Cómo pasar de DOS a Linux.

Por Guido Gonzato guido@ibogfs.df.unibo.it
Traducido por David Martín Carreño, davefx@bigfoot.com v1.0, 11 de diciembre de 1996. Traducción: 8 de marzo de 1998.

Este documento Como está dedicado a todos los (¿próximamente anticuados?) usuarios de DOS que acaban de decidir pasarse a Linux, el clónico gratuito de UNIX para ordenadores x86. Dadas las similitudes entre DOS y Unix, el propósito de este documento es ayudar al lector a traducir su conocimiento de DOS al entorno Linux, con todo lo que ello lleva de productivo.

1 Introducción

1.1 ¿Es Linux adecuado para usted?

¿Quiere pasar de DOS a Linux? Buena idea, pero cuidado: puede no serle útil. Quiero decir: no hay nada que sea *el mejor ordenador* o *el mejor sistema operativo*: depende de a lo que se dedique, por lo que no creo que Linux sea la mejor solución para todos, incluso aunque sea técnicamente superior a muchos sistemas operativos comerciales. Usted se beneficiará inmensamente de Linux si lo que necesita es software para programar, Internet, TeX... software técnico en general. Pero si necesita software comercial, o si no le apetece aprender y escribir comandos, deje Linux y busque en otra parte.

Linux no es (por ahora) tan fácil de usar y configurar como Windows o el Mac, así que prepárese para trastear un poco. Después de estos avisos, déjeme decir que confío al 100% en que usted pertenece a la clase de usuario que encontrará en Linux el Nirvana informático. Está hecho para usted. Y recuerde que, de todos modos, Linux y DOS/Windows pueden coexistir en la misma máquina.

1.2 Requisitos previos para la lectura de este Como

Asumiré que:

- conoce los comandos y conceptos básicos del DOS;
- Linux, posiblemente con el sistema X Window, está adecuadamente instalado en su ordenador;
- su shell —el equivalente de `COMMAND.COM`— es `bash`;
- usted comprende que esta guía es sólo un paso incompleto. Para más información, busque en el *Linux, Instalación y Primeros Pasos* de Matt Welsh y en la *Guía del Usuario de Linux* de Larry Greenfield. Ambas, en versión inglesa, disponibles en `ftp://sunsite.unc.edu/pub/Linux/docs/LDP` y, traducidos al castellano, en la página del proyecto LuCAS: `http://www.infor.es/LuCAS`

1.3 Ya está. Ahora cuénteme más.

Supongo que acaba de instalar Linux y los programas que necesita en su ordenador, que ya tiene una cuenta propia (si no la tiene aún, ¡teclee `adduser` ahora mismo!) y que Linux está ejecutándose. Acaba de introducir su nombre y su clave, y ahora está mirando a la pantalla pensando... *¿Y ahora qué?*

Bien, no se desespere. Puede hacer casi las mismas cosas que solía hacer con DOS, y muchas más. Si estuviera ejecutando DOS en lugar de Linux, estaría realizando alguna de las siguientes tareas:

- ejecutar programas y crear, copiar, ver, borrar, imprimir, renombrar ficheros;
- cambiar de directorio, crearlos, borrarlos y listar sus contenidos;

- formatear disquetes y copiar ficheros de/hacia ellos;
- editar el `AUTOEXEC.BAT` y el `CONFIG.SYS`;
- escribir sus propios ficheros `.BAT` y/o programas Qbasic;
- el restante 1%.

Estará contento al saber que todas esas tareas pueden llevarse a cabo bajo Linux de una manera similar a como se hacen bajo DOS. Bajo DOS, el usuario medio usa muy pocos de los más de 100 comandos disponibles: lo mismo, hasta cierto punto, sucede con el Linux.

Unas pocas cosas que señalar antes de continuar:

- primero, cómo salir. Para apagar Linux: si ve una pantalla en modo texto, pulse Ctrl-Alt-Supr, espere a que el sistema realice unas tareas y le diga que todo está bien, y entonces apague el ordenador. Si está trabajando bajo el sistema X Window, pulse primero Ctrl-Alt-Del, y después Ctrl-Alt-Supr. Nunca apague o reinicie el ordenador directamente: el sistema de ficheros podría dañarse.
- al contrario que en DOS, Linux tiene mecanismos de seguridad intrínsecos, debido a su naturaleza multiusuario. Los ficheros y los directorios tienen permisos asociados a ellos, y por lo tanto el usuario normal puede no tener acceso a alguno de ellos; (ver la Sección 2.3). Sólo el usuario cuyo nombre de acceso sea `root` tiene el poder absoluto. (Esta persona es el administrador del sistema. Si trabaja en un ordenador propio, usted también será root). DOS, por el contrario le dejaría machacar todos los contenidos de su disco duro.
- si está realmente animado a experimentar, inténtelo usted mismo: seguramente no podrá hacer ningún daño. Puede conseguir alguna ayuda escribiendo en la línea de comandos ($ es el símbolo de la línea de comandos estándar, # es el de root);

  ```
  $ help
  ```

  Esto le porporcionará ayuda acerca de `bash`; puede conseguir información acerca de un comando concreto escribiendo:

  ```
  $ man comando
  ```

  el cual, si tiene instaladas las páginas man, ejecutará la página de manual `man` asociada al comando. También puede probar con:

  ```
  $ apropos comando
  $ whatis comando
  ```

  y presionar `q` para salir;
- la mayor parte del poder y la flexibilidad de Unix viene de los conceptos simples de redirección y *piping*, o entubamiento, más potentes que bajo DOS. Los comandos simples pueden agruparse para realizar tareas complejas. ¡Use estas características!
- **Convenciones**: <...> significa algo que debe ser especificado, mientras que `[...]` es algo opcional. Ejemplo:

  ```
  $ tar -tf <fichero.tar> [> fichero_redir]
  ```

  `fichero.tar` debe ser indicado, pero la redirección a `fichero_redir` es opcional.
- desde ahora *LPM* significa *para más información Lea las Páginas del Manual*.

1.4 Para el Impaciente

¿Quiere empezar ya? Eche un vistazo a esto:

DOS	Linux	Notas
BACKUP	tar -Mcvf dispositivo dir/	completamente distinto
CD nombredir\	cd nombredir/	casi la misma sintaxis
COPY fich1 fich2	cp fich1 fich2	idem
DEL fichero	rm fichero	cuidado no hay undelete
DELTREE directorio	rm -R directorio/	idem
DIR	ls	no exactamente la misma sintaxis
EDIT fichero	vi fichero	creo que no le gustara
	emacs fichero	este es mejor
	joe fichero	mas parecido al edit del DOS
FORMAT	fdformat	
	mount, umount	sintaxis bastante distinta
HELP comando	man comando	misma filosofia
MD directorio	mkdir directorio/	casi la misma sintaxis
MOVE fich1 fich2	mv fich1 fich2	idem
NUL	/dev/null	idem
PRINT fichero	lpr fichero	idem
PRN	/dev/lp0,	
	/dev/lp1	idem
RD directorio	rmdir directorio/	idem
REN fich1 fich2	mv fich1 fich2	no para varios ficheros
RESTORE	tar -Mxpvf device	sintaxis distinta
TYPE fichero	less fichero	mucho mejor
WIN	startx	¡mundos aparte!

