Introducción a la Terminal y Línea de Comandos

Qué es una terminal

La terminal es un intermediario entre el computador y el usuario que recibe nuestras órdenes y las traduce en lenguaje que la máquina pueda entender.

La sintaxis de la ejecución de un programa en una terminal:

$comando -flag1 -flag2 argumento1 argumento2	
comando	Variable de entorno que ejecuta un programa
flag	Modificador del comando que se ejecuta
argumento	Entrada que se le da al programa.

```
$tail -n 10 file.txt
```

Comandos básicos para la manipulación de archivos

ls

Podemos listar los archivos de un directorio utilizando $ls:

$ls -flag	
a	Muestra todos los archivos. Incluidos los ocultos
t	Ordena por fecha de modificación
x	Ordena por nombre y después extensión
X	Ordena por extensión y luego por nombre
l	Muestra todos los datos de un archivo: Usuario, grupo, permisos, tamaño, fecha y hora.
R	Muestra el contenido de todos los subdirectorios de forma recursiva
S	Ordena los resultados por tamaño de archivo

```
$ls -a
```

Atajos a directorios

Podemos hacer uso de los archivos virtuales “..” y “.” para referirnos al directorio anterior y al actual respectivamente.

```
$cp /path/to/dir/file.txt .
```

El comando copiará el archivo file.txt que se encuentra en “/path/to/dir/” a nuestro directorio actual (.)

pwd

Podemos hacer uso de pwd para que la consola nos imprima el path de la ubicación en la que nos encontramos:

```
$pwd
```

Podría tener el siguiente output:

```
/Users/gollum/Desktop
```

cd

Podemos movernos entre directorios del sistema de archivos haciendo uso de cd:

$cd path	
path	El programa se situaría en la posición indicada en path.

```
$cd /desktop/website
```

Nos situaría en la carpeta “website” que se encuentra en nuestro escritorio.

Podemos hacer uso de los atajos “~” y “-” para ir al home o al último directorio visitado respectivamente:

```
$cd ~
```

```
$cd -
```

mkdir

Podemos hacer uso del comando mkdir para crear directorios.

$mkdir dir	
dir	El programa creará el directorio especificado en dir.

```
$mkdir fotos
```

Creará el directorio “fotos” en la carpeta en la que se encuentre la consola.

cp

Podemos hacer uso del comando cp para copiar archivos.

$cp file dir	
file	Refiere al archivo que se copiará
dir	Dirección destino en la que se copiará el archivo

```
$cp index.html /Desktop/website
```

Copiará el archivo index.html en la carpeta “website” del escritorio.

rm

Podemos hacer uso del comando rm para borrar archivos.

$rm file	
file	Refiere al archivo que se borrará

```
$rm index.html
```

Borrará el archivo index.html

mv

Podemos hacer uso del comando mv para mover archivos.

$mv file dir	
file	Refiere al archivo que se moverá
dir	Dirección destino en la que se moverá el archivo

```
$mv index.html /Desktop/website
```

Moverá el archivo index.html a la carpeta “website” del escritorio.

rmdir

Podemos hacer uso del comando rmdir para eliminar un directorio. Únicamente permite eliminar directorios vacíos.

$rmdir dir	
dir	Directorio que se eliminará

```
$rmdir /desktop/website/
```

Eliminará la carpeta “website” dentro del escritorio.

touch

Podemos crear un archivo haciendo uso del comando touch

$touch file	
file	Archivo que se creará

```
$touch /desktop/website/index.html
```

Creará el archivo “index.html” dentro de la carpeta “website2.

cat

Podemos ver el contenido de un archivo con cat

$cat file	
file	Archivo del que se verá el contenido

```
$cat index.html
```

Nos mostrará el contenido de index.html

head

Podemos ver las primeras líneas de un documento con head

$head -n x file	
n	Permite especificar una cantidad de líneas
x	Cantidad de líneas que se quieren ver (de arriba hacia abajo)
file	Archivo del que se verán las primeras n líneas

```
$head -n 20 index.html
```

Nos mostrará las primeras 20 líneas del archivo index.html

tail

Podemos ver las últimas líneas de un documento con tail

$tail -n x file	
n	Permite especificar una cantidad de líneas
x	Cantidad de líneas que se quieren ver (de abajo hacia arriba)
file	Archivo del que se verán las últimas n líneas

```
$tail -n 20 index.html
```

Nos mostrará las últimas 20 líneas del archivo index.html

grep

Podemos ver las líneas de un archivo en las que existe una expresión con grep

$grep flag expression file	
i	Realiza la búsqueda sin tomar en cuenta mayúsculas o minúsculas.
expression	Expresión de búsqueda.
file	Archivo en el que se buscará.

```
$greg -i “p>”$ index.html
```

La consola nos devolverá todas las líneas del archivo “index.html” que terminen en “p>”.

awk

Podemos delimitar partes de un documento con awk

$awk -F ‘delimitador’ ‘{ command }’	
F	Permite ingresar un delimitador
delimitador	Delimitador que separa los valores (coma, tab)
file	Archivo que se filtrará.
comando	Filtro que se aplicrá.

```
$awk -F ‘,’  ‘{ echo $1 }’ houses.csv
```

Devolverá la primera columna del archivo delimitado por comas “houses.csv”.

