Trabajo Práctico 0

Donikian Santiago , *Padrón Nro. 85689*
santiagodonikian@hotmail.com
Dubini Richard, *Padrón Nro. 85440*
r_dubini@hotmail.com

2do. Cuatrimestre de 2009
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

20 de septiembre de 2009

1. Objetivos

El objetivo de este trabajo práctico es que nos familiaricemos con las herramientas que se usarán a lo largo de la cursada:

- Sistema operativo Linux
- Máquina virtual GxEmul
- Lenguaje de programación C
- Lenguaje de marcado LaTeX

2. Introducción

El trabajo consiste en instalar y ejecutar, correctamente, la máquina virtual GxEmul que se usará para emular la arquitectura MIPS corriendo el sistema operativo NetBSD. Una vez hecho esto, habrá que implementar una aplicación en lenguaje C similar a cut de Unix (extrae bytes o grupos de bytes de cada linea de un archivo, y los escribe en stdout) y compilarla en el host (con sistema operativo linux) y en NetBSD. A lo largo del informe se describirán los pasos seguidos para desarrollar el trabajo práctico.

3. Desarrollo

A continuación veremos los pasos que se siguieron para hacer el trabajo práctico.

3.1. Paso 1: Instalación de Linux

Este paso dependió de cada miembro del grupo, dado que alguno de ellos ya tenían instalado el sistema operativo necesario para trabajar. En general trabajamos con la distribución Ubuntu 8.10 (basada en Debian) que se puede bajar libremente de http://www.ubuntu.com/ . La instalación no genera mayores inconvenientes ya que esta distribución está pensada para usuarios finales.

3.2. Paso 2: Descarga e instalación de GxEmul

La descarga se realizó desde el sitio indicado en la cursada, dicha versión de GxEmul ya viene configurada con el NetBSD. El archivo descargado, es un bzip, es decir, un archivo comprimido. La sentencia UNIX para descomprimirlo es:

bzip2 -dc [nombre de archivo] — cpio –sparse -i -v

Nota: bzip2 es un programa libre desarrollado bajo licencia BSD que comprime y descomprime ficheros usando los algoritmos de compresión de Burrows-Wheeler y de codificación de Huffman. Para ejecutar GxEmul se utiliza la sentencia:

./gxemul -e 3max -d netbsd-pmax.img

3.3. Paso 3: Tunel Ssh

A continuación listaremos los comandos más importantes:

- ifconfig lo:0 172.20.0.1 configuramos una ip para que NetBSD se conecte con el Host (Ubuntu).
- ssh -R 2222:127.0.0.1:22 USER@172.20.0.1 este comando se ejecuta desde la consola de NetBSD para conectarse al Ubuntu
- ssh -p 2222 root@127.0.0.1 este comando se ejecuta desde la consola de Ubuntu para conectarse al NetBSD
- scp -p 2222 ARCHIVO USER@IP:/DIRDONDEQUIEROCOPIAR este comando sirve para copiar archivos desde el Host hasta el NetBSD (y viceversa).

3.4. Paso 4: Implementación del programa

3.4.1. Ingreso de parámetros

Los parámetros validos que puede recibir la función son:
-V, –version Print version and quit.
-h, –help Print this information and quit.
-d, –delimitert Use the first character of the specified string as field delimiter instead of TAB.
-f, –field LIST specifies field positions to be extracted.
-b, –bytes LIST specifies byte positions to be extracted.
-s, –ignore Ignore lines not containing delimiters.

Para parsear los parámetros se uso la función:

int getopt_long (int argc, char *const *argv, const char *shortopts, const struct option *longopts, int *indexptr)

Esta función permite recoger los parámetros de entrada del programa tanto en su formato corto como en su formato largo. Ejemplo: -h –help. Existen otras funciones como argp que son propias de Linux. Se eligió esta por que es compatible con NetBSD (también es compatible con Windows).

Los parámetros de entrada se interpretan así:

./tp0 [opción] [argumentos de la opción][archivos de entrada]

- Todos los parámetros que no empiecen con «-» o «–» y no sean argumento de alguna opción (la opcion -s no tiene argumentos) serán considerados como archivos de entrada. El programa limita a solamente 10 archivos de entrada. Si hay más de 10 archivos de entrada no se tendrán en cuenta. En el caso de que no se especifique archivo de entrada, se lee de stdin.
- Siempre se usará la salidad standard para mostrar los resultados.
- Las opticiones habilitarán banderas que luego le avisarán al programa que hacer.

3.4.2. Funciones para contar

Dentro de la aplicación, dividos el conjunto de las funciones para contar en dos:

- Las funciones para contar archivos.
- Las funciones para contar cadenas de texto.

El criterio para las funciones para contar archivos fue el siguiente: Se considera que más probable que el usuario pida una unica opción que las tres opciones juntas (cantidad de líneas, bytes, palabras). Por ende, se decidió el archivo se lea dentro de cada función.

```
int fcountWord(FILE* file);
int fcountBytes(FILE* file);
int countLines(FILE* file);
```

```
int countWord(char* cadena);
int countBytes(char* cadena);
```

Para contar la cantidad de bytes se consideró que un caracter ocupa en c 1 byte por ende una cadena de caracteres ocupará 1byte x la cantidad de caraceteres. El operador \n y el espacio se consideran como un caracter. Para contar la cantidad de líneas se consideró que una línea termina en \n . Para contar la cantidad de palabras se divide la cadena por sus espacios y se cuenta la cantidad de palabras que quedan.

