Introducción a GNU/Linux v0.2

Pablo Galaviz

13 de abril de 2005

Índice general

Capítulo 1

Introducción

El propósito de estas notas es dar una referencia de los comandos más utilizados, homogeneizar un poco los conocimientos del grupo y proporcionar referencias y documentación[1] para una posterior consulta. Así mismo se dará una revisión de los comandos utilizados para ejecutar programas en paralelo, monitorear los recursos del sistema y en general para utilizar el cluster del laboratorio.

1.1. El sistema operativo *Linux*

Existen dos tipos de sistemas operativos: Los monolíticos (como Windows) en los que el sistema operativo es un gran programa que se encarga de todo y los sistemas tipo UNIX, que son pequeños programas que en conjunto realizan todas las funciones de interacción entre el usuario, los procesos y el hardware. Un sistema operativo tipo UNIX se representa en forma de capas (ver figura 1.1) que interactúan entre si, la capa más interna representa el **hardware** de la computadora. El hardware se comunica con el **núcleo** del sistema operativo; el núcleo es un programa que se encarga de administrar los procesos, los sistemas de archivos y en general de controlar el hardware del sistema por medio de *controladores de dispositivo*. El núcleo tiene un **entorno de sistema**, que se encarga de la interacción entre el usuario, los programas (compiladores, editores, etc.), daemons y el núcleo. El entorno del

[1] Se anexa un disco con documentos en formato PDF de manuales y libros bajados de la red.

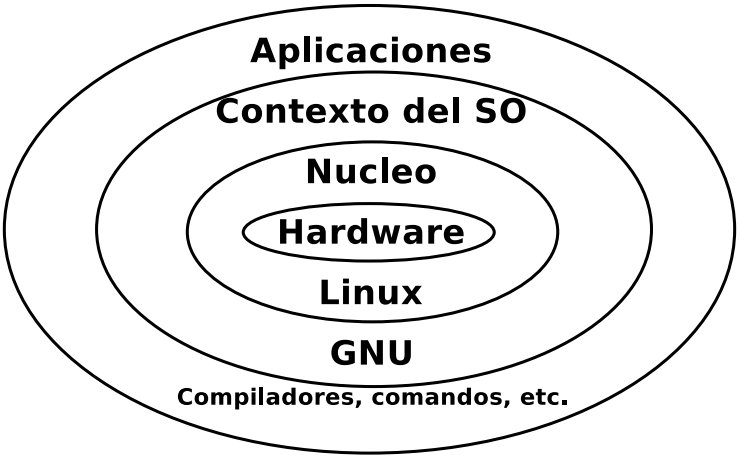

Figura 1.1: Capas de un sistema operativo tipo UNIX.

sistema consta de muchos programas[2] como por ejemplo interpretes de linea (shell), porgramas de administración (como fdisk), comandos de linea (como ls y cd), etc.

Linux es solo un núcleo de sistema operativo (clon de UNIX) y representa solo el 3 % del sistema operativo basado en él. El proyecto GNU (Gnu's Not Unix) está dedicado a desarrollar un sistema operativo libre -de código abierto- seguro y funcional. Las distribuciones conocidas como linux, son en realidad distribuciones basadas en el sistema GNU con el núcleo Linux, la denominación correcta para estos sistemas es GNU/Linux.

1.2. Breve historia de GNU/Linux

El abuelo de los sistemas tipo UNIX se llamó **MULTIX** (MULTiplexed Information Computing System, Sistema de computación de información multiplexada) fue un proyecto de investigación llevado a cabo por GE, AT&T Bell Laboratories y el MIT a finales de los años 60's. En 1969, Ken Thompson, Dennis Ritchie y los investigadores de AT&T Bell Laboratories desarrollaron UNIX como una modificación de MULTIX para minicomputadoras[3]. En 1973

[2]En un sistema típico son alrededor de 1,000 paquetes

[3]Antiguamente las computadoras se clasifican por su capacidad como: supercomputadoras, mainframe, minicomputadoras y microcomputadoras (PC), en la actualidad la jerarquía es más diversa.