Si necesita más que una tabla de comandos, continúe con las secciones siguientes.

2 Ficheros y Programas

2.1 Ficheros: Nociones preliminares

Linux tiene un sistema de ficheros —la estructura de directorios y los ficheros que contienen— muy similar al del DOS. Los ficheros tienen nombres que obedecen unas normas especiales, están guardados en directorios y algunos son ejecutables, y entre éstos , la mayoría tiene opciones en la línea de comandos. Incluso puede utilizar comodines, redirección y tuberías como en DOS. Sólo hay unas pocas diferencias:

- bajo DOS, los nombres de ficheros están en el llamado formato 8.3; por ejemplo `NOTENOUG.TXT`. Bajo Linux podemos hacerlo mejor. Si instaló Linux usando un sistema de ficheros tal como *ext2* o *umsdos*, puede utilizar nombres más largos (hasta 255 caracteres), y con más de un punto en ellos: por ejemplo, `Este.es.un.nombre.de.fichero.MUY.largo`. Dése cuenta de que he utilizado tanto mayúsculas como minúsculas: en efecto...
- Linux es sensible a las mayúsculas y las minúsculas en los nombres de ficheros o comandos. De hecho, `FICHERO.tar.gz` , `Fichero.tar.gz` y `fichero.tar.gz` son tres ficheros distintos. `ls` es un comando, `LS` sería un error;
- no hay extensiones obligadas como `.COM` y `.EXE` para los programas, o `.BAT` para los ficheros de procesamiento por lotes. Los ficheros ejecutables se marcan con un asterisco. Por ejemplo:

```
$ ls -F
cindy.jpg    cjpg*    Soy_un_directorio/  mi_1er_script*  old~
```

Los ficheros `cjpg*` y `mi_1er_script*` son "programas" ejecutables. Bajo DOS, las copias de seguridad de los ficheros suelen acabar en `.BAK`, mientras que bajo Linux acaban con un gurruño `~`. Un fichero cuyo nombre empieza con un punto es considerado como oculto. Ejemplo: el fichero `.Soy.un.fichero.oculto` no se mostrará a un comando `ls`;

- Las opciones de los programas bajo DOS se obtienen con `/opción`. En Linux se obtienen con `-opción` o `--opción`. Ejemplo: `dir /s` se convierte en `ls -R`. Fíjese en que muchos programas DOS (como `PKZIP` o `ARJ`) utilizan opciones de tipo Unix.

Puede ahora saltar a las Sección 2.4, pero yo de usted seguiría leyendo.

2.2 Enlaces simbólicos

Unix tiene un tipo de fichero que no existe bajo DOS: el enlace simbólico. Puede pensar que es un puntero o enlace a un fichero o a un directorio y que puede utilizarse en lugar del fichero o del directorio al que apunta; es similar a los "Accesos Directos" de Windows 95. Ejemplos de enlaces simbólicos son `/usr/X11`, que apunta a `/usr/X11R6`; `/dev/modem`, que apunta a `/dev/cua0` o a `/dev/cua1`, según donde esté el módem.

Para crear un enlace simbólico:

```
$ ln -s <fichero_o_directorio> <nombre_del_enlace>
```

Ejemplo:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

Ahora puede referirse a `g77manual.txt` en lugar de `/usr/doc/g77/DOC`.

2.3 Permisos y Propiedades

Los ficheros y directorios de DOS tienen los siguientes atributos: `A` (fichero), `H` (oculto), `R` (sólo-lectura), y `S` (sistema). Sólo `H` y `R` tienen sentido bajo Linux: los ficheros ocultos comienzan con un punto, y los de sólo lectura, tienen activado el permiso "`r`".

Bajo Unix un fichero tiene *permisos* y un propietario, que pertenece a un grupo. Mire este ejemplo:

```
$ ls -l /bin/ls
-rwxr-xr-x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

El primer campo contiene los permisos del fichero `/bin/ls`, que pertenece a `root`, del grupo `bin`. Dejando la información restante a un lado (el libro de Matt esta ahí para ese propósito), sólo recordaré lo que significa `-rwxr-xr-x` (de izquierda a derecha):

`-` es el tipo de fichero (`-` = fichero normal, `d` = directorio, `l` = enlace, etc.); `rwx` son los permisos del propietario del fichero (leer, escribir, ejecutar); `r-x` son los permisos para el grupo del propietario del fichero (leer y ejecutar); (no tocaré el concepto de grupo, puede pasar sin él mientras sea un novato ;-) `r-x` son los permisos para todos los demás usuarios (leer, ejecutar).

A eso se debe el porqué no puede borrar el fichero `/bin/ls` a menos que sea root: no tiene el permiso de escritura para hacer eso. Para cambiar los permisos de un fichero, el comando es:

```
$ chmod <quienXperm> <fichero>
```

donde `quien` podría ser:

- `u` usuario, que es el propietario,
- `g` (grupo),
- `o` (otros).

`X` puede ser tanto `+` como `-`, y `perm` puede ser:

- `r` (lectura),
- `w` (escritura)
- `x` (ejecución).

Ejemplos:

```
$ chmod u+x fichero
```

esto habilita el permiso de ejecución para el propietario del fichero. Atajo: `chmod +x fichero`.

```
$ chmod go-wx fichero
```

esto quita el permiso de escritura y de ejecución para todo el mundo menos al usuario.

```
$ chmod ugo+rwx fichero
```

esto le da a todo el mundo el permiso de lectura, escritura y ejecución.

```
$ chmod +s fichero
```

esto convierte al fichero en *setuid* o *suid*, esto es, un fichero que al ejecutarse lo hace con privilegios de root.

Una manera más corta de referirse a los permisos es con números: `rwxr-xr-x` puede ser expresado como `755` (cada letra corresponde a un bit: `---` es `0`, `--x` es `1`, `-w-` es `2`, `-wx` es `3`...). Parece difícil, pero con algo de práctica el concepto se domina.

`root`, al ser superusuario, puede cambiar los permisos de los ficheros de todo el mundo. Hay mucha más información acerca de esto: LPM.

2.4 Traduciendo comandos de DOS a Linux

A la izquierda, los comandos de DOS; a la derecha, sus correspondientes de Linux.