Edición de documentos de texto plano

vim

$vim file	
file	Archivo que se editará con vim (si no existe, se creará)

```
$vim index.html
```

Para comenzar con la edición debe oprimirse la letra “**i**”.

Para salir del modo edición, “**esc**”.

Para activar el modo comandos “**:**”.

Para guardar: “**W**”, Para salir: “**Q**”, Para guardar y salir: “**X**”.

Procesamiento de outputs

Como estándar, existen 3 canales de datos (stream) para un programa:

- Standard input
- Standard output
- Standard error

Por lo general, el canal de output y el de error se confunden como uno mismo, pero son vías diferentes que siempre deberán tratarse como tal.

Cargar argumentos externos

Podemos cargar los argumentos de un comando desde un archivo externo con <

$command < file	
file	Archivo que cuenta con los argumentos necesarios

Supongamos la existencia de un archivo “files.txt” que tiene en cada línea el nombre de diferentes directorios que queremos crear.
En vez de crearlos uno por uno con “mkdir” podemos hacer lo siguiente:

```
$mkdir < files.txt
```

Y se crearán todos los directorios que se encuentren listados en files.txt

Exportar output de un programa

Podemos usar > para guardar la salida de un resultado en un archivo.

$command > file	
file	Archivo en el que se guardará el output de command

```
$ls > log-file.txt
```

El archivo “log-file.txt” guardará el output que brinde el comando ls

Podemos usar >> para guardar la salida de un resultado al final un archivo sin sobreescribir el contenido existente.

$command >> file	
file	Archivo en el que se guardará el output de command

```
$ls >> log-file.txt
```

Se guardará el output que brinde el comando ls al final del archivo “log-file.txt” sin sobrescribir el contenido existente.

Pipes

El pipe (|) Nos permite ejecutar comandos con el standard output de otro comando. Ejemplo:

$history | grep “rm”
history generaría el historial de todos los comandos ejecutados en el terminal y grep “rm” realizaría una búsqueda sobre esos comandos para determinar cuáles han sido para borrar archivos (rm borra archivos). Por tanto, el output de history | grep “rm” nos devolvería directamente los archivos que se han borrado recientemente a través de la terminal.

Gestión de permisos de archivos

chmod

$chmod permissions file	
permissions	Permisos que se agregan a file • g(-/+)p: Determina el grupo de usuarios que se selecciona y el permiso que se quiere añadir. Por ejemplo: o-w determina el grupo “others” y quita el permiso de escritura (w). • (+-/)p: Determina la adición o eliminación de un permiso para todos los grupos, por ejemplo: +x da permiso de ejecución a todos los grupos. • 000: Determina los permisos en base al significado binario de los números. (Ver permisos binarios de la clase de fundamentos de ingeniería). Por ejemplo: 777.
file	Archivo en el que se guardará el output de command

```
$chmod 777 hola.sh
```

chmod le brindaría permisos de lectura, escritura y ejecución sobre el archivo “hola.sh” a todos los grupos.

Gestión de procesos

&

Se puede ejecutar un comando directamente en segundo plano con &

$comando &

```
$hola.sh &
```

En el ejemplo anterior, “hola.sh” se ejecutará en segundo plano y en la consola se podrá ejecutar un segundo comando mientras tanto. Más adelante, la terminal dará un aviso cuando “hola.sh” haya terminado su ejecución.

Ctrl + z

Un comando que ya ha sido ejecutado en primer plano puede pausarse haciendo uso de ctrl + z, al igual que puede recuperarse su ejecución con el comando fg.

ps ax

El comando ps ax permite ver un listado de todos los procesos que se están ejecutando en el sistema.

top

El comando top permite ver interactivamente todos los programas en ejecución.

kill

El comando kill permite cerrar forzosamente un programa.

$kill -priority process_number	
priority	Prioridad con la que se desea terminar un proceso (siendo la máxima prioridad 9)
process_number	Identificativo del proceso que se quiere terminar. El ID del proceso se obtiene con ps ax

```
$kill -9 23450
```

En el ejemplo anterior, kill terminará con el proceso 23450 con máxima prioridad.

killall

El comando killall permite cerrar forzosamente un programa.

$kill -priority program_name	
priority	Prioridad con la que se desea terminar un proceso (siendo la máxima prioridad 9)
program_name	Nombre del proceso que se quiere eliminar.

```
$kill -9 ‘php colgado.php’.
```

En el ejemplo anterior, killall terminará con el proceso ‘php colgado.php’ con máxima prioridad.

Compresión y combinación de archivos

gzip

El comando gzip permite comprimir y descomprimir archivos.

