Introducción a Linux

Introducción a los Algoritmos, FaMAF, UNC

1er. cuatrimestre 2012

En esta materia los alumnos tendrán la oportunidad de utilizar las computadoras disponibles en los laboratorios para la realización de ejercicios prácticos, como la verificación de pruebas o la implementación de programas. Para ello, utilizarán el sistema operativo Linux y el lenguaje de programación Haskell. Este documento intenta cubrir los conceptos básicos necesarios para desarrollar las acitvidades de taller utilizando principalmente la interfaz en modo texto.

1 Algunas definiciones

- **Sistema operativo**: es el conjunto de programas que administran los recursos de la computadora (memoria, disco, etc) y que ayuda en el desarrollo y ejecución de los programas (o software) de más alto nivel, es decir, programas desarrollados por los usuarios, aplicaciones de oficina, aplicaciones de internet, etc.
- **Software libre**[1]: El software libre es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Más precisamente, significa que los usuarios de programas tienen las cuatro libertades esenciales.
  - La libertad de ejecutar el programa, para cualquier propósito.
  - La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.
  - La libertad de redistribuir copias para que pueda ayudar al prójimo.
  - La libertad de distribuir copias de sus versiones modificadas a terceros. Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.
- **Linux** es un sistema operativo, con varias características que lo diferencian del resto de sistemas que podemos encontrar en el mercado: entre otras, podemos destacar que es software libre y que está formado por un núcleo (en Inglés *kernel*) más un gran número de programas o bibliotecas que hacen posible su utilización.

  Linux se distribuye bajo la licencia *GNU General Public License* y, por lo tanto, el código fuente tiene que estar siempre accesible y cualquier modificación o trabajo derivado tiene que tener esta licencia.
- **Lenguaje de programación**: es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, y es utilizado para controlar el comportamiento físico y lógico de una computadora; también permite especificar de manera precisa sobre qué datos se debe operar, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo ciertas condiciones; algunos ejemplos de lenguajes de programación son Haskell, C, Java, Python.
- **Compilador**: es un programa que traduce el código fuente generado por lenguajes de programación a un código de máquina para alguna arquitectura particular. El código generado por un compilador puede ejecutarse repetidas veces sin necesidad de volver a compilar; si es necesario hacer cambios en el código fuente el programa debe recompilarse para crear un nuevo ejecutable que incluya los cambios.

- **Intérprete**: es un programa que "lee" el código fuente y lo ejecuta pero a diferencia de un compilar, el intérprete no genera nignún código ejecutable.En el caso particular de Haskell, el lenguaje de programación que utilizarán en esta materia, existe tanto un compilador com un intérprete de Haskell.

2 Interfaz gráfica

Como la mayoría de los sistemas operativos, Linux provee un entorno gráfico (en Inglés *Grahical User Interface* o *GUI*), en el cual se puede acceder a las distintas aplicaciones a través de ventanas, interactuando con el sistema principalmente por medio clicks de mouse. Una característica particular del entorno gráfico es que tiene distintos espacios de trabajo (en Inglés *workspaces*) que permiten agrupar distintas ventanas incrementando el tamaño del escritorio ya que por defecto existen 4 espacios de trabajo distintos. La idea de tener distintos escritorios es poder organizarse mejor, agrupando las ventanas relacionadas en distintos escritorios, por ejemplo, en uno se ponen las ventanas relacionadas al trabajo (terminales, entonros de programación, etc) y en otro las ventanas de pasatiempos (juegos, música, etc)

Dado que el objetivo de este apunte es reforzar el uso de la interfaz en modo texto, no daremos más detalles de cómo utilizar esta interfaz; para ello, consultar [2,3].

Figure 1: Captura de pantalla de un entorno gráfico de Linux (distribución Ubuntu).