```
COPY:       cp
DEL:        rm
MOVE:       mv
REN:        mv
TYPE:       more, less, cat
```

Operadores de redirección y de tuberías:

< > >> |

Comodines: `*` `?`

```
nul: /dev/null

prn, lpt1:  /dev/lp0 o /dev/lp1; lpr
```

EJEMPLOS

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>copy joe.txt joe.doc           $ cp joe.txt joe.doc
C:\GUIDO>copy *.* total                 $ cat * > total
C:\GUIDO>copy fractals.doc prn          $ lpr fractals.doc
C:\GUIDO>del temp                       $ rm temp
C:\GUIDO>del *.bak                      $ rm *~
C:\GUIDO>move paper.txt tmp\            $ mv paper.txt tmp/
C:\GUIDO>ren paper.txt paper.asc        $ mv paper.txt paper.asc
C:\GUIDO>print letter.txt               $ lpr letter.txt
C:\GUIDO>type letter.txt                $ more letter.txt
C:\GUIDO>type letter.txt                $ less letter.txt
       idem                             $ more *.txt *.asc
       idem                             $ cat section*.txt | less
C:\GUIDO>type letter.txt > nul          $ cat letter.txt > /dev/null
```

Notas:

- `*` es más inteligente bajo Linux: `*` equivale a todos los ficheros excepto los ocultos; `.*` equivale a todos los ficheros ocultos; `*.*` equivale sólo a aquellos ficheros que tienen un punto en medio del nombre, seguido de otros caracteres; `p*r` equivaldría tanto a `peor` como a `por`; `*c*` equivaldría tanto a `pecado` como a `peca`.
- cuando se usa `more`, pulse espacio para ir leyendo a través del fichero, `q` o Ctrl-C para salir. `less` es más intuitivo y permite utilizar las teclas del cursor;
- no hay `UNDELETE`, así que piénselo dos veces antes de borrar cualquier cosa;
- además de los < > >> del DOS, Linux tiene el operador `2>` para redirigir los mensajes de error (stderr); más aún, el operador `2>&1` redirige stderr a stdout (la salida estándar), mientras que `1>&2` redirige stdout a stderr;
- Linux tiene otro comodín: los corchetes `[]`. Usar `[abc]*` equivale a los ficheros que empiezan por `a`, por `b` o por `c`; `*[I-N,1,2,3]` equivale a los ficheros que acaban por `I, J, K, L, M, N, 1, 2, 3`;
- no hay un `RENAME` como en DOS; esto es, `mv *.xxx *.yyy` no funciona;
- use `cp -i` y `mv -i` para ser avisado cuando un fichero vaya a ser sobreescrito.

2.5 Ejecución de programas: Multitarea y Sesiones

Para ejecutar un programa, escriba su nombre tal y como lo haría bajo DOS. Si el directorio (Sección 3) donde el programa está guardado está incluido en la variable de entorno `PATH` (Sección 5.1), el programa comenzará a ejecutarse. Excepción: al contrario que bajo DOS, en Linux un programa localizado en el directorio actual no se ejecutará a manos que el directorio actual (simbolizado por "`.`") esté incluido en el PATH. Para evitarlo, suponiendo que el programa se llame `prog`, teclee `./prog`.

Éste es el aspecto típico de una línea de comandos:

```
$ comando -o1 -o2 ... -on par1 par2 ... parn  < input > output
```

donde `-o1`, ..., `-on` son las opciones del programa, `par1`, ..., `parn` son los parámetros del programa. Puede encadenar varios comandos en la línea de comandos:

```
$ comando1 ; comando2 ; ... ; comandoN
```

Esto es todo acerca de ejecutar comandos, pero es fácil ir un paso más allá. Una de las principales razones para usar Linux es que es un sistema operativo multitarea —puede ejecutar varios programas (a partir de ahora, procesos) a la vez—. Puede lanzar procesos en segundo plano (*background*) y seguir trabajando inmediatamente. Más aún, Linux permite tener varias sesiones abiertas simultáneamente: es como tener muchos ordenadores en los que trabajar a la vez.

- Para cambiar a la sesión 1..6:

  ```
  $ Alt-F1 ... Alt-F6
  ```

- Para comenzar una nueva sesión sin dejar la actual:

  ```
  $ su - <mi_nombre_de_usuario>
  ```

  Ejemplo:

  ```
  $ su - root
  ```

  Esto es útil, por ejemplo, cuando se necesita montar un disco (Sección 4): normalmente, sólo `root` puede hacer eso.

- Para acabar una sesión:

  ```
  $ exit
  ```

  Si hay trabajos parados (ver más abajo), será avisado.

- Para lanzar un proceso en primer plano:

  ```
  $ nomprog [-opciones] [parametros] [< input] [> output]
  ```

- Para lanzar un proceso en segundo plano, añada un *ampersand*: `&`, al final de la línea de comandos:

  ```
  $ nomprog [-opciones] [parametros] [< input] [> output] &
  [1] 123
  ```

  el shell o intérprete de comandos identifica el proceso con un número de trabajo (p.e. 1; ver más abajo), y con un PID (123 en nuestro ejemplo).

- Para ver cuántos procesos hay:

  ```
  $ ps -a
  ```

  Esto generará una lista de procesos actualmente en ejecución.

- Para matar un proceso:

  ```
  $ kill <PID>
  ```

Puede necesitar matar un proceso cuando no sabe cómo cerrarlo de la manera correcta... ;-). A veces, un proceso solo podrá ser matado con alguna de las siguientes instrucciones:

```
$ kill -15 <PID>
$ kill -9 <PID>
```

Además, el intérprete de comandos permite suspender temporalmente (parar) un proceso, mandar un proceso al segundo plano, y traer un proceso del segundo plano al primer plano. En este contexto, los procesos son denominados *trabajos*.

- Para ver cuántos trabajos hay:

  ```
  $ jobs
  ```

  aquí los trabajos son identificados por su número de trabajo, no por su PID.

- Para parar un proceso ejecutándose en primer plano (no siempre funciona):

  ```
  $ Ctrl-C
  ```

- Para suspender un proceso ejecutándose en primer plano:

  ```
  $ Ctrl-Z
  ```

- Para mandar un proceso suspendido al segundo plano (convirtiéndolo en trabajo):

  ```
  $ bg <trabajo>
  ```

- Para traer un trabajo al primer plano:

  ```
  $ fg <trabajo>
  ```

- Para matar un trabajo:

  ```
  $ kill <%trabajo>
  ```

  donde `trabajo` puede ser 1, 2, 3; el `%` indica que nos referimos a un número de trabajo, y no a un PID. Usando estos comandos puede formatear un disco, comprimir un puñado de ficheros, compilar un programa, y descomprimir un fichero simultáneamente, y todavía tener la línea de comandos a su disposición. Inténtelo con el DOS. Inténtelo con Windows, sólo para ver la diferencia de prestaciones (siempre que no se le cuelgue, claro).