$gzip -d file	
d	Se adiciona en caso de que se quisiera descomprimir en vez de descomprimir.
file	Nombre del archivo que se quiere comprimir

```
$gzip index.html
```

En el ejemplo anterior, gzip crearía el archivo index.html.gz

tar

El comando tar permite empaquetar archivos.

$tar flag name	
cf	Empaqueta
tf	Muestra contenido del paquete
cvf	Empaqueta y ver contenido del paquete
xf	Desempaquetar
czf	Empaquetar y comprimir
xf	Desempaquetar y descomprimir
name	Nombre del archivo resultante del empaquetamiento
dir	Directorio que se quiere manipular

```
$tar czf paquete.tar.gz /dir/paquete
```

En el ejemplo anterior, tar crearía el archivo “paquete.tar.gz” proveniente de “/dir/paquete”

Búsqueda de archivos

locate

El comando locate permite realizar una búsqueda en todo el sistema de archivos.

$locate word_search	
word_search	Nombre del archivo que se quiere buscar

```
$locate index
```

En el ejemplo anterior, locate devolvería todos los archivos del sistema que incluyan la palabra “index”

Debe tenerse en cuenta que el comando está condicionado de una base de datos que debe ser actualizada. Para conseguirlo, existe el comando $sudo updatedb

whereis

El comando whereis permite realizar búsquedas más complejas y versátiles:

$whereis dir flag term	
dir	Directorio en el que se buscará
perm	Permisos específicos que tiene el archivo. E.j: -perm 666
mtime	Última fecha de modificación. E.j: -mtime +7 (modificado hace más de 7 días)
user	Usuario al que pertenece el archivo. E.j: -user gollum
type	Tipo de archivo: E.j: -type f
exec	Permite realizar una acción con cada resultado encontrado.
term	Término a buscar

```
$find . -type f -mtime +7 -exec cp {} ./backup/
```

En el ejemplo anterior, find buscará archivos de tipo “file” con más de 7 días de antigüedad y tras haberle adicionado el flag exec, el comando cp copiará cada archivo encontrado ({}) en el directorio “./backup/”

find

El comando find permite realizar búsquedas de comandos:

$find comando	
comando	Nombre del comando a buscar

```
$whereis echo
```

En el ejemplo anterior, whereis devolvería todos los archivos del sistema que que poseen el comando echo.

Interacción con HTTP desde consola.

curl

El comando curl nos imprime el HTML de una página web:

$curl flag site	
v	Al adicionar el modificador v, se añadirá al HTML toda la comunicación HTTP.
site	Dominio o dirección IP que ofrece contenido HTML desde el puerto 80.

```
$curl https://platzi.com
```

En el ejemplo anterior, curl devolvería el HTML del home del sitio web Platzi.

wget

El comando wget nos imprime realizar descargas desde servidores HTTP o FTP.

$wget host	
host	Dominio o dirección IP que ofrece el archivo a descargar.

```
$wget https://platzi.com/file.tar.gz
```

En el ejemplo anterior, wget nos guardará el archivo file.tar.gz

Automatizar scripts

at

El comando at nos permite programar la ejecución de un comando en algún momento relativo al presente.

$at now time	
time	El tiempo que tendrá que pasar para que se ejecute el programa. E.j: +2 minutes.

```
$at now +60 minutes
```

En el ejemplo anterior, at nos permitirá programar algún comando que se ejecutará en 60 minutos. Para salir del modo edición de comando, usamos: **ctrl +d.**

crontab

El comando crontab nos permite programar comandos que se ejecutarán de manera periódica.

$crontab flag	
e	Nos permite editar las tareas y crear nuevas.

```
$crontab -e
```

En el ejemplo anterior, crontab nos abrirá un documento de texto en el que podremos adicionar al final nuevos comandos programados con la siguiente sintaxis:

```
minuto hora día(del mes) mes día(de la semana) comando.
```

Cabe destacar que añadiendo un asterisco en vez de un número concreto para algún valor temporal, se hace una referencia para cada valor. Es decir, si en día del mes pongo 10, se ejecutaría el 10 de cada mes, pero si pongo en cambio *, el comando se ejecutará todos los días del mes.

Por ejemplo:

```
30 14 * * 5 echo “Recuerda que es viernes!!!”.
```

Hará un echo en la consola cada viernes a las 14:30 que dice: Recuerda que es viernes!!!.

service

El comando service nos permite controlar los comandos programados.

$service

`$service --status-all`
Nos devolverá el estado de todos los servicios actualmente.
`$sudo service atd start`
Iniciará el servicio de atd.
`$sudo service cron start`
Iniciará el servicio de cron.

Alias

Podemos crear shortcuts a comandos específicos con el uso de alias.

```
$alias conectar1='ssh root@184.144.61.112'
```

Si añadimos el anterior alias, podríamos ejecutar el comando “ssh root@184.144.61.112” utilizando únicamente la palabra del alias “conectar1”.

En el caso de mac, los alias deben guardarse en un archivo llamado .bash_profile en el home del usuario y posteriormente hacer uso del siguiente comando para que la terminal lo detecte:

```
$source ~/.bash_profile
```