Sistemes Operatius 1

Sessió de problemes 7 –22 d'abril del 2020

Objectius

L'objectiu d'aquesta sessió és analitzar dues famílies de funcions que s'han vist a classe de teoria, en particular, `read/write` i `fread/write`. Totes dues serveixen per llegir i escriure dades de disc. Les primeres són crides a sistema i les segones són crides a una llibreria d'usuari que internament fan les crides a sistema a `read/write`.

En aquesta sessió l'objectiu és analitzar la "eficiència" de cadascuna de les funcions. En particular, la signatura de la funció `read` és

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

on `fd` és el descriptor de fitxer, `vector` és lloc de memòria on es volen emmagatzemar les dades llegides i `nbytes` és el nombre de bytes a llegir, que aquí anomenarem "la mida de bloc". El descriptor de fitxer s'obté mitjançant la crida a sistema open.

La signatura de la funció fread és

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

on `vector` és el lloc de memòria es volen emmagatzemar les dades llegides, `mida_element` és la mida de cadascun dels elements a llegir, `nombre_elements` és el nombre d'elements a llegir i `fp` és el fitxer obert amb la crida a `fopen`. Aquesta funció llegeix un total de `mida_element x nombre_elements` bytes de disc.

Ens podem realitzar diverses preguntes a l'hora d'utilitzar aquestes funcions: per exemple, si faig servir les crides a sistema, quina és la eficiència associada si es llegeix un fitxer de byte en byte o en blocs de mida de 4096 bytes? Triga el mateix? De forma equivalent, si faig servir la funció fread és igual si llegeixo un fitxer de byte en byte que en blocs de 4096 bytes?

Eines que es proporcionen

Es demana, primer de tot, executar el codi `create_big_file.c`. Aquest programa generarà, al vostre ordinador, un fitxer de 4GBytes anomenat `big_file.bin`. Sí, anem a crear un fitxer realment gran per poder mesurar l'eficiència dels dos tipus de crida que disposem per llegir o escriure a un fitxer.

Es proporcionen a més dos programes en C, un que utilitza crides a sistema i l'altre les d'usuari, per llegir/escriure un fitxer. Els dos programes utilitzen un paràmetre d'execució per especificar el nombre de bytes, la mida de bloc, a llegir a cada iteració fins que s'ha llegit tot el fitxer. L'objectiu és doncs veure quina és la influència del nombre de bytes que es llegeix a cada iteració en el temps d'execució de l'aplicació.

Observeu els dos codis. Els dos codis llegeixen de l'entrada estàndard i escriuen a la sortida estàndard fins que s'ha llegit tot el fitxer.

Per executar cadascun dels experiments feu servir la següent instrucció

```
time ./io_system_call mida_bloc < nom_fitxer > /dev/null
```

En aquest cas `io_system_call` és el codi que utilitza crides a sistema per accedir a disc. El paràmetre mida_bloc és la mida de bloc amb la qual voleu executar l'aplicació (64, 512, 1024, ...) i nom_fitxer és el nom del fitxer que s'ha generat fa un moment, `big_file.bin`.

Aquesta instrucció executa l'aplicació `io_system_call` fent que l'entrada estàndard vingui del fitxer indicat a la redirecció i que la sortida vagi a `/dev/null`, una mena de "forat negre". Escrivint tot en aquest "forat negre" no es crea cap fitxer a disc cosa que pot esbiaixar els resultats dels experiments (ja que no podem assegurar que a cada experiment el fitxer es creï a les mateixes posicions de disc).

La instrucció `time` mesura el temps d'execució de l'aplicació. Per exemple, ens pot donar un resultat com

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

En aquest cas ens diu que el temps d'execució en mode usuari és de 12.9 segons i que el temps d'execució en el sistema operatiu (en mode nucli) és de 21.14 segons.