2.6 Ejecutando Programas en Ordenadores Remotos

Para ejecutar un programa en una máquina remota cuya dirección IP es `remote.bigone.edu`, teclee:

```
$ slogin remote.bigone.edu -l <login_en_maquina_remota>
```

Tras meter su password, arranque su programa favorito. Obviamente, debe tener una cuenta en la máquina remota.

Si tiene X11, puede incluso ejecutar una aplicación X en un ordenador remoto, mostrándolo en su pantalla de X. Supongamos `remote.bigone.edu` la máquina X remota y `local.linux.box` su máquina Linux. Para ejecutar desde `local.linux.box` un programa X que reside en `remote.bigone.edu`, haga lo siguiente:

- arranque las X, arranque un xterm o un emulador de terminal equivalente, y entonces teclee:

```
$ xhost +remote.bigone.edu
$ slogin remote.bigone.edu -l <login_en_maquina_remota>
```

- tras meter su password, teclee:

```
remote:$ DISPLAY=local.linux.box:0.0
remote:$ programa &
```

  (en vez de `DISPLAY...`, puede que tenga que escribir `setenv DISPLAY local.linux.box:0.0`. Depende del shell remoto).

Ahora `programa` comenzará en `remote.bigone.edu` y se mostrará en su máquina. Aunque mejor no intente esto en una línea ppp.

3 Gestión de Directorios

3.1 Directorios: Nociones preliminares

Hemos visto las diferencias entre los ficheros de DOS y Linux. Entre directorios, bajo DOS el directorio raíz es \ y bajo Linux es / . De manera similar, los directorios anidados se separan mediante \ en DOS y mediante / en Linux. Ejemplo de rutas de fichero:

DOS: `C:\PAPERS\GEOLOGY\MID_EOC.TEX`

Linux: `/home/guido/papers/geology/mid_eocene.tex`

Como bajo DOS, `..` es el directorio padre y `.` es el directorio actual. Recuerde que el sistema no le dejará hacer `cd`, `rd` o `md` donde usted quiera. Cada usuario comienza desde su propio directorio llamado `~/`. En el ejemplo anterior, éste es `/home/guido`.

3.2 Permisos en los directorios.

Los directorios también tienen permisos. Lo que hemos visto en la Sección 2.3 también rige para los directorios (usuario, grupo, y otros). Para un directorio, `rx` significa que puede cambiar a ese directorio, y `w` significa que puede crear o borrar ficheros en el directorio (según los permisos de los ficheros, por supuesto), o el directorio mismo.

Por ejemplo, para prevenir que otros usuarios husmeen en `/home/guido/text`:

```
$ chmod o-rwx /home/guido/text
```

3.3 Equivalencia de comandos de DOS a Linux

```
DIR:            ls, find, du
CD:             cd, pwd
MD:             mkdir
RD:             rmdir
DELTREE:        rm -R
MOVE:           mv
```

EJEMPLOS

```
DOS                                     Linux
---------------------------------------------------------------------
```

```
C:\GUIDO>dir                          $ ls
C:\GUIDO>dir file.txt                 $ ls file.txt
C:\GUIDO>dir *.h *.c                  $ ls *.h *.c
C:\GUIDO>dir/p                        $ ls | more
C:\GUIDO>dir/a                        $ ls -l
C:\GUIDO>dir *.tmp /s                 $ find / -name "*.tmp"
C:\GUIDO>cd                           $ pwd
        n/a - ver nota                $ cd
        idem                          $ cd ~
        idem                          $ cd ~/temp
C:\GUIDO>cd \otros                    $ cd /otros
C:\GUIDO>cd ..\temp\trash             $ cd ../temp/trash
C:\GUIDO>md newprogs                  $ mkdir newprogs
C:\GUIDO>move prog ..                 $ mv prog ..
C:\GUIDO>md \progs\turbo              $ mkdir /progs/turbo
C:\GUIDO>deltree temp\trash           $ rm -R temp/trash
C:\GUIDO>rd newprogs                  $ rmdir newprogs
C:\GUIDO>rd \progs\turbo              $ rmdir /progs/turbo
```

Notas:

1. cuando se use `rmdir`, el directorio a borrar debe estar vacío. Para borrar un directorio y todos sus contenidos, use `rm -R` (bajo su propia responsabilidad).
2. el carácter `~` es un atajo para el nombre de su directorio de usuario. Los comandos `cd` o `cd ~` le llevarán a su directorio personal desde dondequiera que esté; el comando `cd ~/tmp` le llevará a `/home/su_directorio_de_usuario/tmp`.
3. `cd -` "deshace" el último cd.

4 Disquetes, discos duros y otros métodos de almacenamiento.

4.1 Administración de dispositivos

Nunca habrá pensado acerca de ello, pero el comando de DOS `FORMAT A:` hace mucho más de lo que parece. De hecho, cuando ordene el comando `FORMAT`:

1. Formateará físicamente el disco
2. Creará el directorio `A:` (creará un sistema de ficheros)
3. Pondrá el disco disponible para el usuario (montará el disco).

Estos tres pasos se ordenan separadamente bajo Linux. Puede usar disquetes con formato MS-DOS, aunque haya otros formatos disponibles y sean mejores (el formato MS-DOS no le dejará usar nombres de fichero largos). A continuación se explica cómo preparar un disco (necesitará iniciar una sesión como root):

- Para formatear un disquete estándar de 1.44 megas (A:):

```
# fdformat /dev/fd0H1440
```

- Para crear un sistema de ficheros:

```
# mkfs -t ext2 -c /dev/fd0H1440
```

o para crear un sistema de ficheros MS-DOS:

```
# mformat a:
```

Antes de usar el disco, debe montarlo.

- Para montar el disco:

```
# mount -t ext2 /dev/fd0 /mnt
```

o

```
# mount -t msdos /dev/fd0 /mnt
```

Ahora puede dirigirse a los ficheros del disquete. Cuando haya acabado, antes de sacar el disco deberá desmontarlo.

- Para desmontar el disco:

```
# umount /mnt
```

Ahora puede extraer el disco. Obviamente, debe hacer un `fdformat` y un `mkfs` sólo a los discos no formateados, que no han sido usados nunca. Si quiere utilizar la unidad B:, ponga `fd1H1440` y `fd1` en lugar de `fd0H1440` y `fd0` en los ejemplos anteriores.

