Introducción a Linux

Materia *Introducción a los Algoritmos*, FaMAF, UNC

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, que es *multiusuario* y *multitarea*.

  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.

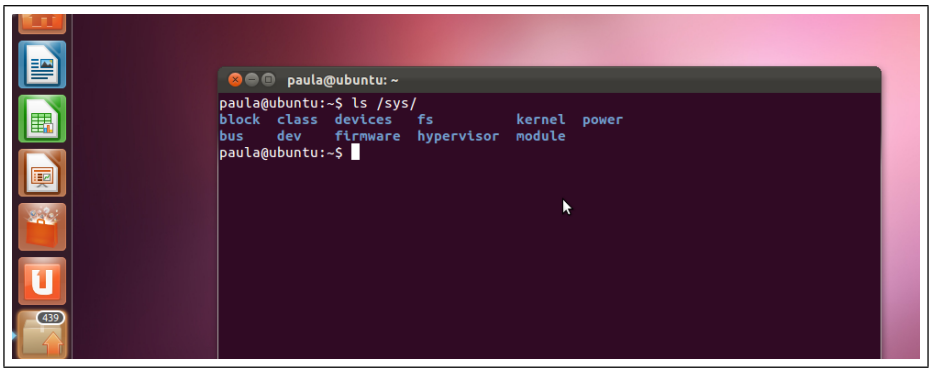

Figure 2: Acceso a la línea de comandos usando una consola en el entorno gráfico de Linux.

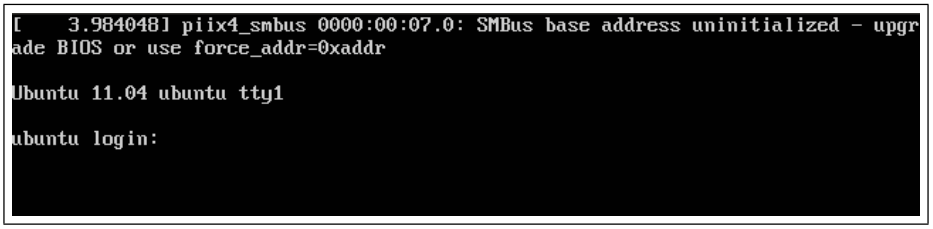

Figure 3: Acceso a la línea de comandos en una terminal (modo texto).

4 Usuarios, grupos y permisos

Hay diferentes tipos de usuarios en Linux, entre los que distinguimos los dos más importantes: Los administradores y los usuarios comunes. El usuario administrador (root) tiene control total sobre el sistema, pudiendo instalar, configurar o borrar aplicaciones, crear o borrar usuarios o archivos, instalar nuevo hardware, etc. Un usuario común en cambio sólo puede limitarse a realizar operaciones con sus propios archivos y aplicaciones, y tendrá acceso a las operaciones permitidas por el administrador a través de us sistema de permisos; esto es, en general, por medidas de seguridad (por ejemplo, no sería conveniente que cualquier alumno de famaf pudiera borrar los usuarios correspondientes a docentes o programas utilizados por otros alumnos). Para restringir el acceso de ciertos usuarios a algunos archivos o directorios hay que cambiar los permisos de cada archivo al que no queramos que dichos usuarios tengan acceso o al directorio que los contenga. Los permisos son justamente lo que la palabra indica: la autoridad que se puede ejercer ante un archivo o directorio, o ante otro. Para ver cuáles son los permisos de cada archivo/directorio debemos ejecutar el comando `ls -l`.

Suponiendo que la salida de este comando fuera la siguiente:

```
-rwxr-xr-x 3 root root 1024 Oct 14 15:44 archivo.de.prueba
```

estaríamos viendo en ese orden: lista de permisos (`-rwxr-xr-x`), número de links al archivo (`3`), usuario dueño (`root`), grupo de usuarios al que pertenece (`root`), tamaño (`1024`), fecha de creación/modificación (`Oct 14 15:44`) ynombre del archivo (`archivo.de.prueba`).

Para interpretar los permisos que cada archivo tiene, se debe tener en cuenta que: los primeros 4 parámetros corresponden al dueño del archivo; los siguientes 3 parámetros corresponden al grupo al que pertenece el usuario dueño y los últimos tres parámetros corresponden a cualquier otro usuario. Las letras significan el tipo de permiso que cada usuario o grupo tiene al archivo. La letra "r" quiere decir que el usuario solo tendrá la posibilidad de leer el contenido del archivo, pero no modificarlo ni ejecutarlo. La letra "w" le da al usuario permiso de escritura, es decir que el usuario tiene la posibilidad de modificar de cualquier manera el archivo o directorio. La letra "x" le da al usuario la posibilidad de ejecutar el archivo o directorio (con ejecutar un directorio entendemos que podemos acceder a él).

