendo ejemplos si se cree conveniente. En caso que sea correcta, comentar la razón por la cual crees que es correcta.

a) Un proceso que se está ejecutando en un momento determinado puede pasar a la cola de "waiting" después de una interrupción.

b) Las señales SIGUSR1 y SIGUSR2 pueden ser utilizadas para matar un proceso en ejecución.

c) La lectura de un fichero grande (por ejemplo 4GB) mediante la llamada a sistema `read` es más efectiva (menor tiempo) que con la libreria de usuario `fread`.

d) En el momento de reservar memoria dinámica con un `malloc` se asignan las direcciones tanto en el espacio virtual como el físico.

e) Las direcciones en el espacio físico y virtual se relacionan (mapean) mediante un marco de página.