El valor "real" ens indica el temps cronològic que ha trigat l'aplicació en executar-se: des que s'ha començat a executar fins que ha finalitzat d'executar-se, incloent el fet que hi ha hagut canvis de context a altres processos pel mig, el temps que el procés ha estat bloquejat, etc.

Per fer les proves es proposa que feu servir mides de bloc de 64 bytes, 1024 bytes, 4096 bytes, 16384 bytes, 65336 bytes i 262144 bytes.

Hi ha un detall més. Hem d'assegurar que els experiments amb les diferents mides de block s'executin amb les mateixes "condicions". El vostre sistema operatiu (sigui Linux o Mac) utilitza una memòria cau (i.e. caché) a la RAM per emmagatzemar-hi els fitxers que s'han llegit, de forma que futures lectures al mateix fitxer siguin més ràpides. I això esbiaixaria els experiments amb les diferents mides de bloc!

Per assegurar aquesta que els experiments s'executen amb les mateixes condicions, cal buidar doncs aquesta memòria cau abans d'executar cada experiment. A Linux hi ha una solució senzilla si teniu accés d'administrador (root). En concret, executeu

```
sync && echo 3 > /proc/sys/vm/drop_caches
```

Aquesta instrucció buida la memòria cau per assegurar que no hi ha cap part del fitxer a memòria RAM. D'aquesta forma ens assegurem que cada experiment s'executi de forma en les mateixes condicions que els altres experiments.

En cas que no tingueu accés a administrador o no tingueu Linux, caldrà que reinicieu l'ordinador per a cada experiment, evitant obrir el fitxer `big_file.bin` abans d'executar l'experiment.

Exercici a entregar

Es demana executar els experiments amb diferents mides de bloc per veure quina és la influència de la mida del bloc en la lectura del disc. Feu-ho pel codi que utilitza crides a sistema com el que fa crides a la llibreria d'usuari. Feu una taula perquè es vegin clar les diferències d'execució en el temps d'usuari, de nucli i temps real als experiments que feu.

Per fer les proves feu servir mides de bloc de 64 bytes, 1024 bytes, 4096 bytes, 16384 bytes, 65336 byes i 262144 bytes. Recordar buidar la memòria cau per a cada experiment. En cas que no es pugui executar la instrucció per buidar la memòria cau (sigui perquè no es té accés a administrador o es disposa d'un ordinador Mac) caldrà re-iniciar l'ordinador cada cop que es vulgui fer una nova prova.

Quines diferències veieu entre fer les proves amb les funcions read/write i fread/write? Podeu treure algunes conclusions de les proves que heu fet amb els dos tipus de funcions? Podeu explicar per què, per exemple, en alguns casos, el temps real és molt més gran que la suma del temps d'usuari i el temps de sistema?

Avaluació

Es demana entregar un petit informe de **màxim** tres pàgines (sense incloure la portada en cas que la vulgueu incloure) que inclogui, en taules, els temps execució d'usuari, de sistema i real obtinguts per a les funcions read/write i fread/fwrite fent servir les mides de bloc esmentades abans. Recordeu buidar la memòria cau per a cada experiment. Comenteu els resultats obtinguts.

Executeu tots els experiments en un únic ordinador. En cas que tingueu accés a ordinadors amb discs diferents (per exemple, un disc magnètic i un disc SDD) també és interessant que feu proves amb aquests dos discos. A nivell d'avaluació, però, és més que suficient fer les proves amb un únic ordinador.

Sigueu breus i concisos a en comentar els resultats obtinguts. La data límit per entregar aquest document és aquest divendres 24 d'abril.

Entregueu un únic informe per parella. L'avaluació per part del professor serà de 0, 2.5, 5, 7.5 o 10. Per fer aquesta avaluació es faran servir criteris equivalents als que es fan als informes que entregueu a pràctiques. En altres paraules, es valorarà més el fet que interpreteu i comenteu els resultats obtinguts amb un sol ordinador que el fet que feu experiments amb múltiples ordinadors i no feu comentaris concloents respecte els resultats obtinguts.