Glosario de comandos

En el Práctico 1 vimos muchos de estos comandos y ejemplos de ellos, además de empezar a usarlos para la resolución de algunos problemas básicos con la terminal de Linux. La intención de este archivo es una referencia a la cual puedan recurrir en clase u otra instancia si no se acuerdan de un comando o alguno de sus argumentos más frecuentes.

0. `ssh`

- **Descripción**: Permite acceder de manera segura a una máquina remota a través de una conexión encriptada.
- **Parámetros comunes**:
  - `user@host`: Especifica el usuario y la dirección del host remoto.
  - `-p`: Especifica el puerto de conexión (por defecto es el 22).

```
ssh usuario@direccion_del_servidor
ssh -p 22222 usuario@direccion_del_servidor
```

1. `cd`

- **Descripción**: Cambia el directorio actual.
- **Parámetros comunes**:
  - `cd /ruta/a/directorio`: Cambia al directorio especificado.
  - `cd ..`: Sube un nivel en el directorio actual.
  - `cd ~`: Cambia al directorio personal del usuario.

```
cd /aca/algun/directorio
```

2. `ls`

- **Descripción**: Lista el contenido de un directorio.
- **Parámetros comunes**:
  - `-l`: Muestra los archivos en formato de lista larga, incluyendo permisos, propietario, tamaño y fecha de modificación.

- `-a`: Muestra todos los archivos, incluidos los ocultos (que comienzan con `.`).
- `-h`: Muestra tamaños de archivos en formato legible para humanos (ej. `K`, `M`).

```
ls -lah
```

3. `pwd`

- **Descripción**: Imprime el directorio de trabajo actual (Print Working Directory).
- **Parámetros comunes**:
  - Sin parámetros: Muestra la ruta completa del directorio actual.

```
pwd
```

4. `cat`

- **Descripción**: Muestra el contenido de un archivo o concatena varios archivos.
- **Parámetros comunes**:
  - Sin parámetros: Muestra el contenido del archivo.
  - **Concatenación**: Puedes concatenar múltiples archivos en uno solo.

```
cat archivo.txt
cat archivo1.txt archivo2.txt > archivo_combinado.txt
```

5. `cp`

- **Descripción**: Copia archivos o directorios.
- **Parámetros comunes**:
  - `-r`: Copia directorios de forma recursiva.
  - `-i`: Pregunta antes de sobrescribir archivos existentes.
  - `-p`: Preserva los atributos del archivo (como la fecha de modificación).

```
cp archivo.txt /ruta/al/destino/
cp -r directorio/ /ruta/al/destino/
```

6. `mv`

- **Descripción**: Mueve o renombra archivos o directorios.
- **Parámetros comunes**:
  - `-i`: Pregunta antes de sobrescribir archivos existentes.
  - `-u`: Solo mueve si el archivo de origen es más reciente que el archivo de destino o si no existe.

```
mv archivo.txt /ruta/al/destino/
mv archivo_viejo.txt archivo_nuevo.txt
```

7. `mkdir`

- **Descripción**: Crea un nuevo directorio.
- **Parámetros comunes**:
  - `-p`: Crea directorios de forma recursiva, es decir, también crea todos los directorios padres necesarios.

```
mkdir nuevo_directorio
mkdir -p ruta/al/nuevo/directorio
```

8. `rm`

- **Descripción**: Elimina archivos o directorios.
- **Parámetros comunes**:
  - `-r`: Elimina directorios de forma recursiva. **Hay que tener mucho cuidado porque no hay vuelta atrás luego de borrar un archivo.**

```
rm archivo.txt
rm -r directorio/
```

9. `grep`

- **Descripción**: Busca patrones (entre ' ') en archivos o texto.
- **Parámetros comunes**:

- `-i`: Ignora mayúsculas y minúsculas.
- `-n`: Muestra el número de línea junto con la coincidencia.
- `-r`: Realiza una búsqueda recursiva en todos los archivos de un directorio.
- `-v`: Invierte la coincidencia, mostrando líneas que no coinciden con el patrón.
- `-o`: (abreviatura de "only matching") muestra las partes de las líneas que coinciden exactamente con el patrón de búsqueda, en lugar de mostrar la línea completa.
- `-A`: (abreviatura de "after") permite mostrar un número específico de líneas que siguen a la línea coincidente con el patrón de búsqueda.

```
grep -rn 'patrón' /ruta/al/directorio
grep -A 3 'error' archivo.log
```

10. `more`

Descripción: `more` permite visualizar el contenido de un archivo de texto página por página en la terminal. Es útil cuando el archivo es demasiado largo para caber en una sola pantalla.

```
more archivo.txt
```

Explicación:

- Este comando abrirá el archivo `archivo.txt` y mostrará su contenido una página a la vez. Para avanzar a la siguiente página, presiona la barra espaciadora (`Space`), y para salir, presiona `q` .