3 Interfaz modo texto

El otro tipo de interfaz que provee la mayoría de los sistemas operativos, es la interfaz en modo texto donde a diferencia de las anteriores, la interacción se da a través de comandos específicos ingresados por teclado. Ejemplos de estas interfaces son *Microsoft DOS* y la *línea de comandos* (o *command line*) de Linux. Una forma de acceder a la línea de comandos es abriendo en el entorno gráfico una *consola* y comenzar a tipear comando ahí, y otra manera es "logueandose" en una terminal fuera del entorno gráfico. Estas opciones se ilustran en las Figuras 2 y 3 , respectivamente.

Estos modos de acceso no son excluyentes, es decir que mientras estamos en el entorno gráfico podemos loguearnos en una o más terminales virtuales y una combinación de teclas nos permiten navegar por las ditintas terminales. Normalmente, Linux tiene 6 terminales virtuales y la 7ma. es en la que se corre la interfaz gráfica.

Desde el entorno gráfico, presionando Ctrl+Alt+F1 vamos a la primer terminal, con Ctrl+Alt+F2 vamos a la segunda y así sucesivamente y desde una terminal fuera del entorno gráfico podemos movernos por las distintas terminales con la combinación Alt+F1 vamos a la terinal 1, Alt+F2 terminal 2, etc y con alt+F7 volvemos al entorno gráfico.

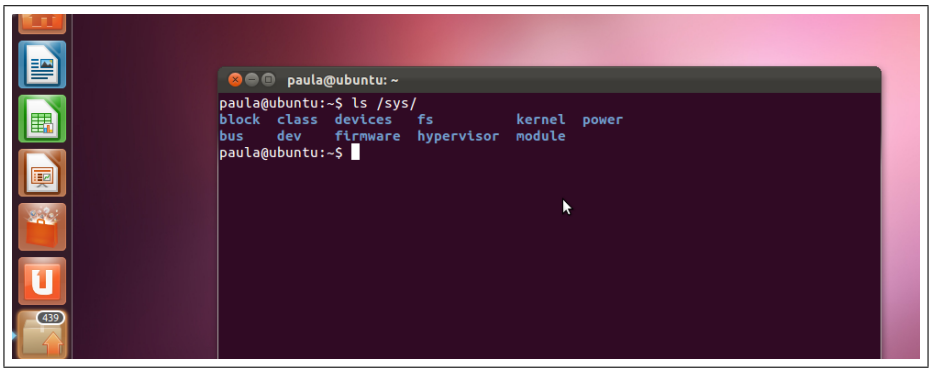

Figure 2: Acceso a la línea de comandos usando una consola en el entorno gráfico de Linux.

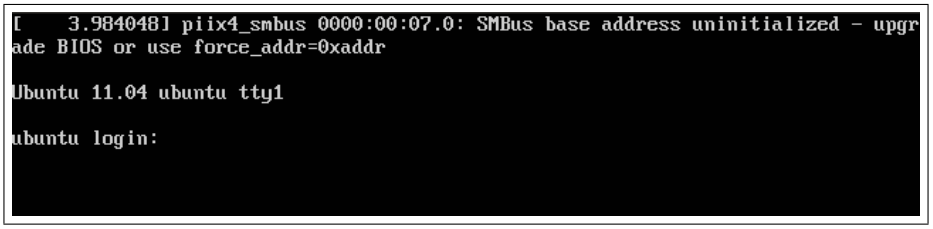

Figure 3: Acceso a la línea de comandos en una terminal (modo texto).

3.1 El shell

En el modo texto existe un programa encargado de recibir los comandos que tipea el usuario, analizarlos, pasarlos al sistema operativo para que ejecute la acción indicada y mostrar el resultado. A este programa se le conoce como *shell* y la Figure 2 muestra una consola donde el comando tipeado por el usuario (*ls*) pide mostrar los archivos del directorio */sys/* y resultado de tal acción es mostrardo por pantalla. Existen distintos shells pero los más comunes son: *sh* (llamada "Bourne shell"), *bash* ("Bourne again shell"), *csh* ("C Shell"), *Tcsh* ("Tenex C shell"), *ksh* ("Korn shell") y *zsh* ("Zero shell").