Ken Thompson y Dennis Ritchie reescribieron el sistema operativo utilizando el lenguaje C (que ellos mismos desarrollaron), lo cual permitió hacer el sistema *portable* a diferentes arquitecturas de hardware. Rápidamente UNIX creció y se desarrollo en dos ramas: El **system V** rama comercial del sistema, desarrollada por varias compañias como Sun Microsystems (Solaris), HP (HP/UX) y Silicon Graphics (IRIX). La rama *académica* BSD (Berkeley Software Distribution), versión no comercial de UNIX, de la cual derivan muchas distribuciones como FreeBSD y OpenBSD (con versiones para múltiples plataformas), así como, sistemas comerciales como Apple OS X.

En 1984, Richard Stallman, fundó el proyecto **GNU** que se basa en toda una filosofía sobre el desarrollo del software. La idea de Stallman es crear un sistema operativo que sustituya a UNIX pero de código abierto, en el cual el usuario disponga del código fuente de los programas y los pueda modificar y redistribuir. El sistema GNU se comenzó a desarrollar por la capa más externa (el entorno del sistema) y en la actualidad el kernel Hurd (basado en el microkernel Mach[4]), aún esta en etapa de desarrollo. Los programas GNU son ampliamente utilizados en los propios sistemas UNIX y al ser de fuente abierta las compañias distribuyen sus propias vesiones de los paquetes GNU.

En 1992 Linus Torvalds publicó a través de internet un kernel de sistema operativo (clon de UNIX, escrito en lenguaje C), rápidamente se difundió y desarrollo en la comunidad especializada. El proyecto GNU vio la oportunidad de utilizar el kernel de fuente abierta en su sistema, rápidamente surgieron varias distribuciones GNU/Linux, entre las más populares se encuentran Debian, SUSE, RedHat y Mandrake.

En la actualidad el árbol genealógico de los sistemas UNIX es muy complicado (ver figura 1.2), existen alrededor de 47 distribuciones GNU/Linux, muchas de ellas se pueden descargar de la red. Las compañias han adoptado como estrategia comercial mantener dos frentes de batalla contra el monopolio de Microsoft, por un lado distribuyen una versión comercial, pero también apoyan proyectos de código libre. Por ejemplo, RedHat tiene el proyecto Fedora y Apple tiene el proyecto Darwin. A los científicos nos interesa el sistema operativo GNU/Linux por su flexibilidad, estabilidad y por las herramientas que proporciona. **Ekbek** el nuevo cluster[5] de computadoras del Laboratorio de Supercmputo Astrofisco (**LaSumA**), utiliza una distribución basada

[4]Mach es un proyecto de la universidad Carnegie Mellon.

[5]Cuenta con 32 servidores HP ProLiant ML330 G3 con procesadores duales intel Xeon EM64T de 3.0 Ghz y 4 GB de RAM, lo cual lo hace uno de los clusters más potentes del país.

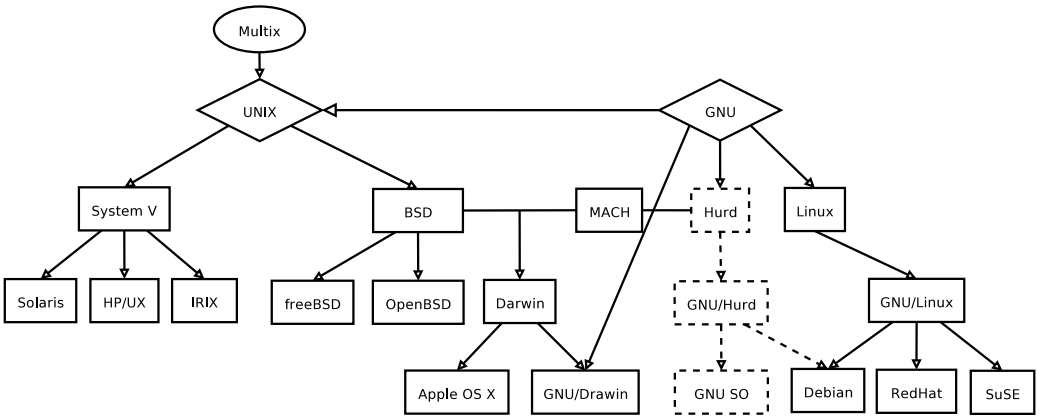

Figura 1.2: Árbol genealógico de los sistemas UNIX.

en RedHat Enterprise, especialmente diseñada para clusters, llamada Rocks Cluster Distribution. Esta versión de GNU/Linux es utilizada por cerca de 519 laboratorios y compañias, es desarrollada por la UC San Diego, por la UC Berkeley y apoyada por compañias como HP y Dell.