Todo lo que solía hacer con A: y B: se hace ahora utilizando `/mnt` en su lugar. Ejemplos:

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>dir a:                         $ ls /mnt
C:\GUIDO>copy a:*.*                     $ cp /mnt/* /docs/temp
C:\GUIDO>copy *.zip a:                  $ cp *.zip /mnt/zip
C:\GUIDO>a:                             $ cd /mnt
A:>_                                    /mnt/$ _
```

No hace falta decir que la manera de proceder con los disquetes también funciona con otros dispositivos, como por ejemplo, otro disco duro o una unidad CD-ROM. Esto es para montar el CD-ROM:

```
# mount -t iso9660 /dev/cdrom /mnt
```

Ésta era la manera "oficial" de montar discos, pero hay un truco. Como es algo incómodo tener que ser root para montar un disquete o un CD-ROM, puede darse permisos a cada usuario de esta manera:

- como root, crear los directorios `/mnt/floppy`, `mnt/a:`, y `/mnt/cdrom`
- añadir en `/etc/fstab` las siguientes líneas:

```
/dev/cdrom        /mnt/cdrom    iso9660   ro,user,noauto          0       0
/dev/fd0          /mnt/a:       msdos     user,noauto             0       0
/dev/fd0          /mnt/floppy   ext2      user,noauto             0       0
```

Ahora, para montar un disquete MS-DOS, un disquete ext2, y un CD-ROM:

```
$ mount /mnt/a:
$ mount /mnt/floppy
$ mount /mnt/cdrom
```

Cualquier usuario puede acceder a `/mnt/floppy`, `/mnt/a:`, y `/mnt/cdrom`. Para escribir en `/mnt/floppy` sin ser root, después de preparar el floppy es necesario hacer:

```
# mount /mnt/floppy
# chmod 777 /mnt/floppy
# umount /mnt/floppy
```

Recuerde que si considera importante la seguridad, dejar que todo el mundo pueda montar discos de esta manera constituye un buen agujero en la misma.

4.2 Copias de Seguridad

Ahora que sabe cómo se manejan disquetes, etc. un par de líneas bastan para ver cómo hacer una copia de seguridad. Hay muchos paquetes que pueden servirle, pero lo más simple para hacer una copia de seguridad multivolumen (como root) es:

```
# tar -M -cvf /dev/fd0H1440 /directorio_a_guardar
```

Asegúrese de tener un disquete formateado en la unidad, y otros ya preparados. Para restaurar sus ficheros, inserte el primer disquete en la unidad y utilice:

```
# tar -M -xpvf /dev/fd0H1440
```

5 Personalización del Sistema

5.1 Ficheros de inicialización del sistema

Dos ficheros importantes bajo DOS son el `AUTOEXEC.BAT` y el `CONFIG.SYS`, los cuales se utilizan al rearrancar el sistema para inicializarlo, dar valores a algunas variables de entorno como `PATH` y `FILES`, y posiblemente lanzar un programa o fichero de procesamiento por lotes. Bajo Linux hay varios ficheros de inicialización, algunos de los cuales no deberían ser modificados hasta que usted supiese con seguridad lo que está haciendo. De todos modos, estos son los más importantes:

```
FICHEROS            NOTAS
/etc/inittab        ¡no tocar por ahora!
/etc/rc.d/*         idem
```

Si todo lo que necesita es establecer el `PATH` y otras variables de entorno, o desea cambiar los mensajes del login o ejecutar automáticamente un programa tras iniciar una sesión, eche un vistazo a los siguientes ficheros:

```
FICHEROS                      NOTAS
/etc/issue                    establece el mensaje de antes del login
/etc/motd                     establece el mensaje de despues del login
/etc/profile                  establece el PATH y otras variables, etc.
/etc/bashrc                   define alias y funciones, etc. (ver mas abajo)
/home/su_home/.bashrc         define sus alias y sus funciones
/home/su_home/.bash_profile   establece el entorno y ejecuta sus programas
/home/su_home/.profile        idem
```

Si el último fichero existe (fíjese en que es un fichero oculto), se leerá tras el inicio de sesión y se ejecutarán los comandos en él almacenados.

Ejemplo; mire este `.profile`:

```
# Soy un comentario
echo Entorno:
printenv | less   # equivalente del comando SET bajo DOS
alias d='ls -l'   # es facil comprender lo que es un alias
alias up='cd ..'
echo "Recuerde que su path es "$PATH
echo "Hoy es `date`"  # usa la salida del comando 'date'
echo "Que tenga un buen dia, "$LOGNAME
# Lo siguiente es una funcion del shell
ctgz() # Lista los contenidos de un fichero .tar.gz
{
  for file in $*
  do
    gzip -dc ${file} | tar tf -
  done
}
# fin de .profile
```

`PATH` y `LOGNAME`, lo adivinó, son variables de entorno. Hay muchas otras accesibles; para buscar ejemplos, LPM de aplicaciones como `less`.

6 Ficheros de Inicialización de Programas

Bajo Linux, casi todo puede ser configurado de acuerdo con sus necesidades. La mayoría de los programas tienen uno o más ficheros de inicialización con los que puede trastear, a menudo llamados `.nombreprogramarc`, situados en su directorio `home`. Los primeros que querrá modificar son los de configuración del gestor de ventanas para X-Window. Si utiliza el `fvwm2` serán:

```
/usr/X11/lib/X11/fvwm2/system.fvwmrc2
```

Para el resto de programas con el que se encontrará tarde o temprano, LPM.

7 Un poco de Programación

7.1 Los Scripts del Shell: Ficheros `.BAT` con esteroides

Si ha utilizado ficheros `.BAT` para crear atajos de largas líneas de comando (yo suelo hacerlo), el objetivo puede ser obtenido insertando las líneas de alias convenientes (ver ejemplo de más arriba) en el `profile` o en el `.profile`. Pero si sus ficheros `.BAT` son más complicados, le encantará el lenguaje de Script que el shell pone a su disposición: es tan potente como el Qbasic (o más). Tiene variables, estructuras como `while`, `for`, `case`, `if-then-else`, y montones de nuevas características: puede ser una buena alternativa a un lenguaje de programación "de verdad".

Para escribir un script —el equivalente a un fichero `.BAT` bajo DOS— todo lo que tiene que hacer es escribir un fichero ASCII estándar que contenga las instrucciones, guardarlo, y entonces hacerlo ejecutable con el comando `chmod +x fichero`. Para ejecutarlo, teclee su nombre.

Aviso: el editor del sistema se llama `vi`, y es un hecho probado que la mayor parte de los nuevos usuarios lo encuentran muy difícil de usar. No voy a explicar cómo usarlo, porque no me gusta y no lo uso. Vea el manual "*Linux: Instalación y Primeros Pasos*" de Matt Welsh (aunque sería mejor utilizar otro editor tal como `joe` o `emacs` para X). Baste decir aquí unos comandos muy básicos:

- para insertar texto, pulse `i` y después el texto;

- para salir de vi sin guardar, pulse ESC y después :q!
- para guardar y salir, pulse ESC y luego :wq

Escribir scripts bajo bash es una materia tan extensa que requeriría un libro para abarcarla toda, y no voy a profundizar más allá en este tema. Sólo daré un ejemplo de script, del cual se pueden extraer las reglas básicas:

