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

22 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

Se dividio el conjunto de funciones principales de la siguiente manera:

- funciones de validacion.

  getOptions(int argc, char** argv,char** ofname) valida parametros y setea sus repectivos valores. argc=numero de parametros, argv=lista de parametros
  validateCommand() valida que las combinaciones de flags sean validas
  validateRange(char* datos) valida y obtiene los rangos para los cuales se toman los bytes y los campos

- funciones de parseo.

  listFields(char* a) obtiene los campos seteados en el vector 'rango'de la cadena á', y los imprime por consola
  listBytes(char* a) obtiene los bytes seteados en el vector 'rango'de la cadena á', y los imprime por consola

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-desktop:~/workspace/TpsOrga$ gcc -Wall -O0 -o tp0.o TP0.c
richy@richy-desktop:~/workspace/TpsOrga$ ./tp0.o -help

The tp0 utility is used to extract sections from each line of input (usually fro
m a file)
Usage:  tp0 [-h] [-V] [-f LIST] [-d CHARACTER] [-b LIST] [-s] [files]

-V, --version           Print version and quit.
-h, --help              Print this information and quit.
-d, --delimiter         Use as field delimiter instead of tab character.
-b, --bytes             Specifies bytes positions to be extracted.
       LIST [n,]        Bytes position to be extracted, separated by points
       LIST [n-n]       Extract bytes in this range
       LIST [n-]        Extract bytes from this position to the end
       LIST [-n]        Extract bytes from the beginning to this position
-f, --field             Specifies fields positions to be extracted.
       LIST [n,]        Fields to be extracted, separated by points
       LIST [n-n]       Extract fields in this range
       LIST [n-]        Extract from this field to the end
       LIST [-n]        Extract from the beginning to this field
-s, --ignore            Ignore lines not containing delimiters
files                   Files to be cutted

Examples:
        tp0 -b -3 inputl.in
richy@richy-desktop:~/workspace/TpsOrga$ echo abc.def.ghi | ./tp0.o -f 1,3 -d .
abc.ghi
richy@richy-desktop:~/workspace/TpsOrga$ echo "123456789">/tmp/input.in
richy@richy-desktop:~/workspace/TpsOrga$ ./tp0.o -b 2 /tmp/input.in
2
richy@richy-desktop:~/workspace/TpsOrga$ cat input1.in
12345
67890
richy@richy-desktop:~/workspace/TpsOrga$ cat input2.in
abcdef
ghijk
richy@richy-desktop:~/workspace/TpsOrga$ ./tp0.o -b 1 input1.in input2.in
1
6
a
g
richy@richy-desktop:~/workspace/TpsOrga$ ./tp0.o -b 2- input1.in
2345
7890
richy@richy-desktop:~/workspace/TpsOrga$ ./tp0.o -f 1 -d . -s input1.in
richy@richy-desktop:~/workspace/TpsOrga$
```

Figura 1: Corridas en Linux

```
root@:/# gcc -Wall -O0 -o tp0.o TP0.c
root@:/# ./tp0.o -help

The tp0 utility is used to extract sections from each line of input (usually from
 a file)
Usage:  tp0 [-h] [-V] [-f LIST] [-d CHARACTER] [-b LIST] [-s] [files]

-V, --version           Print version and quit.
-h, --help              Print this information and quit.
-d, --delimiter         Use as field delimiter instead of tab character.
-b, --bytes             Specifies bytes positions to be extracted.
        LIST [n,]       Bytes position to be extracted, separated by points
        LIST [n-n]      Extract bytes in this range
        LIST [n-]       Extract bytes from this position to the end
        LIST [-n]       Extract bytes from the beginning to this position
-f, --field             Specifies fields positions to be extracted.
        LIST [n,]       Fields to be extracted, separated by points
        LIST [n-n]      Extract fields in this range
        LIST [n-]       Extract from this field to the end
        LIST [-n]       Extract from the beginning to this field
-s, --ignore            Ignore lines not containing delimiters
files                   Files to be cutted

Examples:
         tp0 -b -3 inputl.in
         tp0 -f 1-5 -d . tes.txtroot@:/#
root@:/# echo abc.def.ghi | ./tp0.o -f 1,3 -d .
abc.ghi
root@:/# echo "123456789">/tmp/input.in
root@:/# ./tp0.o -b 2 /tmp/input.in
2
root@:/# cat input1.in
12345
67890
root@:/# cat input2.in
abcdef
ghijk
root@:/# ./tp0.o -b 1 input1.in input2.in
1
6
a
g
root@:/# ./tp0.o -b 2- input1.in
2345
7890
root@:/# ./tp0.o -f 1 -d . -s input1.in
root@:/#
```

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