SESIÓN TP8

Sistemas Operativos 1
Mayo 2025

1 INTRODUCCIÓN

El objetivo de esta sesión es analizar dos familias de funciones que se han visto en las sesiones de teoría para la manipulación de ficheros. En particular, se analizarán las llamadas al sistema (`read, write`) y las funciones de librería de usuario (`fread, fwrite`). Estas últimas, realizan internamente llamadas al sistema `read/write`. La idea es analizar la "eficiencia" de cada una de estas funciones a la hora de manipular ficheros.

2 FUNCIONES A ANALIZAR

La sintaxis de la función `read` es:

```
read(fd, vector, nbytes);
```

donde `fd` es el descriptor de fichero (que se obtiene a partir de la llamada a sistema `open`), `vector` es la posición de memoria donde se quieren guardas los datos leídos, y `nbytes` es el número de bytes a leer, que llamaremos "medida del bloque".

La sintaxis correspondiente a la función `fread` es:

```
fread(vector, mida_element, nombre_elements, fp);
```

donde vector es la posición de memoria en la que se guardaran los datos leídos, mida_element es la medida de cada uno de los elementos a leer, nombre_elements es el número de elementos a leer y fp es el fichero abierto con la llamada al sistema fopen. Esta función lee un total de mida_elements x nombre_elements bytes de disco.

Nos podemos realizar diverses preguntas a la hora de utilizar estas funciones a la hora de utilizarlas durante la programación. Tanto si utilizamos llamadas al sistema como funciones de la librería de usuario,

¿Cuál es la eficiencia asociada si (read o fread) le lee un fichero de byte en byte o en bloques de tamaño 4096 bytes?

¿Cuál de las dos estrategias requiere más tiempo? ¿Por qué?

3 MATERIAL DISPONIBLE

Se pide comenzar ejecutando el código `create_big_file.c`. Este programa **generará un archivo de 4GBytes** llamado `big_file.bin`. Este archivo grande nos permitirá poder medir la eficiencia de los dos tipos de funciones que disponemos para leer o escribir en un archivo.

Se proporcionan además dos programas en C, uno que utiliza llamadas a sistema y el otro las de usuario, para leer/escribir un archivo. Ambos programas utilizan un parámetro de ejecución para especificar el número de bytes, y el tamaño de bloque a leer en cada iteración hasta que se ha leído todo el archivo. El objetivo es ver cuál es la influencia del número de bytes que se lee en cada iteración en el tiempo de ejecución de la aplicación.

Analizar los dos códigos. Ambos códigos leen de la entrada estándar y escriben en la salida estándar hasta que se ha leído todo el archivo.

Para ejecutar cada uno de los experimentos utiliza la siguiente instrucción:

```
time ./io_system_call 1024 < big_file.bin > /dev/null
```

En ese caso `io_system_call` es el código que utiliza llamadas a sistema para acceder a disco. El valor 1024 es el tamaño de bloque con el que se ejecutará la aplicación.

Observa que se utiliza redirección para indicar que la entrada estándar es el archivo `big_file.bin`. La salida estándar es `/dev/null`, una especie de "agujero negro". Todo lo que se escribe se pierde. Escribiendo todo en este "agujero negro" no se crea ningún archivo a disco lo que puede sesgar los resultados de los experimentos (ya que no podemos asegurar que en cada experimento el archivo se cree en las mismas posiciones de disco).

Como hemos visto en otras sesiones, la instrucción `time` mide el tiempo de ejecución de la aplicación. Por ejemplo, nos puede dar un resultado como

```
real 0m34.055s
user 0m12.902s
sys 0m21.145s
```

En este caso se indica que el tiempo de ejecución en modo usuario (“user”) es de 12.9 segundos y que el tiempo de ejecución en el sistema operativo (en modo núcleo; “sys”) es de 21.14 segundos.

El valor "real" nos indica el tiempo cronológico que ha tardado la aplicación en ejecutarse: desde que se ha empezado a ejecutar hasta que ha finalizado de ejecutarse, incluyendo el hecho de que ha habido cambios de contexto en otros procesos de por medio, el tiempo que el proceso ha sido bloqueado, etc.

Tener en cuenta que la máquina virtual (*guest*) es un proceso de la máquina real (*host*) y que los tiempos medidos pueden estar sesgados ya que el sistema operativo *guest* puede poner a dormir la máquina virtual en cualquier momento (como cualquier otra aplicación).