11. `less`

- **Descripción**: `less` es similar a `more`, pero ofrece más funcionalidades, como la capacidad de moverse hacia adelante y hacia atrás dentro del archivo. Es más flexible y rápido para navegar en archivos grandes.

Ejemplo: Buscar dentro de un archivo

```
less archivo.txt
```

- **Buscar**:
  - Una vez dentro de `less`, puedes buscar una palabra o patrón presionando `/` seguido del término de búsqueda.

- **Ejemplo**: Para buscar la palabra "error", presiona `/error` y luego `Enter`.
- Navega entre las coincidencias usando `n` (siguiente) y `N` (anterior).

Controles en `less`:

- `Space`: Avanza una página.
- `b`: Retrocede una página.
- `g`: Va al inicio del archivo.
- `G`: Va al final del archivo.
- `/patrón`: Busca un patrón dentro del archivo.
- `n`: Muestra la siguiente coincidencia en la búsqueda.
- `N`: Muestra la coincidencia anterior en la búsqueda.
- `q`: Sale de `less`.

12. `head` ; `tail`

- **Descripción**: Muestra las primeras/últimas líneas de un archivo.
- **Parámetros comunes**:
  - `-n X`: Muestra las primeras/últimas X líneas del archivo (por defecto, muestra 10).

```
head -n 20 archivo.txt
```

13. `chmod`

- **Descripción**: Cambia los permisos de archivos o directorios.
- **Parámetros comunes**:
  - `u`, `g`, `o`: Especifica si los cambios afectan al usuario (`u`), grupo (`g`) u otros (`o`).
  - `r`, `w`, `x`: Permisos de lectura (`r`), escritura (`w`) y ejecución (`x`).
  - Modo numérico: Cambia los permisos mediante un código de tres números (ej. `755`), uno para el propietario del archivo, otro para el grupo, y otro para otros usuarios. Cada dígito puede ser una suma de los siguientes valores:
- **4**: Permiso de lectura (`r`)
- **2**: Permiso de escritura (`w`)
- **1**: Permiso de ejecución (`x`)

```
chmod u+x archivo.sh
```

```
chmod 644 archivo.txt
```

14. `>` (Redirección de Salida) ; `>>` (Redirección sin sobreescribir)

- **Descripción**: `>` Redirige la salida de un comando a un archivo, sobrescribiendo su contenido.
- **Ejemplo**:

```
echo "Texto de ejemplo" > archivo.txt
```

Descripción: `>>` Redirige la salida de un comando al final de un archivo existente, sin sobrescribir su contenido. Verifiquen con un ejemplo.

15. `|` (Pipe)

- **Descripción**: Pipe pasa la salida de un comando como entrada para otro comando.
- **Ejemplo**:

```
cat archivo.txt | grep 'patrón'
```

- **Qué hace**:
  - `cat archivo.txt`: Muestra el contenido completo del archivo `archivo.txt`.
  - `|` **(Pipe)**: Toma la salida del comando `cat archivo.txt` y la pasa como entrada al siguiente comando (`grep`).
  - `grep 'patrón'`: Busca dentro del texto proporcionado todas las líneas que contienen la palabra o patrón `patrón`.
- **Resultado**:
  - Este comando mostrará solo las líneas del archivo `archivo.txt` que contienen el texto `patrón`.

16. `find`

- **Descripción**: Busca archivos y directorios en un árbol de directorios.
- **Parámetros comunes**:
  - `-name`: Busca archivos por nombre.

- `-type`: Especifica el tipo de archivo (`f` para archivo, `d` para directorio).
- `-size`: Busca archivos de un tamaño específico.

```
find /ruta/a/buscar/ -name "*.txt"
find . -type d -name "proyectos"
find . -size +50M
```

17. `locate`

- **Descripción**: Encuentra rápidamente archivos y directorios mediante una base de datos indexada.
- **Parámetros comunes**:
  - Sin parámetros: Busca archivos o directorios que coincidan con el término proporcionado.

```
locate archivo.txt
```