Capítulo 2

Comandos básicos

En esta sección se presentan los comandos básicos y las opciones más útiles de los mismos, se agrupan por la función que desempeñan. Se pretende dar una guia de referencia para tener a la mano a la hora de trabajar en ekbek.

2.1. Comandos de ayuda

La mayoría de los programas y comandos aceptan la opción `--help` la cual proporciona una lista de las opciones del comando. Por ejemplo:

```
ls --help
```

Sin embargo la mejor referencia sobre como utilizar un comando es el manual, el cual se invoca con el comando `man <comando>` o con la versión más reciente: `info <comando>`. Por ejemplo:

```
man ls
info ls
```

siempre es útil acudir al manual como primer fuente de información, sin embargo existen otras referencias útiles como libros o la internet.

2.2. Manejo de archivos y directorios

A continuación se listan algunos comandos para el manejo de archivos y directorios, así como algunas de las opciones más útiles[1].

[1] [ ] indica un parámetro opcional,

- `ls [opciones] [archivos]`; -(**l**i**s**t)- Lista el contenido de un directorio, mostrando todos los archivos que cumplan cierto patrón, ejemplo:
  `ls -a` muestra todos los archivos del directorio actual.
  `ls -l -h *.for` muestra todos los atributos (-l) de todos (*) los archivos que terminan con .for mostrando el tamaño (-h) en Bytes, KBytes, etc.

- `sort [opciones [archivo]`; Ordena alfabéticamente el contenido de un archivo, si no se da un archivo, se ordena lo que se escribe desde el teclado[2] hasta dar la combinación: ctrl+d, ejemplo:
  `sort -n -r milista.l` Ordena el contenido del archivo *milista.l* numéricamente (-n) y en orden inverso (-r).

- `mkdir [opciones] directorio`; Crea un directorio.

- `rmdir [opciones] directorio`; Borra directorios.

- `cd [directorio]`; Cambia de directorio, sin argumentos se llega a home.

- `tree [directorio]`; Muestra la estructura de directorios y archivos en forma esquemática.

- `cp archivo1 ruta/[archivo2]`; Copia el archivo1 en un directorio (ruta) y lo renombra (archivo2).

- `rm archivo`; borra archivos, la opción -r hace el proceso recursivo para borrar todo un directorio. Ejemplo:
  `rm -r Documentos/*.c` borra todos los archivos que terminan en .c aún si son directorios o están contenidos en otros directorios.

- `mv archivo1 archivo2` **m**ue**v**e archivos o directorios de archivo1 a archivo2, si el archivo permanece en el mismo directorio equivale a cambiar de nombre al archivo.

- `more archivo`; muestra el contenido de un archivo de texto por paginas.

- `less archivo`; similar a more (solo en sistemas GNU).

[2]En realidad lo que hay en la entrada estándar.

- `cat archivo1 archivo2 ...` muestra el contenido de uno o varios archivos sin paginar.
- `touch archivo`; cambia la fecha de los archivos, si no existe crea un archivo vacío.
- `locate archivo`; localiza un archivo.
- `whereis programa`; muestra la ubicación de un programa, debe estar en un directorio contenido en PATH[3]
- `file archivo`; describe el tipo de archivo, por ejemplo: imagen, de texto, etc. Nota: file identifica el tipo de archivo aún si su extensión no es la correcta.
- `whatis comando` muestra la descripción de comando.
- `wc [opciones] [archivo]`; **w**ord **c**ount, cuenta las palabras (-w), lineas (-l), caracteres (-m) o bytes (-c) en un archivo de texto.
- `head archivo`; muestra el inicio de un archivo
- `tail archivo`; muestra el final de un archivo
- `ln archivo link` crea enlaces (links) a archivos o carpetas, la opción -s hace un link simbólico.
- `diff archivo1 archivo2` muestra las diferencias entre dos archivos
- `tar [opciones] directorio.tar directorio` empaqueta o desempaqueta archivos .tar, ejemplo:
  `tar -cvf mitar.tar [midirectorio]` empaqueta midirectorio en el archivo mitar.tar, cambiando c por x y omitiendo midirectorio se desempaqueta.
- `gzip archivo.[gzip]` comprime o descomprime (-d) archivos gz

[3]Ver variables de ambiente, más adelante.

