

TALLER DE TERMINAL

Javier Gatón Herguedas
Pablo Martínez López

GUI
Grupo Universitario de Informática
Escuela de Ingeniería Informática, Universidad de Valladolid

Lunes, 1 de Abril de 2019

Comandos en sistemas Linux.

Comandos avanzados.

Vi(m)

Utilidades

Agradecimientos

Comandos básicos (I).

mv & cp

Permiten mover y copiar archivos o directorios. Sigue la sintaxis `mv/cp <ruta actual del archivo> <ruta futura>`.

mkdir & cd

El primero permite crear un directorio y el segundo permite movernos entre directorios, utilizando rutas absolutas o relativas.

rm & rmdir

Permiten eliminar archivos y directorios en ese orden. El comando `rm` tiene múltiples opciones como `-rf`. Es conveniente saber que el carácter `?` significa **cualquier carácter** y el carácter `*` significa **cualquier conjunto de caracteres**.

Comandos básicos. (II)

ls

Lista los elementos contenidos en un directorio. Tiene múltiples opciones como -A o menos -l.

cat

Muestra por pantalla el contenido de un fichero sin necesidad de abrirlo.

pwd

Muestra la ruta absoluta del directorio actual.

Comandos básicos. (III)

ps

Muestra los procesos actuales del sistema y su **PID** o identificador de proceso. Las variantes más comunes son -A (total de procesos) o -ejX (árbol de procesos).

kill

Permite terminar un programa a partir de su PID. Normalmente se utiliza la opción -9 y el PID de un proceso. <kill -9 PID del proceso a terminar >

Comandos básicos. (IV)

man

Manual del sistema. Muestra las opciones y las explica de los comandos y llamadas al sistema. Todos los comandos explicados tienen opciones. Se explican las más básicas pero os animamos a ver cuales más hay.

sudo & chmod

Sudo es el acrónimo de "super user do" y nos permite acceder como root. Chmod nos permite modificar los permisos de los ficheros y directorios entre: usuario/grupo/otros. Es posible hacerlo en modo verbal u octal.

Otros

echo - passwd - whoami - date

Comandos en sistemas Linux.

Comandos avanzados.

Vi(m)

Utilidades

Agradecimientos

GREP & FIND

grep

Busca una expresión en un archivo o conjunto de archivos y muestra las coincidencias por pantalla. Su sintaxis es `<grep cadena_a_buscar archivo>`. La opción **-n** nos especifica el número de línea y puedo darnos una búsqueda más eficiente.

find

Busca y muestra por pantalla todos los archivos en un directorio. Su sintaxis es `<find directorio>`. Con la opción **-name** podemos buscar un fichero determinado. Su sintaxis es `<find directorio -name archivo>`.

TR & NOHUP

tr

Modifica una salida de texto acorde a los parámetros del comando. Por ejemplo: `<echo "hola a todos" — tr a-z A-Z>`. Puede ser útil para tratar ficheros cambiando parámetros: `<tr a-z A-Z <fichero_inicial> fichero_final>`.

nohup

El nombre proviene de "No Hangups". El comando ejecutado no se detiene una vez comenzado hasta que termine o sea terminado manualmente. Si el usuario que inició el comando cierra la sesión sigue ejecutando. Cabe destacar también el carácter **&**, el cual permite ejecutar un comando en segundo plano.

Redireccionamiento.

Podemos redirigir la salida de los comandos bien a otros comandos o bien a ficheros.

Tuberías

Redirigen la salida de un comando a otro mediante el carácter '|'. Un ejemplo sería **grep "usuario" /ABC/doc.txt | wc -l**¹

Redireccionamiento básico o "flechitas"

Redirigen la salida de un comando o programa a un fichero. Existen dos variantes: '>', que sobrescribe el contenido del fichero y '>>', que añade al contenido. Añadiremos un **2** antes de los guiones para redirigir los errores. La sintaxis es: <comando > fichero>.

¹wc permite contar diversos campos, en este caso líneas.

Vi(m)

Utilidades

Agradecimientos

Comandos en sistemas Linux.

Comandos avanzados.

Introducción y modos

Vi y su versión mejorada (de cara al público) es un editor de texto que se basa en comandos. Esto puede causar una gran dificultad al iniciarse, pero proporciona una gran eficiencia para trabajar al dominarlo.

Visualizar

Modo inicial dedicado a los comandos. No se permite la escritura y se utiliza para las modificaciones a partir de comandos.

Insertar

Modo dedicado a la inserción de texto. Accedemos a él de varias formas: i(insertar a la izquierda del cursor), a(insertar a la derecha del cursor), o(insertar en línea inferior) u O(insertar en línea superior).

Reemplazar

Reemplaza caracteres. Similar a presionar "Insert".

Comandos en vi. (I)

Cortar, copiar y pegar

Los comandos son **dd**, **yy** y **p** en este orden. Podemos sustituir la segunda letra para cambiar su funcionalidad: w (word), l(letter).

Búsqueda y desplazamiento

Podemos utilizar el comando / seguido de una palabra para buscarla en el sistema. Avanzamos de palabra con **n** y retrocedemos con **N**. Podemos buscar una línea determinada con **:**. La última línea se accede con **\$**.

Comandos en vi (II).

Reemplazo

Utilizaremos el comando **r** para reemplazar una determinada letra. Para reemplazar palabras utilizaremos el comando **:s/palabra/reemplazo** para reemplazar en la línea actual. Utilizaremos **:s %** para abarcar todo el texto.

