¿Qué es un shell? y ¿Qué es Bash?

Un shell es interfaz de usuario que sirve para acceder a los servicios que el sistema operativo brinda. Por servicios me refiero cosas como crear un archivo o directorio, eliminar un archivo, guardar información en un archivo, ejecutar un programa, leer los contenidos de un DVD, básicamente cualquier cosa que hacemos día a día en una computadora.

En general hay dos tipos de shells:

- **Shells Gráficos**: Estos shells ya los conocen, son los que les permiten abrir ventanas, cerrarlas, seleccionarlas, etc. En Windows el shell gráfico es el explorador de Windows (explorer.exe). En Linux hay muchos shells, uno de los más conocidos es Unity y es el que usa la distribución Ubuntu.
- **Shells CLI (command-line interface)**: Estos son los que vamos a estudiar. A pesar de ser mucho más básicos que los shells gráficos nos van a permitir (luego de mucha práctica) hacer cosas muy interesantes.

Si bien hay muchos shells CLI tanto para Linux como para Windows, vamos a estudiar uno en particular llamado Bash. Bash es un shell para Linux que viene instalado por defecto en la mayoría de las distribuciones de Linux.

Los shells CLI sólo pueden mostrar texto, no tienen imágenes ni videos ni ventanas. Dado que lo único que tenemos es texto, la única forma de indicarle qué hacer a la computadora es a través de texto y la única forma que tiene la computadora de respondernos es a través de texto.

Por ejemplo, si queremos pedirle que nos muestre la hora y la fecha escribimos `date`, presionamos enter y la computadora responde:

```
sáb ago 12 00:37:15 ART 2017
```

Toda la interacción con un shell CLI funciona de esta manera. Se ingresa texto y el shell CLI responde con texto.

¿Cómo instalo Bash?

Como ya vimos, Bash ya viene instalado en casi cualquier distribución de Linux, por lo que no será necesario instalar nada.

¿Cómo abrimos Bash?

Parar abrir Bash vamos a usar una terminal. Esta terminal simula un shell CLI desde un shell gráfico. Esto es mucho más cómodo que no tener una shell gráfica y sólo tener texto!

Entonces, abramos una terminal! Si están en Ubuntu vayan al lanzador de aplicaciones y busquen una aplicación llamada Terminal.

Una vez abierta la terminal van a ver algo asi:

```
fran@fran-laptop:~$
```

Bueno, esto se llama prompt, les indica qué usuario está usando la máquina (`fran`), qué computadora estoy usando (`fran-laptop`) y en qué directorio estoy (`~` este es el directorio donde están sus archivos personales en linux, `~` es lo mismo que `/home/fran` ; no se preocupen si no entienden esto, con la práctica van a entender).

El directorio actual

Cuando usen un shell CLI, es importante saber en qué directorio se encuentran. Por ello es que el prompt siempre lo indica. El directorio actual en el prompt es lo que se encuentra después de los `:` y antes de `$`. Por ejemplo:

- `fran@fran-laptop:~/Escritorio$` indica que estoy en el Escritorio
- `fran@fran-laptop:~/Descargas$` indica que estoy en Descargas
- `fran@fran-laptop:~/Documentos/Viejo` indica que estoy en el directorio Viejo, adentro del directorio Documentos.

El comando `ls`

El comando `ls` significa *list* (listar) y lista los archivos y directorios que existen en el directorio actual. Para probarlo, escriban `ls` y presionen enter. En mi caso el shell responde:

```
fran@fran-laptop:~$ ls
Code  Escritorio  Descargas  EOxposure  Musica  OS  Otro  Imágenes
```

El comando `cd`

El comando `cd` significa *change directory* (cambiar directorio) y bueno... nos permite movernos de directorio en directorio.

Para probarlo, escriban `cd Desktop`, `cd Escritorio`, `cd Descargas` o lo que prefieran! Noten el cambio en el prompt. Por ejemplo:

```
fran@fran-laptop:~$ cd Documentos
fran@fran-laptop:~/Documentos$ cd Viejo
fran@fran-laptop:~/Documentos/Viejo$
```

Para volver atrás (al directorio anterior) pueden usar `cd ..`:

```
fran@fran-laptop:~/Documentos/Viejo$ cd ..
fran@fran-laptop:~/Documentos$ cd ..
fran@fran-laptop:~$
```

Para volver al inicio (a `~`) desde cualquier directorio pueden usar `cd ~`:

```
fran@fran-laptop:~/Documentos/Viejo$ cd ~
fran@fran-laptop:~
```

Creando archivos y directorios: `touch` y `mkdir`

Para crear un archivo vacío llamado, por ejemplo, `ejemplo.py` basta hacer:

`fran@fran-laptop:~/$ touch ejemplo.py`

Como ejemplo prueben hacer esto:

```
fran@fran-laptop:~$ ls
Code  Escritorio  Descargas  EOxposure  Musica  OS  Otro  Imágenes
fran@fran-laptop:~$ touch ejemplo.py
fran@fran-laptop:~$ ls
```

```
Code  Escritorio  Descargas  EOxposure  Musica  OS  Otro  Imágenes ejemplo.py
fran@fran-laptop:~$
```

Noten que un nuevo archivo ha sido creado con el nombre que le indicamos.

