La terminal de UNIX

La Shell (o terminal) es un interprete de comandos. Es simplemente un modo alternativo de controlar un ordenador basado en una interfaz de texto. La terminal nos permite ejecutar software escribiendo el nombre del programa que queremos ejecutar en la terminal. Podemos pedirle al ordenador que ejecute un programa mediante el ratón ciclando en distintos lugares del escritorio o podemos escribir una orden para conseguir el mismo objetivo. Por ejemplo, para pedirle al ordenador que nos de una lista de los archivos presentes en un directorio podemos abrir un navegador de archivos o podemos escribir en la terminal:

$ ls folder_name

file_1.txt

file_2.txt

Ninguna de las dos formas de comunicarse con el ordenador es mejor que la otra aunque en ciertas ocasiones puede resultar más conveniente utilizar una u otra Las ventajas de la línea de comandos son:

- •Necesidad. Existe mucho software que está sólo disponible en la terminal. Esto es especialmente cierto en el área de la bioinformática.
- •Flexibilidad. Los programas gráficos suelen ser muy adecuados para realizar la tarea para la que han sido creados, pero son difíciles de adaptar para otras tareas. Los programas diseñados para ser usados en la línea de comandos suelen ser muy versátiles.
- •Reproducibilidad. Documentar y repetir el proceso seguido para realizar un análisis con un programa gráfico es muy costoso puesto que es difícil describir la secuencia de clicks y doble clicks que hemos realizado. Por el contrario, los procesos realizados mediante la línea de comandos son muy fáciles de documentar puesto que tan sólo debemos guardar el texto que hemos introducido en la pantalla.
- •Fiabilidad. Los programas básicos de Unix fueron creados en los años 70 y han sido probados por innumerables usuarios por lo que se han convertido en piezas de código extraordinariamente confiables.
- •Recursos. Las interfaces gráficas suelen consumir muchos recursos mientras que los programas que funcionan en línea de comandos suelen ser extraordinariamente livianos y rápidos. Este poco uso de recursos facilita, por ejemplo, que se utilice a través de la red.

El problema de la terminal es que para poder utilizarla debemos saber previamente qué queremos hacer y cómo. Es habitual descubrir como funciona un programa con una interfaz gráfica sin tener que leer un manual, esto no sucede en la terminal.

Para usar la línea de comandos hay que abrir una terminal. Se abrirá una terminal con un mensaje similar a:

usuario $

Este pequeño mensaje se denomina prompt y el cursor parpadeante que aparece junto al él indica que el ordenador está esperando una orden. El mensaje exacto que aparece en el prompt puede variar ligeramente, pero en Ubuntu suele ser similar a:

usuario@ordenador:~/documentos$

En el prompt de Ubuntu se nos muestra el nombre del usuario, el nombre del ordenador y el directorio en el que nos encontramos actualmente, es decir, el directorio de trabajo actual.

Cuando el prompt se muestra podemos ejecutar cualquier cosa, por ejemplo le podemos pedir que liste los ficheros mediante el comando ls (LiSt)::

usuario $ ls

lista_libros.txt

rectas_cocina/

ls, como cualquier otro comando, es en realidad un programa que el ordenador ejecuta. Cuando escribimos la orden (y pulsamos enter) el programa se ejecuta. Mientras el programa está ejecutándose el prompt desaparece y no podemos ejecutar ningún otro comando. Pasado el tiempo el programa termina su ejecución y el prompt vuelve a aparecer. En el caso del comando ls el tiempo de ejecución es tan pequeño que suele ser imperceptible.

Los programas suelen tener unas entradas y unas salidas. Dependiendo del caso estas pueden ser ficheros o caracteres introducidos o impresos en la pantalla. Por ejemplo, el resultado de ls es simplemente una lista impresa de ficheros y directorios en la interfaz de comandos.

Normalmente el comportamiento de los programas puede ser modificado pasándoles parámetros. Por ejemplo, podríamos pedirle al programa ls que nos imprima una lista de ficheros más detallada escribiendo:

$ ls -l

Ayuda

Cada comando tiene unos parámetros y opciones distintos. La forma estándar de pedirles que nos enseñen cuales son estos parámetros suele ser utilizar las opciones ‘–help’, ‘-h’ o ‘-help’, aunque esto puede variar en comandos no estándar.

$ ls --help

Modo de empleo: ls [OPCIÓN]... [FICHERO]...

List information about the FILEs (the current directory by default).

Sort entries alphabetically if none of -cftuvSUX nor --sort.

