66:20 Organización de computadoras
Trabajo práctico 0: Infraestructura básica.

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 9), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo[1], y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2].

En la clase del 30/8 hemos repasado, brevemente, los pasos necesarios para la instalación y configuración del entorno de desarrollo

También se utilizará `gprof`[7], una herramienta de profiling, para evaluar el desempeño de programas escritos en C.

[1] `http://groups.yahoo.com/group/orga6620`

5. Descripción.

El siguiente trabajo práctico consiste en implementar una versión minimalista del comando *sort*[5], utilizando dos algoritmos de ordenamiento: *Quicksort*[3] y *Stooge sort*[4].
Una vez implementados, procederemos a realizar mediciones para evaluar las posibildades de mejora y el desempeño relativo entre ambas implementaciones, utilizando los programas `time`[6] y `gprof`[7].

6. Implementación.

El programa debe leer los datos de entrada desde *stdin* o bien desde uno o mas archivos. La salida del programa debe imprimirse por *stdout*, mientras que los errores deben imprimirse por *stderr*. El algoritmo de busqueda puede seleccionarse mediante las opciones `-q` o `-s` para *quicksort* o *stooge sort* respectivamente.
Mostramos el mensaje de ayuda mediante la opción `-h`.

```
$tp0 -h
tp0 [OPTIONS][file...]
-h, --help              display this help and exit.
-V, --version           display version information and exit.
-q, --quick             use the quicksort algorithm.
-s, --stooqe            use the stoogesort algorithm.

$cat 1984.p1.txt
It was a bright cold day in April, and the clocks were striking thirteen.
Winston Smith, his chin nuzzled into his breast in an effort to escape the
vile wind, slipped quickly through the glass doors of Victory Mansions,
though not quickly enough to prevent a swirl of gritty dust from entering
along with him.
$tp0 -s 1984.p1.txt
along with him.
It was a bright cold day in April, and the clocks were striking thirteen.
though not quickly enough to prevent a swirl of gritty dust from entering
vile wind, slipped quickly through the glass doors of Victory Mansions,
Winston Smith, his chin nuzzled into his breast in an effort to escape the
$
```

6.1. Portabilidad

Como es usual, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad. Para satisfacer esto, el programa deberá funcionar al menos en NetBSD/pmax (usando el simulador GXemul [1]) y la versión de Linux (Knoppix, RedHat, Debian, Ubuntu) usada para correr el simulador, Linux/i386.

7. Mediciones.

Utilizar `time` para tomar el tiempo que tardan ambos algoritmos en ordenar una entrada de tamaños 1KB, 8KB, 16KB para:

- Un archivo ya ordenado
- Un archivo en orden inverso

Graficar para cada algoritmo el tiempo insumido contra el tamaño de muestra, para los dos casos. Graficar el speedup de *Quicksort* contra *Stoogesort* para los diversos valores de N, para los tres casos.

8. Profiling

Supongamos que queremos optimizar el *Selection sort* valiéndonos de la herramienta `gprof`.

- ¿Qué tamaño de muestra nos conviene más tomar para hacer el profiling?
- Si tuviera que elegir una y sólo una función para optimizar (asumiendo que cualquiera puede ser arbitrariamente optimizada), ¿Cuál elegiría? ¿Cuál sería el speedup máximo obtenible mediante la optimización de esa función?

9. Informe.

El informe debe incluir:

- Informe describiendo el desarrollo del trabajo práctico.
- Comando(s) para compilar el programa.
- Corridas de prueba, con los comentarios pertinentes.
- CD conteniendo todo el material digital.
- Código assembly MIPS32 generado por el compilador **(solo en formato digital)**
- Código fuente.
- Este enunciado.

10. Fechas de entrega.

- Primera entrega: 13-9
- Revisión: 20-9
- Vencimiento: 27-9

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project, `http://www.netbsd.org/`.

[3] Quicksort `http://en.wikipedia.org/wiki/Quick_sort`

[4] Stoogesort `http://en.wikipedia.org/wiki/Stooge_sort`

[5] sort http://en.wikipedia.org/wiki/Sort_(Unix)

[6] time man page http://unixhelp.ed.ac.uk/CGI/man-cgi?time

[7] GNU gprof http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html