`touch` permite crear más de un archivo. Por ejemplo:

`fran@fran-laptop:~$ touch ejemplo1.py ejemplo2.py ejemplo3.py`

Crea tres archivos llamados ejemplo1.py, ejemplo2.py y ejemplo3.py.

Como ejercicio prueben crear directorios con el comando `mkdir` que funciona igual a `touch`. Verifiquen que funcionó usando `ls`.

Eliminando archivos y directorios: `rm` y `rm -r`

Para eliminar un archivo usar:

```
fran@fran-laptop:~$ touch ejemplo1.py ejemplo2.py ejemplo3.py
fran@fran-laptop:~$ rm ejemplo1.py ejemplo2.py ejemplo3.py
```

Para eliminar un directorio usar:

```
fran@fran-laptop:~$ mkdir Viejo
fran@fran-laptop:~$ rm -r Viejo
```

Tengan cuidado que **estos comandos no son reversibles** (no hay papelera de reciclaje).

Renombrando y moviendo archivos y directorios: `mv`

Este comando puede mover archivos y directorios de un directorio al otro como cambiar el nombre de los mismos.

Supongamos que queremos cambiar el nombre del directorio (o archivo) `Viejo` a `Nuevo`. Para ello:

```
fran@fran-laptop:~$ mkdir Viejo
fran@fran-laptop:~$ mv Viejo Nuevo
```

Supongamos que queremos mover el archivo `ejemplo.py` al directorio `Viejo`. Para ello:

```
fran@fran-laptop:~$ touch ejemplo.py
fran@fran-laptop:~$ mkdir Viejo
fran@fran-laptop:~$ mv ejemplo.py Viejo
```

Como `Viejo` ya existe, `ejemplo.py` es movido a `Viejo` en vez de ser renombrado a `Viejo`. De esta forma el comando `mv` distingue entre mover y cambiar de nombre.

La estructura de los comandos

Como habrán notado los comandos tienen una estructura que se repite:

- Primero viene el nombre del comando (por ejemplo, `ls`, `cd`, `mv`, `touch`)

- Opciones (por ejemplo, `-r` como en el caso de `rm -r`')
- Archivos o directorios (por ejemplo `Viejo` o `Nuevo`)

Las opciones vienen en varios formatos:

- Guión + letra(s) (cada letra siendo una opción). Por ejemplo `-rasf`, `-ra -sf` o `-r -a -s -f` (son todas equivalentes)
- Dos guiones + palabras. Por ejemplo `--verbose` ó `--verbose-mode`.

Las opciones modifican el comportamiento de un comando. De esta forma un comando puede hacer muchas cosas distintas según las opciones que toma. Como ya vimos `rm -r` puede borrar directorios mientras que `rm` sin esa opción no puede.

Además de tomar archivos o directorios los comandos también pueden tomar texto que le pasemos como es el caso del comando `echo` que imprime en pantalla lo que le decimos:

```
fran@fran-laptop:~$ echo hola compu!
hola compu!
```

Ayuda!!

Hay comandos que explican lo que hacen otros comandos. Lo malo es que son un poco difíciles de entender. Tienen un lenguaje muy técnico y no suelen dar ejemplos. El más importante de estos comandos de ayuda es `man` (de manual)

Prueben los siguientes comandos:

```
fran@fran-laptop:~$ man ls
fran@fran-laptop:~$ man mv
fran@fran-laptop:~$ man rm
fran@fran-laptop:~$ man mkdir
fran@fran-laptop:~$ man touch
```

Podrán notar que `man` explica con mucho detalle qué hace cada opción. Esto es fundamental para entender qué hace un comando. Estos manuales son muy útiles y es importante saber usarlos.

En resumen

Comando	Función
ls	Listar los archivos y directorios del directorio actual
cd nombreDelDirectorio	Cambiar de directorio
touch nombreDelArchivo	Crear un nuevo archivo
mkdir nombreDelDirectorio	Crear un nuevo directorio
rm nombreDelArchivo	Eliminar un archivo
rm -r nombreDelDirectorio	Eliminar un directorio

man comando	Ayuda sobre un comando
mv archivoODirectorioQueExiste directorioQueExiste	Mover el archivo o directorio a otro directorio
mv archivoODirectorioQueExiste nombre	Renombrar el archivo

En el último caso `nombre` no puede ser un directorio ya existente, si ya existe en vez de renombrarlo se lo mueve.