2.3. Comandos de usuario I

- `passwd`; Permite cambiar la contraseña.
- `su [usuario]`; **s**wich **u**ser, cambia de usuario
- `whoami` ; muestra el nombre de usuario efectivo.
- `logname` ; muestra el nombre de usuario.
- `id [usuario]`; muestra datos de identificación del usuario (uid, gid, grupos, etc.)
- `finger usuario`; da información del usuario (login, nombre, shell, directorio, etc.)

Capítulo 3

Comando útiles para trabajar con el cluster

Esta sección es de nivel intermedio por lo que se utilizarán algunos términos técnicos sobre redes, en la bibliografía y enlaces recomendados hay más información al respecto.

3.1. Variables de ambiente

Los compiladores, bibliotecas y otros porgramas (como CACTUS), utilizan variables de ambiente, las cuales indican a los programas en que parte del sistema se encuentra cierta información, como por ejemplo: la *ruta* de acceso a un programa, a bibliotecas o alguna opción relevante (el nombre de un compilador).

3.1.1. Algunas variables de ambiente

`PATH` es una variable que indica las ruta de acceso para encontrar un comando, un valor típico de esta variable es el siguiente:

`/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin`

nótese que el separador de campo son dos puntos, cada campo contiene una dirección en el sistema de archivos en donde puede estar el comando, si deseamos ejecutar un programa que no se encuentra en el `PATH` debemos especificar la ruta completa, por ejemplo:

`/opt/mpich/gnu/bin/mpirun`; suponiendo que estamos en otro directorio.

`./holamundo.f90`; suponiendo que holamundo.f90 se encuentra en el directorio actual.

para añadir un directorio a nuestra ruta predeterminada se utiliza el siguiente comando: `PATH=$PATH:/mi/nuevo/directorio` el modificador $ regresa el valor de la variable como se vera más a delante.

Otras variables de ambiente se listan a continuación:

- `HOME`; ruta de acceso al directorio principal del usuario.
- `LOGNAME`; nombre de inicio de sesión.
- `SHELL`; ruta al programa del shell actual.
- `PS1`; indicativo principal de la linea de comandos de la shell.
- `PS2`; indicativo secundario de la linea de comandos de la shell.
- `TERM`; nombre de la terminal.

3.1.2. Comandos y variables de ambiente

Los siguientes comandos nos permiten ver y modificar las variables de ambiente:

- `echo $VARIABLE`; echo hace eco de lo que se escribe a continuación, sin embargo con el modificador $ nos permite ver el valor de la variable.
- `export VARIABLE`; exporta el valor de la variable a los *subshells* que generemos a partir del shell actual. (permite *ver* a otros programas un nuevo valor dado a una variable.
- `env`; muestra el valor de cada una de las variables de ambiente.

como nota final cabe mencionar que se pueden generar nuevas variables de ambiente, que por convención se escriben siempre con mayúsculas, por ejemplo:

```
MI_VARIABLE='Valor de mi variable'
```

3.1.3. Archivos de configuración del shell

Al iniciar una nueva sesión se leen archivos de configuración del shell, para el shell bash: `.bash_profile` o `.bashrc`, en estos archivos podemos añadir rutas a directorios (por ejemplo un comando: `export PATH=$PATH=:/opt/mpich` o *alias* a comandos (por ejemplo `alias l='ls -l'`).

3.2. Comandos de usuario II

- `who`; muestra que usuarios están conectados al sistema y desde donde.
- `w`; muestra los usuarios conectados y el comando que ejecutan.
- `last`; muestra información de los últimos usuarios que han usado el sistema.
- `mail usuario@host`; programa de correo electrónico (nota: se utiliza la combinación `ctrl+d` para terminar)
- `write usuario`; manda un mensaje a la pantalla de un usuario.