3.2 Indicador del sistema o *prompt*

Cuando el shell se inicia, aparece el indicador de sistema (o *prompt* en Inglés) que es una línea del estilo `equipo:/directorio/actual$`. El último carácter indica el tipo de usuario conectado: `$` especifica un usuario normal y `#` especifica el administrador, llamado *root*.

3.3 Redireccionamiento

Linux posee mecanismos que permiten redirigir la entrada-salida estándar a archivos usando el caracter `>`, es decir, redirigir el resultado de un comando que se encuentra a la izquierda a un archivo que se encuentra a la derecha del comando; por ejemplo:

`ls -al /home/ > homes.txt` → averiguar qué hace este comando
`echo "Hola" > miarchivo` → averiguar qué hace este comando
`cat miarchivo > toto2` → averiguar qué hace este comando

El propósito de la redireccin es el de crear archivos nuevos mientras que uso del carácter `>>` permite agregar la salida estándar a un archivo.

De manera similar, el caracter `<` indica una redirección de la entrada estandar, por ejemplo, el siguiente comando envía el contenido del archivo `toto.txt` con el comando `cat`, cuyo único propósito es mostrar el contenido en la salida estandar:

```
cat < toto.txt
```

Por último, el uso de la redirección `<<` permite la lectura por entrada estandar, hasta que se encuentre la cadena ubicada a la derecha. En el siguiente ejemplo, se lee la entrada estandar hasta que se encuentra la palabra `STOP`, luego se muestra el resultado:

```
cat << STOP
```

3.4 Tuberías de comunicación o *pipelines*

Las tuberías (en Inglés *pipes*) son mecanismos de comunicación específicos para Linux y se denota con el símbolo `|`. Los pipes permiten asignar la salida estandar de un comando a la entrada estandar de otro, de la misma forma en que una tubería permite la comunicación entre la entrada estandar de un comando y la salida estandar de otro. Por ejemplo, en el siguiente comando, la salida estandar del comando `ls -la` (que muestra los archivos de un directorio) se envía al programa `sort`, el cual debe extraer el resultado en orden alfabético.

Ejercicio: ejecutar `ls -al | sort` en una consola y ver qué resultado da

También es posible conectar una cierta cantidad de comandos usando varios pipes, por ejemplo, el siguiente comando muestra todos los archivos del directorio actual, luego selecciona las líneas que contienen la palabra `zip` utilizando el comando `grep` y finalmente cuenta la cantidad total de líneas extraídas:

Ejercicio: ejecutar `ls -l | grep zip | wc -l` en una consola y ver qué resultado da

4 Forma de trabajo en los prácticos

Dado que en esta materia pretendemos fomentar el uso de la línea de comandos para que los alumnos se familiaricen con las prácticas que serán comunes en el futuro, a continuación describimos cómo trabajar en las máquinas del laboratorio para realizar los ejercicios prácticos utilizando Haskell fuera del entorno gráfico.

1. Luego de prender la pc, ésta muestra la pantalla de ingreso al entorno gráfico, por lo tanto, presionar Ctrl+Alt+F1 para ir a una terminal virtual

2. Loguearse con su usuario y clave; para quienes no tengan usuario/clave utilizar *visita* como usuario y como clave. Notar que el usuario *visita* no permite almacenar ningún archivo (al salir de la terminal todos los archivos guardados se borrarán automáticamente) por lo que se recomienda guardarlos en un dispositivo usb o mandárselos por mail (a uno mismo o un compañero)

3. Tipear `ghci` para abrir el intérprete de Haskell; luego de cargarse se muestra un prompt del estilo `Prelude>`

4. Loguearse en otra terminal (presionando Alt+F2) y abrir ahí un editor de texto para crear el archivo de programas que se cargará en Haskell

5. Supongamos que hemos resuelto algunos ejercicios, guardamos el archivo como `practico-introalg.hs` (notar que la extensión debe ser .hs)

6. Para cargarlo en Haskell me cambio a la terminal 1 (con Alt+F1) y tipeo `:l practico-introalg`; en este momento ya empezamos a trabajar con Haskell, probando las funciones programadas o arreglando algún error que muestre el intérprete.