Otro modo de acceder a una documentación más detallada es acceder al manual del programa utilizando el comando man (MANual):

$ man ls

(para terminar pulsar "q")

man es un programa interactivo, cuando ejecutamos el comando el programa se abre y el prompt desaparece. Man es en realidad un visor de ficheros de texto por lo que cuando lo ejecutamos la pantalla se rellena con la ayuda del programa que hemos solicitado. Podemos ir hacia abajo o hacia arriba y podemos buscar en el contenido de la ayuda. El prompt y la posibilidad de ejecutar otro programa no volverán a aparecer hasta que no cerremos el programa interactivo. En el caso de man para cerrar el programa hay que pulsar la tecla “q”.

Completado automático e historia

El intérprete de comandos dispone de algunas utilidades para facilitarnos su uso. Una de las más utilizadas es el completado automático. Podemos evitarnos escribir una gran parte de los comandos haciendo uso de la tecla tabulador. Si empezamos a escribir un comando y pulsamos la tecla tabulador el sistema completará el comando por nosotros. Para probarlo creemos los ficheros datos_1.txt, datos_2.txt y tesis.txt::

~$ touch datos_1.txt

~$ touch datos_2.txt

~$ touch experimento.txt

Si ahora empezamos a escribir cp e y pulsamos el tabulador dos veces, el intérprete de comandos completará el comando automáticamente::

~$ cp e

~$ cp experimento.txt

Si el intérprete encuentra varias alternativas completará el comando hasta el punto en el que no haya ambigüedad. Si deseamos que imprima una lista de todas las alternativas disponibles para continuar con el comando deberemos pulsar el tabulador dos veces.

~$ cp d

$ cp datos_

datos_1.txt datos_2.txt

~$ cp datos_

Otra de las funcionalidades que más nos pueden ayudar es la historia. El intérprete recuerda todos los comandos que hemos introducido anteriormente. Si queremos podemos obtener una lista de todo lo que hemos ejecutado utilizando el comando history. Pero lo más socorrido es simplemente utilizar los cursores arriba y abajo para revisar los comandos anteriores. Otra forma de acceder a la historia es utilizar la combinación de teclas control y r. De este modo podemos buscar comandos antiguos sencillamente.

Algunos comandos para python

Un programa de Python puede conocer la ruta desde donde está siendo ejecutado vía os.getcwd(), que no es otra cosa que lo que acabamos de explicar.

```
1.import os

2.print(os.getcwd())
```

Cuando a alguna función que trabaje con archivos le indicamos una ruta relativa, es en relación con este valor a partir del cual se resuelve la ruta final. Por ejemplo, el siguiente código:

```
1.f = open("archivo.txt")
```

Es siempre equivalente a:

```
1.import os

2.import os.path

3.f = open(os.path.join(os.getcwd(), "archivo.txt"))
```

Ejecutando aplicaciones

Dijimos que desde la terminal podemos ejecutar otras aplicaciones de consola. Por ejemplo, la consola interactiva de Python vía el siguiente comando:

- Windows
- Linux

```
1.C:\>C:\Python27\python.exe
```

Puesto que el acceso a esta aplicación es bastante frecuente, podemos acceder a ella aun prescindiendo de su ruta compelta:

- Windows
- Linux

```
1.C:\>python
```

Esto es posible por cuanto la ruta de Python está configurada en la variable de entorno PATH, lo cual ocurre automáticamente durante la instalación. Si por alguna razón no está configurada, podemos hacerlo manualmente como explicamos en este artículo.

El comando where (Windows) o which (Linux) nos indica la ruta completa de alguno de estos «atajos»:

- Windows
- Linux

```
1.C:\>where python

2.C:\Python27\python.exe
```

Otra aplicación de estas características es pip, la herramienta para instalar paquetes de terceros de Python, que por lo general se instala junto con él.

- Windows
- Linux

```
1.C:\>where pip
```

```
2.C:\Python27\Scripts\pip.exe
```

Cuando tenemos múltiples versiones de Python instaladas, solo una de ellas estará asociada con los comandos python y pip. Para acceder a las otras, deberemos indicar su ruta completa.

Ahora bien, al llamar a una aplicación desde la terminal podemos pasarle argumentos o parámetros. Cuando al programa python le pasamos como argumento el nombre de un script, en lugar de abrir la consola interactiva ejecuta dicho archivo.

```
1.python script.py
```

O bien usando una ruta absoluta:

- Windows
- Linux

```
1.C:\>python C:\Programas\script.py
```

Cuando un argumento contiene espacios, a menudo es necesario indicarlo entre comillas.

- Windows
- Linux

```
1.C:\>python "C:\Programas\mi script.py"
```

Desde Python podemos acceder a los argumentos pasados a través de la consola, vía la lista sys.argv.

```
1.import sys
```

```
2.print(sys.argv)
```