4 EJERCICIO A ENTREGAR

Se pide ejecutar los experimentos con diferentes tamaños de bloque para ver cuál es la influencia del tamaño del bloque en la lectura del disco. Esta operación debe realizarse tanto con el código que utiliza llamadas al sistema como el que hace llamadas a la librería de usuario.

Realiza una tabla para que se vean claro las diferencias de ejecución en el tiempo de usuario, de núcleo y tiempo real en los experimentos que realice. Para realizar las pruebas se recomienda utilizar tamaños de bloque de 64 bytes, 128 bytes, 512 bytes, 1024 bytes, 4096 bytes, 16384 bytes, 65336 bytes y 262144 bytes. Al realizar las pruebas se recomienda ejecutar varias veces el código para un mismo valor para ver los márgenes de tiempo de ejecución que se obtienen (mostrar media y deviación estándar).

Es conveniente que el tamaño del archivo sea mayor que el de RAM asociado a su máquina virtual (o la que ejecute estas pruebas si las ejecuta directamente a la máquina nativa). Esto se debe a que, al leer por primera vez el archivo, los datos asociados se almacenan en el buffer interno del sistema operativo (“caché de disco"). El sistema operativo aprovecha toda la RAM disponible que puede ser buffer interno. Si el tamaño del archivo es inferior a la RAM disponible, las siguientes veces que se lea del archivo ya no será necesario leerlo de disco, sino que se accederá directamente a la copia que hay memoria. En cambio, si el tamaño del archivo es mayor que la RAM disponible, el sistema operativo deberá acceder a disco para leer los datos.

A continuación, se debe responder a las siguientes preguntas.

1. ¿Qué diferencias de tiempo de ejecución (tiempo real, tiempo de usuario y de sistema) se observa al realizar las pruebas con las funciones `read/write` y `fread/write`? Razona la respuesta.
2. ¿Por qué en algunos casos existen diferencias claras de tiempo de ejecución entre las dos formas de leer los datos de disco? ¿Por qué el tiempo real es mayor que la suma del tiempo de usuario y el tiempo de sistema?
3. ¿En este contexto, el tipo de los datos que se están leyendo del fichero ¿podría influir en los tiempos de ejecución? Razona tu respuesta.
4. Supongamos que cada experimento se ejecutara N veces en paralelo utilizando forks. ¿Creéis que tendría algún impacto en los resultados? Razona tu respuesta.

5 PROCESO DE EVALUACIÓN

Se pide entregar un **informe en formato PDF** por cada grupo (grupoXX.pdf, donde XX es el número de grupo de prácticas) de máximo tres páginas (sin incluir la portada/índice si se desea incluirlos). Dicho informe debe incluir una **introducción**, descripción de **pruebas realizadas** incluyendo una tabla con los tiempos ejecución de usuario, de sistema y real obtenidos para las funciones `read/write` y `fread/fwrite` utilizando los tamaños de bloque antes indicados (64 bytes, 128 bytes, 512 bytes, 1024 bytes, 4096 bytes, 16384 bytes, 65336 bytes y 262144 bytes), así como **respuestas a las preguntas** indicadas en el apartado anterior y **conclusiones y valoración personal**. Recuerda vaciar la caché para cada experimento.

Se recomienda ejecutar todos los experimentos en un único ordenador. Si tiene acceso a ordenadores con discos diferentes (como un disco magnético y un disco SDD) también es interesante que haga pruebas con estos dos discos. Sin embargo, a nivel de evaluación es más que suficiente hacer las pruebas con un único ordenador.

En el informe se le pide incluir información sobre:

1. La plataforma en la que ha ejecutado las pruebas (máquina virtual, máquina nativa)
2. RAM de la máquina virtual (o la máquina nativa) en la que se han ejecutado las pruebas. Indique el tamaño del archivo generado.
3. Información (si la conoce) respecto al disco en el que se han hecho las pruebas: magnético, SSD, etc.

Se recomienda ser breve y conciso al comentar los resultados obtenidos.

La evaluación por parte del profesor será entre 0 y 10. Para realizar esta evaluación se utilizarán criterios equivalentes a los que se realizan en los informes que entrega a prácticas.

La fecha límite de entrega es mañana jueves 8 de mayo hasta las 15h mediante la tarea asociada al campus virtual. Es muy importante que haga las entregas con el nombre de archivo correcto.