Salida y guardado

Para guardar utilizamos el comando **:w** y para salir el comando **:q**, pudiendo combinarlos con **:wq**. Utilizar el comando **ZZ** tiene una función idéntica. Para salir sin guardar utilizaremos **:q!**.

Vi(m)

Utilidades

Agradecimientos

Comandos en sistemas Linux.

Comandos avanzados.

Paquetes.

Dedicamos esta sección a comprender como instalar paquetes en sistemas Linux.

apt-get

Descarga e instala un paquete de los repositorios de la distribución. Estos paquetes se actualizan a través de dos comandos: apt-get **update**, el cual descarga las actualizaciones; y apt-get **upgrade**, el cual instala las actualizaciones.

tar-gz

Aunque es una colección de archivos y no un paquete como tal, es útil resaltarlo pues puede parecer confuso inicialmente. Utilizaremos el comando **tar xvf** para extraer los archivos. Normalmente este tipo de archivos trae instrucciones para su ejecución.

SSH & SCP.

ssh

Permite acceder a un servidor remoto. En el caso de nuestra universidad, **jair**. Su sintaxis es: `<ssh usuario@dominio>` (@jair.lab.inf.uva.es)

scp

Permite transferir archivos de un servidor remoto al sistema o al revés. Su sintaxis (para transferir del sistema al ordenador, de otra forma sería a la inversa) sería: `<scp usuario@servidor: directorio remoto directorio actual>`

CRON & CRONTAB

Cron es un **daemon** ejecutado al iniciar el sistema el cual comprueba si existen tareas a ser ejecutadas a una hora determinada. Encontramos el archivo **crontab** en `/etc/crontab`. Este archivo contiene las tareas a ejecutar y su hora asociada.

La sintaxis de las tareas es la siguiente: `<dia del mes - mes - dia de la semana - hora - minuto - usuario - comando>`.

- Podemos utilizar un comando o la ruta a un script.
- Al hacer el script no debemos olvidar darle permisos de ejecución: **chmod u/g/o +x archivo**.

Variables

- En shell también hay variables.
- Usamos **\$nombreVariable** para acceder a su valor.
- Usamos **nombreVariable=nuevoValor** para modificar su valor.
- Las variables son para la shell actual, para mantenerlas en subshells hijas hay que declararlas con **export variable=valor**.
- Existen variables de sistema como **PATH**² o **PS1**³ que son importantes para el funcionamiento del mismo.

²Las rutas al sistema se declaran mediante esta variable.

³Permite modificar el prompt.

Alias.

- Los alias son comandos simples que no toman argumentos y que no ocupan más que una sentencia. Se utilizan para funcionalidades sencillas.
- Encontraremos el fichero **.bash_aliases** en nuestro directorio HOME ⁴.
- Accedemos a él mediante vi.

Ejemplos de alias

```
alias jair = 'ssh usuario@dominio'  
alias bright = 'xrandr - -output eDP - -brightness'
```

⁴Utilizando ls -A podemos verlo

Scripts.

- Un programa compuesto de comandos en bash
- Se ejecuta en una subshell, no en el shell actual. Para que se ejecute en el actual hay que ejecutarlo con el comando "source".
- Para pasar argumentos al script se introducen normalmente. Para acceder a ellos desde el script se utiliza '\$_', siendo _ el numero de argumento.

Funciones.

- Las funciones se declaran como "función()" y se ejecutan como un comando normal "función".
- Al declarar una función esta no servirá para las subshells de la sesión salvo que la exportes con `export -f function_name`.

Ejemplo de función

```
mkgo(){  
    mkdir "$1"  
    cd "$1"  
}
```

Bashrc.

- El archivo `.bashrc` se ejecuta al iniciar sesión de usuario en el sistema.
- Este archivo contiene varias definiciones, funciones y comandos ejecutados. Añadiremos nuestras sentencias y funciones al final.
- `.bashrc` se ejecuta en sesión interactiva en shell non-login.
- También se ejecuta el archivo `/etc/bash.bashrc`.

Bash_profile

- **.bash_profile.**
- Es una sesión interactiva de login shell.
- También se ejecutan **.bash_login** o **.profile**.
- En este archivo se define el nombre de la máquina.

Customización.

Con estos comandos hemos creado el cartel del taller.

figlet

Permite crear letras de grandes proporciones a partir de letras convencionales.

cowsay

Crea animales en ASCII que dicen un mensaje que le pasemos por parámetros.

fortune

Imprime por pantalla un mensaje aleatorio (normalmente gracioso) de unas bases de datos que seleccionamos al instalar el paquete.

Otros

Existen otros comandos más pasivos como **sl** o **asciiaquarium**.

Comandos en sistemas Linux.

Comandos avanzados.

Vi(m)

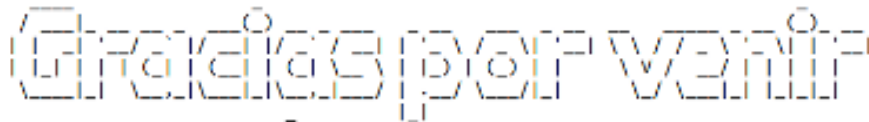
Utilidades

Agradecimientos

Gracias por asistir

Esperamos que os haya gustado y sobre todo que hayáis aprendido y que os hayáis animado a trabajar de forma más eficiente a través de comandos y vi.

```
pablo@hylianpablo:~/IPC$ figlet Gracias por venir
```



```
pablo@hylianpablo:~/IPC$ █
```

Repositorio: <https://github.com/HylianPablo/TallerTerminal2019>

Telegram: @HylianPablo @Javgat