Otros ejemplos podrían ser:

`-rwx------` → el dueño puede leer el contenido (r), modificar (w) y ejecutar (x) el mismo. Los demás usuarios o participantes del mismo grupo no podrán hacer nada con este archivo.

`-rwx--x--x` → tanto el dueño, como el grupo y los demás usuarios pueden ejecutar el archivo al que se refieren. Además el dueño también puede leer y escribir el mismo archivo.

Los permisos pueden ser cambiados solamente por el usuario root o por su dueño. El comando correspondiente para cambiar los permisos de un archivo o directorio es `chmod`. Su sintaxis es `chmod [opciones] permisos archivo/directorio`. Este comando combinado con alguna de las letras indicando usuario, grupo o dueño (u, g, o) seguida del los signos '+' (dar) o '-' quitar y seguido por el tipo de permiso (r, w, x) cambia los permisos del archivo que queramos. Por ejemplo:

`chmod o-x archivo` cambiará `-rwxr-xr-x` por `-rwxr-xr--` y `chmod o+w archivo` cambiará `-rwxr-xr-x` por `-rwxr-xrwx`.

Además de cambiar los permisos de un archivo se puede cambiar el dueño del mismo con el comando `chown`. Veamos un ejemplo:

Si quiero cambiar el dueño del `archivo.de.prueba` para que ahora sea el usuario `user1` pero quiero que el grupo siga siendo el mismo entonces ejecutamos:

```
localhost:/# chown user1 archivo.de.prueba
```

Si en cambio, quiero cambiar tanto el usuario (a `user1`) como el grupo (a `linuxgroup`), debo ejecutar:

```
localhost:/# chown user1.linuxgroup archivo.de.prueba
```

Se puede conseguir más información sobre estos comandos utilizando el comando `man chown` o `man chmod`

5 Sistema multitareas

Como mencionamos anteriormente, Linux es un sistema multitarea, es decir, capaz de correr más de un programa a la vez pero no es posible que tengamos todos los programas corriendo a la vista al mismo tiempo. Linux puede hacer correr un programa en segundo plano mientras que corre uno en un primer plano. Por ejemplo, yo puedo dejar un programa compilandose en segundo plano mientras que edito otro archivo en primer plano, ya que no necesito ver cuando se esta compilando el programa. Para dejar un trabajo en segundo plano debemos simplemente agregar el signo '&' luego del nombre del programa que querramos dejar andando.

Para ver todos los procesos que se estan corriendo (no los comandos) se utiliza el comando `ps`. Para ver, en cambio, solo los comandos que se estan corriendo en segundo plano debemos ejecutar el comando `jobs`. Veamos ahora unos ejemplos:

6 Árbol de directorios

En GNU/Linux todo son archivos, esto es, cualquier elemento presente en el sistema es tratado como un archivo, desde nuestros archivos personales hasta los dispositivos de hardware como la impresora, el mouse o el disco duro. Estos archivos están organizados en lo que se conoce como un *árbol de directorios*, es decir, una jerarquía de directorios y subdirectorios destinados a fines específicos. La figura 4 muestra un ejemplo de los directorios presentes en una instalación estándar de Ubuntu.

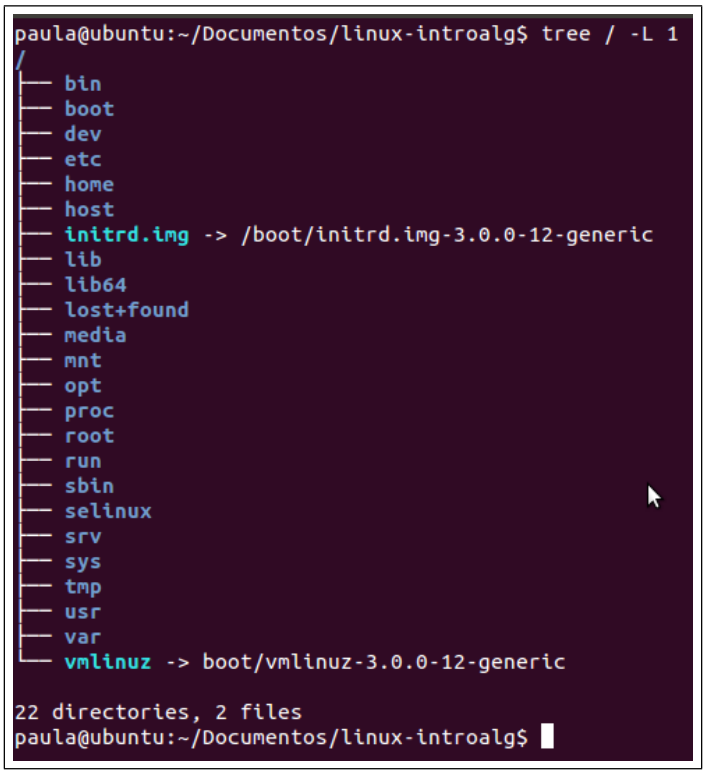

Figure 4: Arbol de directorios estándar.

