Práctica 3. Perfilado de aplicaciones (I)

NOMBRE DEL ALUMNO:

Creación de una máquina virtual para pruebas (~5 min.)

Arranca Linux y entra como "Usuario VMs". Introduce tu usuario y contraseña.

Abre la carpeta `Disco VMs` desde el escritorio y ve al directorio `ECO`. Abre el fichero `ECO.ova` haciendo doble *click*. Pulsa en "Importar" en la ventana de VirtualBox que aparecerá.

Desde VirtualBox, selecciona la máquina virtual "ECO" y pulsa en "Iniciar" para arrancarla. Entra como usuario "usuario", con contraseña "usuario".

gcc (~20 min.)

Como hemos estudiado en clase, una técnica muy utilizada en evaluación del rendimiento es el "profiling" o monitorización de programas. Mediante esta técnica podemos reconocer la parte de código que consume más tiempo o cuál es la secuencia de llamadas entre procedimientos.
Las etapas principales se describen a continuación:

1- **Instrumentación**: Mediante la instrucción: **`$ gcc proc.c -o prog -pg -g -a`** se añaden líneas de instrumentación al programa de nombre proc.c (nosotros lo haremos en el laboratorio con edges.c). El parámetro –pg indica la recogida de datos relativos a las funciones; el parámetro –g indica la recogida de datos relativos a líneas de código fuente; el parámetro –a indica la recogida de datos relativos a bloques de instrucciones.

2- **Ejecución del programa instrumentado:** $ prog (o $ ./prog): debido a la instrumentación hay que tener en cuenta que el nuevo archivo ejecutable puede ser algo más lento. La ejecución genera uno o varios ficheros con información llamados "data profiles".

3- **Lectura e interpretación de la información generada**: Con **`$ gprof prog>prog.gprof`** se produce la recogida de datos. En el paso 2 se genera un fichero de datos en código binario. Este fichero se llama "gmon.out" pero no es legible por el usuario. Para ello, en esta fase se invoca a un comando gprof que traslada los datos de gmon.out a un formato texto legible por el usuario. Gprof. además, admite parámetros que permiten seleccionar los datos que nos interesa conocer. Por ejemplo, -p visualiza la distribución del tiempo de ejecución entre los procedimientos del programa; -q muestra información sobre el grafo de dependencias entre procedimientos.

4- **Recogida de la información:** La información se almacena en "flat profile" (-p) y en "call graph" (-q)

En este ejercicio se proporciona el programa edges.c que inspecciona el borde de una imagen que recibe como parámetro y almacena los resultados en un archivo de salida.
Con los diferentes niveles de programación, lo que vemos es un programa que no está nada optimizado en un principio pero que con optimización mediante una compilación adecuada se pueden mejorar los tiempos de ejecución.
Compila el programa `edges.c` (disponible en la carpeta del laboratorio en el Campus Virtual) con diferentes niveles de optimización (`-O0`, `-O1`, `-O2` y `-O3`) y obtén los tiempos de ejecución

con cada nivel al procesar la imagen `img.pgm` (también disponible en Campus Virtual). La instancia viene dada por:

```
$ ./edges img.pgm out.pgm
```

Repite el ejercicio anterior activando las optimizaciones guiadas por perfil de ejecución. Para ello:

1. Compila el programa (sin optimización) activando la generación del perfil de ejecución (opción `-fprofile-generate`)
2. Ejecuta el programa con la imagen (se genera el fichero `edges.gcda`)
3. Vuelve a compilar el programa con cada nivel de optimización usando el perfil (opción `-fprofile-use`)

Entrega: Escribe a continuación los comandos introducidos, haz una tabla con los resultados obtenidos en pantalla y un breve análisis de los resultados (excepto la consulta al manual).

gprof (~20 min.)

Consulta la página de manual de `gprof`.

Obtén el tiempo consumido por cada función de `edges.c` compilando sin optimización.

¿Qué función intentarías mejorar primero? ¿Cuánto tardaría en ejecutarse el programa si consiguieras mejorar esa función en un 15%? ¿Cuál sería la máxima mejora que podrías obtener mejorando solamente esa función?

Entrega: Copia los resultados y responde a las preguntas.

También se puede hacer un análisis por líneas de código usando la opción `-l` de `gprof`. Lógicamente, es necesario haber compilado con la opción `-g`.

Si has tardado más tiempo del que se supone en realizar esta parte, salta aquí a la parte 2 de la práctica 3, en otro caso puedes continuar con la parte opcional en la página siguiente.

google-pprof (~30 min.) (Opcional)

Instala el paquete `google-perftools` con:

```
$ sudo apt-get update
$ sudo apt-get install google-perftools libgoogle-perftools-dev
```

Consulta la página de manual de `google-pprof`.

Compila el programa `edges.c` enlazando con la biblioteca `profiler`:

```
$ gcc -o edges edges.c -lprofiler
```

Activa el muestreo durante la ejecución del programa definiendo la variable de entorno `CPUPROFILE` con el fichero de perfil:

```
$ CPUPROFILE=/tmp/edges.prof ./edges img.pgm out.pgm
```

Obtén un informe a partir del perfil de ejecución con el comando `google-pprof`:

```
$ google-pprof --text edges /tmp/edges.prof
```

El significado de cada columna es el siguiente:

1. Número de muestras de esta función
2. Porcentaje de muestras de esta función
3. Porcentaje acumulado de muestras de esta función y las anteriores
4. Número de muestras de esta función y de las funciones a las que llama
5. Porcentaje de muestras de esta función y de las funciones a las que llama
6. Nombre de la función

También se puede obtener un informe gráfico en formato PDF (opción `--pdf`), GIF (opción `--gif`) o SVG (copción `--svg`). Por ejemplo:

```
$ google-pprof --gif edges /tmp/edges.prof > edges.gif
$ xdg-open edges.gif
```

Entrega: Escribe a continuación los comandos introducidos, y copia el fichero `edges.gif`.

Se puede modificar la frecuencia de muestreo (por defecto, 100) definiendo la variable de entorno `CPUPROFILE_FREQUENCY`:

```
$ CPUPROFILE=/tmp/edges.prof CPUPROFILE_FREQUENCY=200 ./edges
img.pgm out.pgm
$ CPUPROFILE=/tmp/edges.prof CPUPROFILE_FREQUENCY=300 ./edges
img.pgm out.pgm
```

Por defecto, se usa el temporizador de tiempo de CPU `ITIMER_PROF` (ver `man getitimer`) para muestrear. El temporizador de tiempo real `ITIMER_REAL` es más apropiado para programas que realicen E/S y permite mayor frecuencia de muestreo. Para usarlo, se define la variable `CPUPROFILE_REALTIME`:

```
$ CPUPROFILE=/tmp/edges.prof CPUPROFILE_FREQUENCY=300
CPUPROFILE_REALTIME=1 ./edges img.pgm out.pgm
$ CPUPROFILE=/tmp/edges.prof CPUPROFILE_FREQUENCY=1000
CPUPROFILE_REALTIME=1 ./edges img.pgm out.pgm
```

Obtén el tiempo de ejecución sin muestreo, muestreando con la frecuencia y el temporizador por defecto y muestreando 1000 veces por segundo con el temporizador de tiempo real. ¿Qué sobrecarga se produce en cada caso?

Entrega: Copia los resultados de este último ejercicio y responde a las preguntas.

Se puede obtener un informe por líneas de código en lugar de por funciones, usando la opción `--lines` (es necesario haber compilado con la opción `-g`):

```
$ google-pprof --text --lines edges /tmp/edges.prof
```