```
#!/bin/sh
# ejemplo.sh
# Soy un comentario
# no cambie la primera linea: debe estar ahi
echo "Este sistema es: `uname -a`" # usa la salida del comando
echo "Me llamo $0" # variables intrinsecas
echo "Usted me dio $# parametros: "$*
echo "El primer parametro es: "$1
echo -n "¿Como se llama? " ; read su_nombre
echo fijese en la diferencia: "Hola, $su_nombre" # entrecomillando con "
echo fijese en la diferencia: 'Hola, $su_nombre' # entrecomillando con '
DIRS=0 ; FILES=0
for fichero in `ls .` ; do
  if [ -d ${fichero} ] ; then # si el fichero esta en el directorio
    DIRS=`expr $DIRS + 1`  # DIRS = DIRS + 1
  else if [ -f ${fichero} ] ; then
   FILES=`expr $FILES + 1`
  fi
  case ${fichero} in
    gif|*jpg) echo "${fichero}: fichero grafico" ;;
    *.txt|*.tex) echo "${fichero}: fichero de texto" ;;
    *.c|*.f|*.for) echo "${fichero}: fichero de codigo fuente" ;;
    *) echo "${fichero}: fichero generico" ;;
  esac
done
echo "Hay ${DIRS} directorios y ${FILES} ficheros"
ls | grep "ZxY--!!!WKW"
if [ $? != 0 ] ; then # Sacar el codigo del ultimo comando
  echo "ZxY--!!!WKW no encontrado"
fi
echo "ya es suficiente... Para mas informacion teclee 'man bash'."
```

7.2 Sus programas en C

Bajo Unix, el lenguaje del sistema es C, le guste o no. Pero otros lenguajes como FORTRAN, Pascal, Lisp, Basic, Perl, awk... también están disponibles.

Suponiendo que usted sepa C, hay un par de guías para aquellos que han utilizado Turbo C++ o uno de sus hermanos bajo DOS. El compilador de C se denomina gcc y carece de todas las florituras que normalmente acompañan a sus análogos bajo DOS: no tiene IDE, ni ayuda en línea, ni debugger integrado, etc... Es sólo un rústico compilador de línea de comandos, muy potente y eficiente. Para compilar el típico programa hello.c esto es lo que debe teclear:

```
$ gcc hello.c
```

lo que creará un fichero ejecutable llamado a.out. Para cambiar el nombre del ejecutable a otro nombre:

```
$ gcc -o hola hello.c
```

Para enlazar una librería al programa, añada la opción `-lnombre_de_librería`. Por ejemplo, para enlazar la librería `math`:

```
$ gcc -o mathprog mathprog.c -lm
```

(`-lnombre_libreria` fuerza a gcc a enlazar la librería `/usr/lib/libnombre_librería.a`; por lo tanto `-lm` enlaza `/usr/lib/libm.a`).

Bien. Pero cuando su programa está compuesto por varios ficheros de código fuente, necesitará usar la utilidad `make`. Supongamos que ha escrito un evaluador de expresiones: su código fuente se denomina `parser.c` e tiene un `#include` de dos ficheros en su cabecera: `parser.h` y `xy.h`. Entonces, desea utilizar las rutinas de `parser.c` en otro programa, `calc.c`, que hace un `#include` de `parser.h`. ¡Vaya lío! ¿Cómo se puede compilar `calc.c`?

Debe escribir un fichero llamado `makefile`, el cual muestra al compilador las dependencias entre los ficheros de código fuente y los ficheros de código objeto. En nuestro ejemplo:

```
# Este es el makefile, utilizado para compilar calc.c
# ¡Pulse la tecla <TAB> en las posiciones marcadas!

calc: calc.o parser.o
<TAB>gcc -o calc calc.o parser.o -lm
# calc depende de dos ficheros de codigo objeto: calc.o and parser.o

calc.o: calc.c parser.h
<TAB>gcc -c calc.c
# calc.o depende de dos ficheros de codigo fuente

parser.o:  parser.c parser.h xy.h
<TAB>gcc -c parser.c
# parser.o depende de tres ficheros de codigo fuente

# fin de makefile.
```

Guarde esto como `makefile` y teclee

```
$ make
```

para compilar su programa; alternativamente, guárdelo como `calc.mak` y teclee

```
$ make -f calc.mak
```

Y, por supuesto, LPM.

Puede pedir ayuda acerca de las funciones de C, que se encuentra en las páginas `man`, sección 3; por ejemplo:

```
$ man 3 printf
```

Hay muchas librerías disponibles por ahí; entre las primeras que deseará usar están las `ncurses`, para manejar efectos en modo de texto, y `svgalib`, para hacer gráficos. Si se siente suficientemente valiente para atacar la programación de las X, consiga XForms (`bloch.phys.uwm.edu/pub/xforms`) y/o MGUI (`http://www.volftp.vol.it/IT/IT/ITALIANI/MORELLO/index.htm`), dos fantásticas librerías que hacen de la programación bajo X algo realmente fácil. Más aún, si no puede vivir sin un IDE al estilo Borland, consiga el paquete `xwpe` de `ftp://sunsite.unc.edu/pub/Linux/apps/editors/`. Posiblemente sea de su agrado.

8 El restante 1%

8.1 Gestión de Memoria Virtual

Aunque Linux puede ejecutarse en teoría con sólo 2 megas de RAM, cuanto más se tenga, más se puede hacer. El sistema X Window no se ejecutará a menos que tenga 8 megas. Para crear una memoria virtual de 8 megas adicionales, teclee como root:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=8192
# mkswap /swapfile 8192
# sync
# swapon /swapfile
```

Añada la última línea en `/etc/rc.d/rc.local` para hacer que el fichero de memoria virtual esté disponible la siguiente vez que arranque, o añada esta línea en `/etc/fstab`:

```
/swapfile   swap   swap   defaults
```

8.2 Utilización de `tar` y `gzip`

Bajo Unix hay algunas aplicaciones ampliamente utilizadas para archivar y comprimir ficheros. `tar` se utiliza para archivar varios ficheros en uno —es como PKZIP o ARJ, pero no comprime, sólo archiva. Para archivar varios ficheros en uno (que llamaremos archivo de ahora en adelante):

```
$ tar -cvf <nombre_fichero_final.tar> <fichero1> [fichero2...]
```

Para extraer ficheros de un archivo:

```
$ tar -xpvf <nombre_fichero.tar> [fichero]
```

Para listar los ficheros contenidos de un archivo:

```
$ tar -tf <nombre_fichero.tar> | less
```

Puede comprimir los ficheros usando `compress`, el cual es obsoleto y no debería ser utilizado nunca más, o usando `gzip`:

```
$ compress <fichero>
$ gzip <fichero>
```

eso crea un fichero comprimido con la extensión `.Z` (compress) o `.gz` (gzip). Estos programas sólo pueden comprimir un fichero cada vez. Para descomprimir, use:

```
$ compress -d <fichero.Z>
$ gzip -d <fichero.gz>
```

LPM.