18. `echo`

- **Descripción**: Muestra un mensaje en la terminal o redirige texto a un archivo.
- **Parámetros comunes**:
  - Sin parámetros: Imprime el texto especificado.
  - `>>`: Añade el texto a un archivo existente.

```
echo "Hola, mundo!"
echo "Nueva línea" >> archivo.txt
```

19. `uniq`

- **Descripción**: Muestra o elimina líneas duplicadas en un archivo o entrada.
- **Parámetros comunes**:
  - `-c`: Cuenta las ocurrencias de cada línea.
  - `-d`: Muestra solo las líneas que están duplicadas.
  - `-u`: Muestra solo las líneas únicas.

```
sort archivo.txt | uniq
uniq -c archivo.txt
```

20. `sort`

- **Descripción**: Ordena las líneas de un archivo de texto en orden alfabético o numérico.
- **Parámetros comunes**:
  - `-n`: Ordena numéricamente.
  - `-r`: Ordena en orden inverso (descendente).
  - `-k`: Especifica la columna por la cual ordenar.
  - `-t`: Especifica un delimitador de campo.

```
sort archivo.txt # Ordena las líneas alfabéticamente
sort -n archivo.txt # Ordena las líneas numéricamente
sort -r archivo.txt # Ordena las líneas en orden inverso
sort -t',' -k2 archivo.csv # Ordena un archivo CSV por la segunda columna
```

21. `gunzip` ; `gzip`

- **Descripción**: `gunzip` es una utilidad que descomprime archivos que han sido comprimidos con el formato `gzip` (`.gz`). Se utiliza para restaurar archivos a su estado original después de haber sido comprimidos.
- **Parámetros comunes**:
  - Sin parámetros: Descomprime un archivo `.gz` y reemplaza el archivo comprimido con el archivo descomprimido.
  - `-c`: Descomprime el archivo, pero mantiene el archivo original comprimido.
  - `-k`: Conserva el archivo original comprimido después de descomprimirlo.

```
gunzip archivo.txt.gz
gunzip -c archivo.txt.gz > archivo.txt
```

22. `tar`

- **Descripción**: `tar` es una herramienta de archivado que se utiliza para agrupar múltiples archivos en un solo archivo comprimido o descomprimido. A menudo se combina con

`gzip` para crear archivos comprimidos con extensión `.tar.gz`.

- **Parámetros comunes**:
  - `-c`: Crea un nuevo archivo tar.
  - `-x`: Extrae archivos de un archivo tar existente.
  - `-v`: Muestra información detallada del proceso (modo "verboso").
  - `-f`: Especifica el nombre del archivo.
  - `-z`: Comprime o descomprime usando `gzip`.

```
tar -czvf archivo.tar.gz /ruta/al/directorio # Crea un archivo tar.gz
tar -xzvf archivo.tar.gz # Extrae un archivo tar.gz
```

23. `cut`

- **Descripción**: Recorta secciones de cada línea de un archivo o entrada, por ejemplo, seleccionando ciertas columnas.
- **Parámetros comunes**:
  - `-d`: Especifica el delimitador de campos (por defecto es la tabulación, es decir que hay un tab entre columnas).
  - `-f`: Especifica qué campos mostrar.

```
cut -d',' -f1,3 archivo.csv
```

24. `wget`

- **Descripción**: `wget` es una herramienta de línea de comandos que permite descargar archivos desde la web mediante HTTP, HTTPS o FTP. Es ideal para automatizar la descarga de archivos en entornos sin interfaz gráfica. Siempre tengan cuidado con el tamaño del archivo que van a bajar.
- **Parámetros comunes**:
  - `-O`: Especifica el nombre del archivo de salida.
  - `-q`: Descarga en modo silencioso (sin mostrar progreso).

```
wget https://ejemplo.com/archivo.txt
wget -O nuevo_nombre.txt https://ejemplo.com/archivo.txt
```

25. `man` ; `--help`

- **Descripción**: `man` muestra el manual de usuario para un comando.
- **Parámetros comunes**:
  - Sin parámetros: Muestra el manual del comando especificado.

Descripción: `--help` es un parámetro común que se puede añadir a casi cualquier comando en Linux para obtener una descripción rápida de cómo se usa el comando, junto con sus opciones y ejemplos básicos. Es una forma conveniente de acceder a la documentación sin necesidad de abrir manuales completos.