3.4.3. Salida de datos

Como ya se mencionó en la sección de parámetros, la salida de datos considera que si al programa no se le pasa la opción -o con un nombre de archivo, el programa usará la salidad standard y mostrará los datos por pantalla. El formato de los datos es el siguiente:

[tab]lineas[tab]palabras[tab]bytes[tab]nombre archivo

Total: líneas palabras bytes

3.5. Paso 5: Informe con LaTex

Lo primero que hicimos fue buscar las herramientas que necesitamos.
Para Windows:

- MikTex, es una distribución de LaTex para windows. Se puede conseguir gratuitamente en http://miktex.org/
- TeXnicCenter, es un ambiente para crear documentos LaTex. Se puede bajar gratuitamente de http://www.texniccenter.org/

Para Linux: Hay varios ambientes de trabajo por ejemplo el Kile.

3.5.1. Estructura básica de un documento de LaTex

Un documento básico de latex tiene la siguiente estructura:
Clase de documento: \[a4paper,11pt,oneside]{article}
Claramente podemos ver lo que se está definiendo. En particular el parametro report indica el tipo de documento: articulo, libro, reporte, etc...

Paquetes
\usepackage[latin1]{inputenc}
\usepackage[activeacute,spanish]{babel}
Con estos paquetes le decimos que tipo de teclado usamos (latin1) y habilitamos los acentos españoles.

Título y Autor
\title{título}
\author{autor}
\date{\today}
Aquí se configura el título, autor y fecha.

Documento
\begin{document}
\end{document}
Dentro del begin y el end va el contenido del documento.

4. Corridas de prueba

La figura 1 muestra la línea utilizada para compilar el código en Linux, y además se puede apreciar las diferentes pruebas que están como ejemplo en el enunciado.

```
richy@richy-laptop:~/orga6620/tp0$ gcc -Wall -O0 tp0.c -o tp0.o
richy@richy-laptop:~/orga6620/tp0$ ./tp0.o -h
The wc utility displays the number of lines, words,
bytes and characters contained in each input file
(or standard input, by default) to the stan-dard output.

Usage:
 tp0 -h
 tp0 -V tp0 [options]
-V, --version Print version and quit.
-h, --help Print this information and quit.
-l, --lines Print the number of lines.
-b, --bytes Print the number of bytes.
-w, --words Print the number of words.
Example:
 tp0 in.txt in2.txt -o out.txt -l
richy@richy-laptop:~/orga6620/tp0$ echo a | ./tp0.o -b
         2
 richy@richy-laptop:~/orga6620/tp0$ echo abc | ./tp0.o
         1       1       4
richy@richy-laptop:~/orga6620/tp0$ cat arch1.in
a b c
richy@richy-laptop:~/orga6620/tp0$ cat arch3.in
HOLA

MUNDO
richy@richy-laptop:~/orga6620/tp0$ cat arch3.in | ./tp0.o -w
         2
 richy@richy-laptop:~/orga6620/tp0$ ./tp0.o -l arch1.in
         1 arch1.in
richy@richy-laptop:~/orga6620/tp0$ ./tp0.o arch1.in
         1       3       6       arch1.in
richy@richy-laptop:~/orga6620/tp0$ ./tp0.o arch1.in arch3.in
         1       3       6       arch1.in
         3       2       12      arch3.in
         4       5       18 total
richy@richy-laptop:~/orga6620/tp0$ ./tp0.o -b arch1.in arch3.in
         6 arch1.in
         12 arch3.in
         18 total
```

Figura 1: Corridas en Linux

```
root@:/# gcc -Wall -O0 tp0.c -o tp0.o
root@:/# ./tp0.o -h
The wc utility displays the number of lines, words,
bytes and characters contained in each input file
(or standard input, by default) to the stan-dard output.

Usage:
 tp0 -h
 tp0 -V tp0 [options]
-V, --version Print version and quit.
-h, --help Print this information and quit.
-l, --lines Print the number of lines.
-b, --bytes Print the number of bytes.
-w, --words Print the number of words.
Example:
 tp0 in.txt in2.txt -o out.txt -l
root@:/# echo a | ./tp0.o -b
         2
 root@:/# echo abc | ./tp0.o
         1       1       4
root@:/# cat arch1.in
a b c
root@:/# cat arch3.in
HOLA

MUNDO
root@:/# cat arch3.in | ./tp0.o -w
         2
 root@:/# ./tp0.o -l arch1.in
         1 arch1.in
root@:/# ./tp0.o arch1.in
         1       3       6       arch1.in
root@:/# ./tp0.o arch1.in arch3.in
         1       3       6       arch1.in
         3       2       12      arch3.in
         4       5       18 total
root@:/# ./tp0.o -b arch1.in arch3.in
         6 arch1.in
         12 arch3.in
         18 total
```

Figura 2: Corridas en NetBSD

La figura 2 muestra los mismos ejemplos y el comando para compilar en NetBSD.

5. Conclusiones

Si bien las activades del trabajo son sencillas, a lo largo del desarrollo del mismo tuvimos inconvenientes para establecer la conexion SSH desde linux, afortunadamente el error pudo solucionarse facilmente, borrando el archivo known_hosts en el sistema operativo NetBSD. Finalmente podemos decir que el trabajo practico nos sirvió para familiarizarnos con las herramientas utilizadas, si bien todos los integrantes del grupo habían programado en el lenguaje C en otras ocasiones, no todos habian utilizado el sistema operativo

Linux, el lenguaje de marcado LATEX, ni la maquina virtual GxEmul. De las corridas de prueba podemos apreciar que el resultado obtenido era el esperado.

Referencias

[1] http://www.ubuntu.com/

[2] http://www.cs.duke.edu/courses/spring04/cps108/resources/getoptman.html

[3] http://los-pajaros-de-hogano.blogspot.com/2009/03/hal-y-las-marcas-latex-i.html