Las utilidades `unarj`, `zip` y `unzip` (compatibles con ARJ y PK??ZIP) también están disponibles. Los ficheros con la extensión `.tar.gz` o `tgz` (archivados con `tar`, posteriormente comprimidos con `gzip`) son tan comunes en el mundo Unix como los ficheros `.ZIP` bajo DOS. Para listar los contenidos de un fichero `.tar.gz` utilice:

```
$ gzip -dc <fichero.tar.gz> | tar tf - | less
```

o también:

```
$ tar -cvzf <fichero.tar.gz>
```

8.3 Instalación de aplicaciones

Antes de nada: instalar paquetes es trabajo del root. Algunas aplicaciones Linux se distribuyen como ficheros `.tar.gz` o `.tgz`, preparadas específicamente para que sean descomprimidas desde el directorio raíz / escribiendo el siguiente comando:

```
# gzip -dc <fichero.tar.gz> | tar xvf -
```

Los ficheros se descomprimirán en el directorio adecuado, el cual será creado "al vuelo". Los usuarios de la distribución Slackware tienen un programa gestor de paquetes amigable para el usuario; otro programa es `rpm`, el cual está disponible para todas las distribuciones gracias a Red Hat.

Los demás paquetes no deberían ser instalados desde /; típicamente, el paquete contendrá un directorio llamado `nombrepaquete/` y un montón de ficheros y/o subdirectorios dentro de `nombrepaquete/`. Una buena regla es instalar todos esos paquetes bajo `/usr/local`.

Además, otros paquetes se distribuyen con sus fuentes en C o C++, los cuales han de ser compilados para crear los programas binarios. En la mayor parte de los casos, todo lo que debe hacer es ejecutar `make`. Obviamente, necesitará el compilador `gcc`.

8.4 Trucos imprescindibles

- **Terminación de comandos**: presionando `TAB` mientras teclea un comando completará la línea por usted. Ejemplo: tiene que teclear `gcc este_es_un_nombre_de_fichero_largo.c`; con teclear `gcc este TAB` bastará. (Si tiene otros ficheros que comienzan con los mismos caracteres, proporcione los caracteres suficientes para resolver cualquier ambigüedad.)
- **Visión de pantallas anteriores**: presionar `MAYÚS + RE PÁG` (la tecla gris) le permite volver atrás unas cuantas páginas, dependiendo de la memoria de vídeo que posea.
- **Resetear la pantalla**: si por un casual hace un `cat` o un `more` de un fichero binario, su pantalla puede acabar llena de basura. Para arreglar las cosas, teclee `reset` a ciegas o pulse esta secuencia de caracteres: `echo CTRL-V ESC c RETURN`.
- **Pegar texto**: en consola, ver más abajo; en las X, haga click y arrastre para seleccionar el texto en una ventana xterm, después haga click en el botón central (o con los dos a la vez si tiene un ratón de dos botones) para pegar. También hay un `xclipboard` (portapapeles de X) (de momento, sólo para texto); no se confunda por su muy baja velocidad de respuesta.
- **Usar el ratón**: instale `gpm`, el controlador de ratón para la consola. Haga click y arrastre para seleccionar texto, entonces haga un click con el botón derecho para pegar el texto seleccionado. Funciona a través de diferentes consolas virtuales.
- **Mensajes del kernel**: échele un vistazo a `/var/adm/messages` o `/var/log/messages` como root para ver lo que el kernel le dice, incluyendo los mensajes de inicio.

8.5 Programas y comandos útiles

Esta lista refleja mis preferencias y necesidades personales, por supuesto. En primer lugar, dónde encontrarlas. Ya que usted sabe cómo navegar por la red y cómo utilizar `archie` y `ftp`, sólo le daré tres de las más importantes direcciones para Linux: `ftp://sunsite.unc.edu/`, `ftp://tsx-11.mit.edu/`, y `ftp://nic.funet.fi/`. Por favor, use el mirror más cercano.

- `at` le permite ejecutar programas a una hora y fecha especificados.
- `awk` es un lenguaje simple pero potente de manipulación de ficheros de datos (entre otras cosas). Por ejemplo, siendo `datos.dat` su fichero de datos multicampo,

  ```
  $ awk '$2 ~ "abc" {print $1, "\t", $4}' datos.dat
  ```

  imprime los campos 1 y 4 de cada linea de `datos.dat` cuyo segundo campo contenga "`abc`".
- `delete-undelete` borran y recuperan ficheros;
- `df` da información acerca de los discos montados;
- `dosemu` permite ejecutar bastantes (no todos) programas DOS —incluyendo Windows 3.x— con un poco de trasteo;
- `file nombrefichero` le dice qué tipo de fichero es `nombrefichero` (texto ASCII, ejecutable, comprimido, etc.);
- `find` (ver también la sección 3) es uno de los comandos más potentes y útiles. Se utiliza para buscar ficheros que se ajusten a unas determinadas características, y realizar acciones sobre ellos. El uso general de `find` es:

  ```
  $ find <directorio> <expresion>
  ```

  donde `expresion` incluye criterios de búsqueda y acciones. Ejemplos:

  ```
  $ find . -type l -exec ls -l {} \;
  ```

  busca todos los ficheros que son enlaces simbólicos y dice a dónde apuntan.

  ```
  $ find / -name "*.old" -ok rm {} \;
  ```

  busca todos los ficheros que se ajusten a lo especificado y los borra, pidiéndole antes confirmación.

  ```
  $ find . -perm +111
  ```

  busca todos los ficheros cuyos permisos sean `111` (ejecutables para todos).

  ```
  $ find . -user root
  ```

  busca todos los ficheros que pertenecen al root. Hay muchas posibilidades: LPM.
- `gnuplot` es un brillante programa para dibujos científicos;
- `grep` busca cadenas de texto en ficheros. Por ejemplo:

  ```
  $ grep -l "geologia" *.tex
  ```

  lista todos los ficheros `*.tex` que contienen la palabra `geologia`. La variante `zgrep` trabaja en ficheros comprimidos con gzip. LPM;
- `gzexe` comprime binarios ejecutables manteniéndolos ejecutables (similar a PKLITE);