Como puede observarse todos los directorios están bajo el *directorio raíz* (denotado por el símbolo) y tienen distintas utilidades, los más importantes por el momento son:

- `/bin`: contiene muchos de los programas escenciales para el funcionamiento del sistema como los comandos básicos (ls, cp, etc).
- `/dev`: contiene controladores de dispositivos, como discos rigidos, modems, memoria, mouse, etc.
- `/home`: aqui se encuentran generalmente los directorios de inicio de los distintos usuarios; es donde se guardan los archivos personales, archivos de configuración de aplicaciones, etc.
- `/root`: El directorio local para el usuario root. normalmente los demás usuarios del sistema no pueden acceder a él.

En la siguiente sección veremos algunos comandos para manipular directorios y/o archivos, y para movernos por distintos directorios. También puede utilizarse la herramienta *midnight commander* ejecutada a través del comando `mc`, que es una manera más visual de acceder al sistema de directorios y provee también menúes con muchas de las operaciones básicas.

7 Comandos básicos

7.1 El shell

En el modo texto (ya sea logueados en una terminal o ejecutando una consola dentro del modo gráfico) 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").

Para cada uno de los comandos que se mencionen en adelante se puede consultar la página de ayuda correspondiente (llamadas *páginas man*) ejecutando el comando `man comando`.

7.2 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*.

7.3 Visualizar el contenido de un directorio

Para saber cuál es el contenido del directorio en el que estamos situados existe el comando `ls` que puede mostrar simplemente los nombres de los archivos y directorios o puede proveer más información si se lo invoca con distintas opciones. Por ejemplo: estamos situados en el directorio raiz (`/`) y tenemos dentro de éste los directorios "linux", "mis archivos", "fotos" y "home" y además el archivo "linuxfile". Entonces ejecutar `ls` debería mostrar:

```
localhost:/# ls
linux mis archivos fotos home linuxfile
localhost:/#
```

Para averiguar más datos sobre los archivos y directorios el comando ls (asi como cualquier otro comando) posee diversas opciones; por ejemplo, la opción `-S` muestra los archivos ordenados por su tamaño y la opción (muy utilizada) `-l` provee información adicional sobre los archivos como fecha de modificación y dueño.

7.4 Cambiar de directorios

El comando `cd` sirve para movernos de un directorio a otro. Por ejemplo, si estamos en el directorio raíz y queremos entrar al directorio `linux` que se encuentra en dicho directorio debemos tipear:

```
localhost:/# cd linux
```

luego de haber ejecutado el comando el prompt nos muestra que pasamos a estar dentro de dicho directorio:

```
localhost:/linux#
```

Para volver un directorio más arriba ejecutamos `cd ..`

7.5 Crear directorios

Para crear un directorio se utiliza el comando `mkdir` y su sintaxis es la siguiente:

```
localhost:/# mkdir directorio_a_crear
```

Para verificar que se haya creado correctamente se puede utilizar `ls`.

7.6 Copiar archivos

Para ello se usa el comando `cp`. La sintaxis de este comando es:

```
localhost:/# cp archivo_a_copiar destino
```

Si en vez de copiar solo un archivo queremos copiar muchos podemos hacerlo solo agregando el nombre de cada archivo a copiar uno al lado del otro separados por un espacio. Si queremos que se copien todos los archivos que hay en el directorio podemos hacer directamente lo siguiente:

```
localhost:/# cp * destino
```

El asterisco abarca todos los archivos y directorios contenidos en el directorio actual. Sin embargo, los directorios no pueden copiarse igual que los archivos; para copiar todo un directorio debemos agregar la opcion `-R`, de otra manera Linux no copiará ningún directorio:

```
localhost:/# cp -R * destino
```

7.7 Mover archivos

Si lo que queremos no es copiar un archivo o carpeta sino moverlo de un lado al otro, podemos usar `mv`, de la siguiente forma:

```
localhost:/# mv archivo destino
```

Este comando no solo sirve para mover archivos o directorios de un directorio a otro, sino también para renombrarlos. Por ejemplo:

```
localhost:/# mv archivo archivo_nuevo
```

7.8 Borrar archivos

Si queremos eliminar archivos o directorios usamos el comando `rm` con la siguiente sintaxis:

```
localhost:/# rm viejo.txt
```

Al igual que con otros comandos, si queremos tratar un directorio con todos su contenido deberemos usar la opción apropiada, en este caso:

```
localhost:/# rm -R directorio
```

7.9 Visualizar archivos

Para visualizar un archivo, sin tener la posibilidad de modificar su contenido podemos usar el comando `more` seguido del nombre del archivo a ver. Por ejemplo:

```
localhost:/# more manual_linux.txt
```

Si el archivo es muy extenso y no alcanza a mostrarse enteramente en pantalla presionamos la tecla "enter" para avanzar; la tecla `b` retrocederá una página completa; la tecla `'` (apóstrofe o tilde) volverá al principio del documento la tecla `h` para obtener ayuda y la tecla `q` nos permitirá salir.