3.3. Comandos de procesos

- `top [opciones]`; monitorea los procesos que se están ejecutando.
- `ps [opciones]`; muestra una lista de procesos del usuario, -A muestra todos los procesos, aux muestra una lista más detallada.
- `kill -s señal (PID)` envía una señal a un proceso dando como argumento el numero de proceso (PID), la opción -l lista los tipos de señales, en `info kill` se encuentra el significado de cada señal. Ejemplo, uso de la señal de terminación de proceso: `kill -s 9 (PID)`
- `time` mide el tiempo que tarda un proceso en ejecutarse.
- `fg` trae a primer plano un proceso que se encuentra en segundo plano.
- `bg` pone un proceso en segundo plano.
- `comando &` colocado al final de la linea de comando ejecuta en segundo plano.

- `nice -n (nivel) (comando)` ejecuta un proceso con una prioridad mas baja, nivel es un entero entre 0 y 30.

3.4. Comandos de sistema

- df; muestra el espacio libre de los discos.
- `du`; muestra de forma detallada el espacio usado por un directorio.
- `uptime` ;muestra el tiempo transcurrido de encendida la maquina.
- `hostname`; muestra el nombre del servidor.
- `free`; da un registro del estado de la memoria.
- `bc`; calculadora.

3.5. Comandos de red

- `ssh usuario@host`; Establece un shell remoto de forma segura.
- `scp archivo_local usuario@host:/directorio_destino`; copia un archivo de forma segura.
- `netstat`; muestra estado de la red.
- `ifconfig`; muestra la configuración del dispositivo de red, solo el usuario root.
- `ping (ip)` verifica si existe conexión con otra maquina.
- `nslookup` da la IP de una web www.xxxxxxx.com.
- `route -n` muestra la tabla de rutas, solo el usuario root.

3.6. Combinaciones de teclas

- `ctrl+L`; borra la pantalla.
- `ctrl+alt+F1`; cambia de consola.
- `ctrl+z`; suspende el proceso actual.
- `ctrl+d`; manda la señal de fin de archivo EOF.
- `ctrl+c`; termina el proceso en ejecución.

3.7. Redirección

Es posible redireccionar la salida de un comando para procesarlo por otro comando o para guardarlo en disco, solo se mencionarán dos *simbolos* especiales que cumplen esta función sin embargo en cualquier texto especializado se podrá encontrar más información. Para *guardar* la información desplegada en pantalla se puede usar el símbolo > de la siguiente forma:

`comando` > `archivo.dat`; la extensión y el nombre del archivo es cualquiera aceptado por el sistema.

También es posible *entubar* comandos utilizando el símbolo |, en este caso la sintaxis es:

`comando1` |`comando2`; la *salida* del `comando1` será procesada por el `comando2`, por ejemplo, si deseamos visualizar el contenido de un directorio con muchos archivo utilizariamos: `ls /etc/ | more`

Capítulo 4

Donde obtener más información

En esta sección se da una lista con ligas a internet de muchos sitios relacionados con el sistema operativo y el cluster.

4.1. Sistemas GNU/Linux

- La principal fuente de información sobre el sistema GNU es: `http://www.gnu.org`
- Ligas a la mayoría de las distribuciones y las imágenes iso: `http://www.linuxiso.org`
- El portal de linux: `http://www.linux.org`
- Un portal privado de linux: `http://www.linux.com`
- Un portal con enlaces: `http://www.linuxlinks.com`
- Deposito de software en rpm: `http://www.rpmfind.net`

4.2. Documentación

las principales fuentes de documentación son el proyecto `http://tldp.org` y la versión en español `http://es.tldp.org`.

4.3. Clusters

La principal fuente de información es el proyecto `http://www.lcic.org`, la pagina de la distribución utilizada `http://www.rocksclusters.org` y la pagina del proyecto CACTUS: `http://www.cactuscode.org`.

Bibliografía

[1] *Richard Petersen*, Manual de referencia, Linux. Osborne, McGraw-Hill. 1^a edición en español, 2001.

[2] *James Mohr*, Linux, recursos para el usuario. Pearson Educación. 1^a edición en español, 1999.