- `joe` es un excelente editor. Invocándolo tecleando `jstar` conseguirá los mismos caracteres de teclado que WordStar y sus descendientes, incluyendo DOS y los editores Turbo... de Borland;
- `less` es, probablemente, el mejor navegador de texto, y si está adecuadamente configurado, permite navegar por ficheros `zip`, `tar` o `gzip`.
- `lpr fichero` imprime un fichero en segundo plano. Para comprobar el estado de la cola de impresión, use `lpq`; para quitar un fichero de la cola de impresión, use `lprm`;
- `mc` es un maravilloso gestor de ficheros, clon del *comandante norton*;
- `pine` es un simpático programa gestor de correo electrónico;
- `script fichero_script` copia a `fichero_script` lo que aparece en pantalla antes de ejecutar el comando exit. Util para depuración;
- `sudo` permite a los usuarios ejecutar algunas funciones del root (p.e. formatear y montar discos; LPM);
- `uname -a` da información acerca del sistema;
- `zcat` y `zless` son útiles para ver ficheros de texto comprimidos con `gzip` sin descomprimirlos. Un posible uso es:

```
$ zless ficherotexto.gz
$ zcat ficherotexto.gz | lpr
```

- Los siguientes comandos son a menudo utilizados: `bc, cal, chsh, cmp, cut, fmt, head, hexdump, nl, passwd, printf, sort, split, strings, tac, tail, tee, touch, uniq, w, wall, wc, whereis, write, xargs, znew`. LPM.

8.6 Extensiones comunes y programas relacionados

Se podrá encontrar con una gran variedad de extensiones de ficheros. Excluyendo los más exóticos (como los de fuentes, etc.), aquí tenemos una lista de los más usuales:

- `1 ... 8`: páginas de manual. En el extraño caso en que aún no lo tenga, consiga `man`.
- `arj`: archivo hecho con `arj`. Use `unarj` para descomprimirlo.
- `dvi`: fichero de salida producido por TeX (ver más abajo). Use `xdvi` para visualizarlo; Use `dvips` para transformarlo en un fichero PostScript (`.ps`).
- `gif`: fichero gráfico. Consiga `seejpeg`, `xpaint` o `zgv` para visualizarlo.
- `gz`: archivo comprimido con `gzip`.
- `info`: archivo *info* (Algo así como una alternativa a las páginas de manual). Consiga `info`.
- `jpg, jpeg`: fichero gráfico. Consiga `seejpeg` o `zgv`.
- `lsm`: Fichero *Linux Software Map*. Es un fichero de texto ASCII plano que contiene la descripción de un paquete.
- `ps`: Fichero PostScript. Para visualizarlo o imprimirlo consiga `gs` y, opcionalmente, `ghostview`.
- `rpm`: Paquete de Red Hat. Puede instalarlo en cualquier sistema utilizando el gestor de paquetes `rpm`.
- `taz, tar.Z`: archivo hecho con `tar` y posteriormente comprimido con `compress`.
- `tgz, tar.gz`: archivo hecho con `tar` y posteriormente comprimido con `gzip`.

- `tex`: fichero de texto para utilizar con TeX, un poderoso formateador de textos. Consiga el paquete `tex`, disponible en muchas distribuciones; pero tenga cuidado con NTeX, el cual tenía fuentes corruptas y estaba incluido en algunas versiones de la distribución Slackware.
- `texi`: fichero texinfo, a partir del cual se pueden producir tanto archivos TeX como info. Consiga `texinfo`.
- `xbm, xpm, xwd`: fichero gráfico. Consiga `xpaint`.
- `Z`: archivo hecho con `compress`.
- `zip`: archivo hecho con `zip`. Consiga `zip` y `unzip`.

9 Fin, por ahora

¡Felicidades! Se ha iniciado un poco en el Unix y está preparado para comenzar a trabajar. Recuerde que su conocimiento del sistema es aún limitado, y que se espera que practique más con Linux para usarlo cómodamente. Pero si todo lo que quería hacer era conseguir un puñado de aplicaciones y empezar a trabajar con ellas, apuesto a que lo que se incluyó aquí fue suficiente.

Estoy seguro de que habrá disfrutado de usar Linux y seguirá aprendiendo más acerca de él —todo el mundo lo hace—. Apuesto, también, que nunca volverá al DOS. Espero haberme hecho entender y haber realizado un buen servicio a mis 3 o 4 lectores.

9.1 Copyright

10 Agradecimientos

”DOS-to-Linux-HOWTO” fue escrito por Guido Gonzato, `guido@ibogfs.df.unibo.it`. Muchas gracias a Matt Welsh, el autor de ”*Linux: Instalación y Primeros Pasos*”, a Ian Jackson, el autor de ”*Linux frequently asked questions with answers*”, a Giuseppe Zanetti, el autor de ”Linux”, a todos los amigos que me enviaron sugerencias, y especialmente a Linus Torvalds y GNU que nos trajeron Linux.

Este documento se distribuye ”tal cual”. He puesto un gran esfuerzo en escribirlo tan correctamente como he podido. Pese a ello, la información contenida en el mismo debe ser utilizada bajo su propia responsabilidad. En ningún caso el autor será responsable de cualquier daño resultante del uso de este documento.

El correo es bienvenido. Para cualquier duda, sugerencia, crítica, etc., siéntase libre de contactar conmigo.

Disfrute de Linux y de la vida,

Guido `=8-)`

10.1 Traducción

"*CÓMO pasar de DOS a LINUX*" fue traducido por David Martín Carreño, `davefx@bigfoot.com`, como un pequeño grano de arena más dentro del Proyecto INSFLUG. Para más información, vea la sección 11.

He intentado una traducción fidedigna del documento original de Guido, aunque en algunos lugares haya actualizado información o cambiado algunas expresiones por otras más adecuadas a la jerga en nuestro idioma.

Si desea plantear alguna duda, sugerencia o crítica, pues tampoco dude en contactar conmigo.

David!! 3B-)

11 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos (Comos), así como la producción de documentos originales en aquellos casos en los que no existe análogo en inglés.

En el **INSFLUG** se orienta preferentemente a la traducción de documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del INSFLUG para más información al respecto.

En la sede del INSFLUG encontrará siempre las **últimas** versiones de las traducciones: `www.insflug.org`. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

Francisco José Montilla, `pacopepe@insflug.org`.