7. Si debo/deseo modificar el archivo, me cambio a la terminal 2, hago las modificaciones, guardo y vuelvo a la terminal 1

8. Para recargar el archivo y así incorporar los cambios realizados, tipear `:r` en Haskell

9. Para dejar de trabajar, se debe salir de Haskell tipeando `:q` y terminar las sesiones tipeado `logout` en cada una de las terminales que estemos utilizando

5 Lista de comandos de Linux

Comando	Acción
`cd [directorio]` (por ej. /home)	cambiar al directorio entre []
`cd ..`	regresar un nivel en el árbol de directorios; se pueden combinar, como `cd ../temp/`
`cd`	cambiar al directorio `home`
`pwd`	mostrar la ruta del directorio de trabajo
`ls`	ver archivos del directorio actual
`ls -l`	mostrar detalles de archivos y directorios
`ls -a`	mostrar archivos ocultos, se pueden combinar, por ejemplo `ls -la /home/paula/`
`mkdir dir1`	crear un directorio llamado 'dir1'
`rm -f file1`	borrar el archivo con nombre 'file1'
`rmdir dir1`	borrar directorio con nombre 'dir1'
`rm -rf dir1`	borrar el directorio con nombre 'dir1' y todos sus contenidos recursivamente
`mv dir1 new-dir`	renombrar o mover un archivo o directorio
`cp file1 file2`	copiar un archivo
`cp dir/* .`	copiar todos los archivos de un directorio dentro del directorio de trabajo actual
`find / -name file1`	buscar archivos y directorios con el nombre 'file1' desde '/'
`find /home/user1 -name .bin`	buscar archivos con extensión '.bin' dentro del directorio '/ home/user1'
`locate .ps`	mostrar archivos con la extensión '.ps'
`whereis [comando]`	mostrar la ruta del archivo binario, fuente y página del manual (man) para un comando dado
`which halt`	mostrar la ruta completa a un binario / ejecutable
`gunzip file1.gz`	descomprimir un archivo llamado 'file1.gz'
`gzip file1`	comprimir un archivo llamado 'file1'
`rar a file1.rar test-file`	crear un archivo rar llamado 'file1.rar'
`rar a file1.rar file1 file2 dir1`	comprimir 'file1', 'file2' y 'dir1' simultaneamente
`rar x file1.rar`	descomprimir un archivo rar
`unrar x file1.rar`	descomprimir un archivo rar
`tar -cvf archive.tar file1`	crear un tarball (archivo tar) sin compresión
`tar -cvf archive.tar file1 file2 dir1`	crear un archivo tar que contiene a los archivos 'file1', 'file2' y 'dir1'

`tar -tf archive.tar`	mostrar los contenidos de un archivo tar
`tar -xvf archive.tar`	extraer un archivo tar
`zip file1.zip file1`	crear un archivo tar comprimido en zip
`zip -r file1.zip file1 file2 dir1`	comprimir en formato zip varios archivos y directorios simultaneamente
`unzip file1.zip`	descomprimir un archivo zip
`cat file1`	ver el contenido de un archivo empezando por el primer renglón.
`more file1`	ver contenidos de un archivo una pantalla a la vez
`less file1`	similar al comando 'more' pero permite movimiento tanto hacia atras como hacia adelante
`head -2 file1`	ver las dos primeras líneas de un archivo
`tail -2 file1`	ver las ultimas dos líneas de un archivo
`tail -f /var/log/messages`	ver en tiempo real lo que se va agregando al archivo

Table 1: Comandos más útiles de Linux

6 Referencias

1. http://www.gnu.org/philosophy/free-sw.es.html
2. http://www.ice.udl.es/udv/manuals/linux.pdf
3. https://help.ubuntu.com/community/ServerGUI
4. http://oreilly.com/linux/command-directory/
5. http://www.esdebian.org/wiki/lista-comandos-gnulinux-i