MEDIDA DEL TIEMPO Y LA MEMORIA DE UN PROGRAMA

TIEMPO INSTANTE | MEMORIA INSTANTE | TIEMPO TOTAL | MEMORIA TOTAL

Medida del tiempo tardado en una operación

Para medir de forma precisa el tiempo que tarda en ejecutarse cierta operación, se debe usar la función **gettimeofday**, que se encuentra en la librería "sys/time.h". Esta función ofrece teóricamente una precisión de microsegundos (0,001 milisegundos). Su sintaxis es la siguiente:

int **gettimeofday** (struct timeval *_tp_, NULL);

Siendo *timeval* un registro con 2 campos: int **tv_sec**, int **tv_usec**, que indican los segundos y microsegundos, respectivamente, transcurridos desde el 1 de enero de 1970.

Ejemplo de uso para medir la velocidad del usuario en leer un texto:

```
#include <stdio.h>
#include <sys/time.h>
int main(void)
{
    struct timeval ti, tf;
    double tiempo;
    gettimeofday(&ti, NULL);   // Instante inicial
    printf("Lee este mensaje y pulsa ENTER\n");
    getchar();
    gettimeofday(&tf, NULL);   // Instante final
    tiempo= (tf.tv_sec - ti.tv_sec)*1000 + (tf.tv_usec -
ti.tv_usec)/1000.0;
    printf("Has tardado: %g milisegundos\n", tiempo);
}
```

Medida de la memoria ocupada en cierto instante

No existe ninguna función estándar de las librerías de C/C++ que devuelva la memoria ocupada por un programa en un instante concreto*. Pero podemos aprovechar la información que el sistema operativo almacena dentro del directorio **/proc****.

La lectura de esta información no es trivial, así que ofrecemos a los alumnos la librería **memcount**. Esta librería contiene dos archivos: "memcount.hpp" - fichero de cabecera para incluir en los sitios que usen la librería; "memcount.cpp" fichero de implementación de las funciones, que debe ser compilado junto con los otros módulos del programa.

La librería contiene dos funciones (sobrecargadas):

int **mem_total** (void);

Devuelve la memoria total usada por el programa (memoria de datos + memoria de pila) en kilobytes. No necesita ningún parámetro.

int **mem_total** (int &data, int &stack);

Devuelve la memoria total usada por el programa (memoria de datos + memoria de pila) en kilobytes. Además, almacena en **data** la memoria de datos usada, y en **stack** la memoria de pila, ambos en kilobytes.

La precisión de estas funciones puede depender del sistema operativo (en función del tamaño de los bloques de memoria), variando típicamente entre 4 y 32 Kbytes.

Ojo: la memoria de un programa debe tomarse como la **máxima cantidad** de memoria que necesita a lo largo de toda su ejecución. No obstante, si sabemos que la máxima ocupación será al finalizar cierta operación (por ejemplo, como puede ocurrir con las operaciones para insertar datos), basta con llamar a **mem_total** al acabar dicha operación.

* Aunque en teoría sí que existe... La función estándar de C **getrusage**, en la librería "sys/resource.h", tiene por objetivo indicar los recursos (tiempo, memoria y otros) consumidos por un proceso. Sin embargo, la implementación en Linux de la librería no rellena la información del uso de memoria.

** Por este motivo, el método de medición que se explica aquí no es compatible con MS Windows.

Medida del tiempo total de ejecución de un programa

El comando **time** de Linux permite obtener el tiempo total de ejecución de un programa. Su sintaxis es muy sencilla: **time** seguido del comando cuya ejecución queremos medir (con los parámetros correspondientes).

El resultado de **time** se escribe en la salida de error estándar (en C **stderr**, en C++ **cerr**, y en línea de comandos "**2>**"). Por ejemplo, si ejecutamos desde la línea de comandos*:

```
>> (time ls) 2> salida.txt
```

Podemos obtener en "salida.txt" un resultado del tipo:

```
real 0m0.037s
user 0m0.004s
sys 0m0.008s
```

El valor **real** indica el tiempo total transcurrido en ejecutar el comando. Por ejemplo, si hay otros procesos en el sistema, se contará también el tiempo de los mismos. El valor **user** se refiere al tiempo de CPU del proceso en cuestión; por lo tanto, se excluye el tiempo de otros procesos o de los retardos del disco. El valor **sys** es el tiempo de CPU en las llamadas al sistema del proceso. Idealmente, si no hubieran otros procesos y la lectura de disco fuera inmediata, tendríamos que: **user** + **sys** = **real**.

* Los paréntesis no se pueden suprimir en el comando indicado, ya que en otro caso se tomaría como:

```
>> time (ls 2> salida.txt)
```

Lo cual tiene un resultado bien distinto.

Medida de la memoria máxima usada por un programa

En teoría, el comando **time** de Linux debería permitir ver información sobre la memoria usada por un proceso, tal y como lo fija el estándar GNU 1.7. Pero en la práctica, la mayoría de las versiones actuales de Linux no lo hacen. Por ello, hemos implementado un sencillo programa para paliar esta carencia, denominado **memory**. Descargar y compilar con:

g++ memory.cpp -o memory

La sintaxis es muy sencilla: **memory** seguido del comando cuya ejecución queremos medir (con los parámetros correspondientes). El programa **memory** evalúa el uso de memoria unas 16 veces por segundo, y se queda con el valor máximo. Los resultados están dados en kilobytes, e igual que **time** se escriben en la salida de error (**stderr**).

Debido a la forma de medir el uso de memoria mediante un muestreo, puede haber cierta imprecisión en los valores devueltos por **memory**. Por ello, puede ser conveniente contrastar sus resultados con los de la medida usando mem_count.