Otros programas

Desde Bash podemos ejecutar cualquier programa:

```
fran@fran-laptop:~$ python
fran@fran-laptop:~$ firefox
fran@fran-laptop:~$ google-chrome
fran@fran-laptop:~$ libreoffice
```

Notar que puede ser que algunos de estos programas no funcionen si no los tienen instalados. En estos casos Bash muestra el error correspondiente.

Ejemplos más avanzados y curiosidades

Hasta ahora vimos comandos bastante sencillos, pero lo bueno de Bash es que permite expresar operaciones muy complejas. A continuación se muestran algunos ejemplos del tipo de cosas que se pueden hacer con Bash. Quiero aclarar que no pretendo que entiendan cómo funcionan estos comandos, simplemente que sepan el tipo de cosas que Bash permite hacer.

Con este comando podemos bajar todo un sitio entero:

```
wget --random-wait -r -p -e robots=off -U mozilla
http://blog.computationalcomplexity.org/
```

Con este comando podemos hacer tweets cada 1 hora (o cualquier otro intérvalo):

```
while true; do curl -u user:pass -d status="Tweeting from the shell"
http://twitter.com/statuses/update.xml | at midnight; sleep 3600; done;
```

(Tienen que cambiar `user` por su nombre de usuario y `pass` por su contraseña)

Con este comando pueden renombrar todos los archivos que terminen en .py por .pa:

```
for file in *.py; do mv "$file" "`basename "$file" .py`.pa"; done
```

Con este comando podemos averiguar nuestra ip:

```
curl -s http://whatismyip.org/ | grep -oP '(\d{1,3}\.){3}\d+'
```

Para ver más comandos interesantes:

Commandlinefu

Una cosa muy buena de los shells CLI como Bash es que podemos googlear nuestra duda y casi siempre encontramos una respuesta que podemos copiar y pegar en la terminal que resuelve nuestro problema. Y copiar y pegar en vez de seguir una serie de pasos tediosos (como es el caso de los shells gráficos) es mucho más simple.

Otro tutorial

Este video y este video en español explican prácticamente lo mismo que expliqué acá. Mirenlos aunque hayan leido todo!

Otros tutorial más completos

Este tutorial en español me parece bueno. Si tienen dudas siempre pueden buscar videos en youtube o googlear!

Consejos

- No intenten aprender todo de golpe: aprender a manejar un shell CLI lleva su tiempo.
- Practiquen! Es muy importante practicar! En vez de mover un archivo (o crear un archivo o cambiar el nombre de un directorio o apagar la computadora o lo que sea) con el shell gráfico háganlo con la terminal. Si lo hacen muchas veces y se acostumbran a hacer tareas cotidianas con la terminal en vez de con programas gráficos van a mejorar muchísimo.
- Googleeen! googlear cualquier duda que tengan ("cómo apago la computadora CLI linux", "cómo editar un archivo desde la terminal linux") les va a ayudar a mejorar mucho. Un cuarto o más de las cosas que uno escribe en la terminal son el resultado de copiar-y-pegar de Google. Les recomiendo agregar las palabras "terminal" "linux" y "CLI" a las búsquedas para encontrar buenos resultados.

Ejercicios

Los ejercicios marcados con (*) son un poco más difíciles.

1. Aprendan a usar los comandos `cp` (copia archivos), `cp -r` (copia directorios) `cat` (imprime los contenidos de un archivo en pantalla).
2. ¿Qué hacen los siguientes comandos? (ejecutarlos sin borrar o modificar `archivo.txt` entre un comando y el otro)
    - `echo hola como va >> archivo.txt` (hagan `cat archivo.txt` para ver los contenidos)
    - `ls >> archivo.txt` (hagan `cat archivo.txt` para ver los contenidos)
3. Ahora prueben los mismos comandos sustituyendo `>>` por `>`. ¿Qué cambió?
4. `sudo` es un comando que les permite ejecutar comandos que sólo el super-usuario (similar al Administrador en Windows) puede ejecutar. `apt-get install -y` es uno de estos comandos. `apt-get install -y` es comando que sirve para instalar programas. Aprendan a instalar programas usando `sudo apt-get install -y`. Por ejemplo, prueben hacer: `sudo apt-get install -y git`.
5. ¿Para qué sirve `apt-get update`? (usen `man` y Google!)
6. ¿Para qué sirve `apt-get upgrade`?
7. ¿Para qué sirve `apt-get purge`?
8. Busquen al menos 3 programas usando `apt-cache search`.