Para visualizar archivos pero además editarlos será necesario abrirlo con algún editor de texto (si el archivo es de texto), por ejemplo `pico` o `mcedit` o `emacs`.

7.10 Búsqueda de archivos

Hay veces que necesitamos buscar un archivo que contiene alguna información específica pero no sabemos dónde o con qué nombre lo guardamos. En estos casos el comando `grep` nos ayudará bastante. Este comando se encarga de buscar una patrón dentro de los archivos que especifiquemos y la forma de invocar este comando es :

`grep [opciones] patron [dónde buscar]`

Por ejemplo: `localhost:/# grep -r linux /home/paula/` busca los archivos que contengan la palabra *linux* en todos los directorios y subdirectorios dentro de `/home/paula/`; esta búsqueda recursiva se especifica con la opción `-r`.

7.11 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 redirección 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`

7.12 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

7.13 Navegador de archivos MC

Midnight Commander (MC) es una aplicación al estilo del explorador de Windows o Nautilus en el modo gráfico de Linux pero que funciona en modo texto, facilitando bastante el manejo de archivos. La pantalla

principal consiste en dos paneles en los cuales se muestra el sistema de archivos. Se usa de un modo similar a otras aplicaciones que corren en el shell o interfaz de comandos de Unix. Las teclas de cursor permiten desplazarse a través de los ficheros, la tecla insertar se usa para seleccionar ficheros y las Teclas de función (F1, F2, etc) realizan tareas tales como borrar, renombrar, editar, copiar ficheros, etc.

La forma de invocarlo es tipear en una terminal `mc` y aparecerá la aplicación que se muestra en la figura 5.

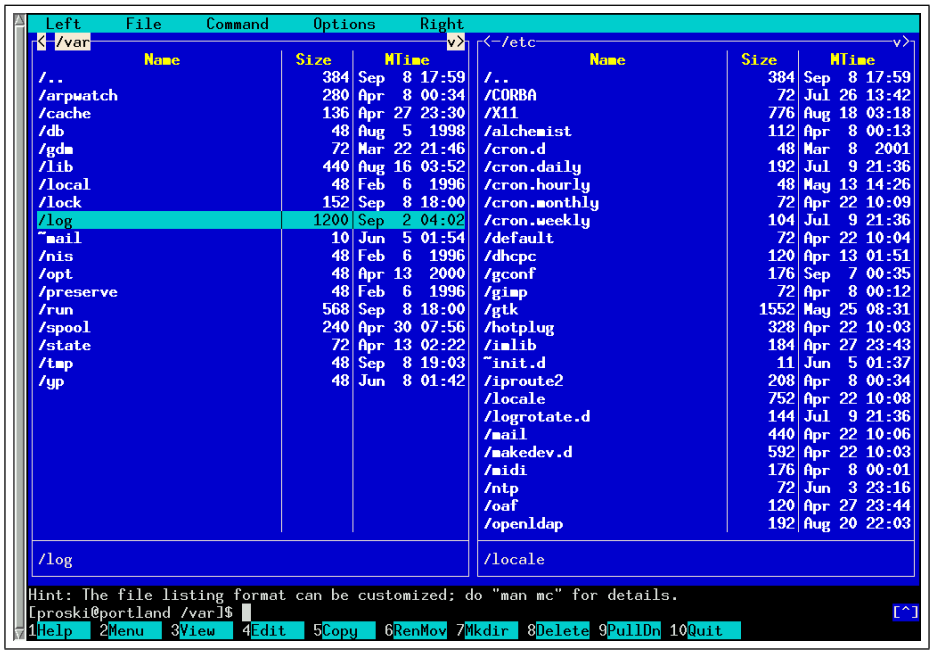

Figure 5: Pantalla del mc.

En `http://linuxeslibre.com.ar/tutos_post_mc.html` pueden encontrar más información sobre su utilización.

8 Forma de trabajo recomendada para los prácticos de la materia

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

9 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
`man [nombre comando]`	muestra información del comando seleccionado, con ejemplos de uso y opciones posibles

Table 1: Algunos comandos útiles de Linux

10 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