Uso: Se coloca después del nombre del comando del cual quieres obtener ayuda.

```
man ls
ls --help
```

26. `touch`

- **Descripción**: Crea un archivo vacío o actualiza la marca de tiempo de un archivo existente.
- **Parámetros comunes**:
  - Sin parámetros: Crea un archivo vacío si no existe, o actualiza la fecha y hora de acceso/modificación si ya existe.

```
touch nuevo_archivo.txt
```

27. `nano`

- **Descripción**: Un editor de texto en la línea de comandos, fácil de usar. Abajo se despliega una lista de opciones para navegar, por ejemplo `Ctrl + Y` para desplazarse una pantalla hacia arriba en el texto, `Ctrl + W` para buscar texto, y `Ctrl + X` para salir de nano (te preguntará si deseas guardar los cambios).
- **Parámetros comunes**:
  - Sin parámetros: Abre el archivo especificado en el editor.

```
nano archivo.txt
```

! ATENCIÓN !

Estos últimos comandos no serán necesarios durante el curso.

Se incluyen para demostrar el potencial que tiene la línea de comando. No tendrán que usarlos durante el práctico pero pueden ser de mucha utilidad.

`sed`

- **Descripción**: Un editor de flujo que permite realizar búsquedas y reemplazos de texto en archivos o entrada estándar.
- **Parámetros comunes**:
  - `s/patrón/reemplazo/g`: Reemplaza todas las ocurrencias de un patrón por un texto específico en cada línea.
  - `-i`: Edita el archivo original en lugar de imprimir los cambios en la salida estándar.

Ejemplos básicos

```
sed 's/viejo/nuevo/g' archivo.txt
# Reemplaza "viejo" por "nuevo" en archivo.txt y lo imprime en la salida
estandar

sed -i 's/error/corrección/g' archivo.txt
# Realiza el reemplazo directamente en el archivo
```

Explicación:

- `s/viejo/nuevo/g`: Reemplaza todas las ocurrencias de "viejo" con "nuevo" en cada línea del archivo `archivo.txt`.
- `g`: Aplica el reemplazo globalmente en cada línea (sin `g`, solo reemplazaría la primera ocurrencia en cada línea).
- `-i`: Edita el archivo `archivo.txt` en el lugar, sin necesidad de crear un archivo temporal.
- `s/abc/xyz/g`: Reemplaza todas las ocurrencias de "abc" con "xyz" en cada línea del archivo.

Ejemplo avanzado

Problema: Supongamos que tienes un archivo FASTA que contiene múltiples secuencias, y deseas eliminar todas las secuencias que contengan una región con más de 10 bases consecutivas de "N" (indicando regiones de baja calidad o secuencias ambigua

```
sed -n '/^>/h; /^>/!H; $!d; x; /NNNNNNNNNN/!p' secuencias.fasta >
filtrado.fasta
```

Explicación:

- `-n`: Desactiva la salida automática.
- `/^>/h`: Guarda las líneas de encabezado (que comienzan con `>`).
- `/^>/!H`: Agrega las líneas de secuencia a un búfer temporal.
- `$!d`: Si no es la última línea, elimina la entrada (esto evita la salida prematura).
- `x`: Intercambia el búfer de patrones con el búfer de retención.
- `/NNNNNNNNNN/!p`: Imprime las secuencias que **no** contienen 10 o más "N" consecutivas.
- **Resultado**: Filtra las secuencias en el archivo `secuencias.fasta`, eliminando aquellas que contienen regiones ambiguas largas, y guarda el resultado en `filtrado.fasta`.

`awk`

Descripción: `awk` es un poderoso lenguaje de programación orientado al procesamiento de texto y extracción de patrones. Se utiliza comúnmente para manipular y analizar datos estructurados en columnas. Existen libros enteros dedicado al uso de `awk`, y si bien no se usa en este curso es una herramienta poderosa para cualquier persona que trabaje extensivamente con la línea de comando.

Ejemplos básicos

Ejemplo 1

```
awk -F',' '{print $2}' archivo.csv
```

- **Explicación**:
  - `-F','`: Define la coma como delimitador de campos (por defecto, `awk` usa espacios o tabulaciones).
  - `'{print $2}'`: Imprime la segunda columna de cada línea del archivo `archivo.csv`.
- **Resultado**:

- Se imprimirán los valores de la segunda columna para cada línea del archivo CSV, útil cuando trabajas con archivos delimitados por comas.

Ejemplo 2

```
awk '$3 > 50 {sum+=$3} END {print "Total:", sum}' archivo.txt
```

- **Explicación**:
  - `'$3 > 50 {sum+=$3}'`: Suma los valores de la tercera columna que sean mayores a 50.
  - `END {print "Total:", sum}`: Después de procesar todas las líneas, imprime el total de la suma.
- **Resultado**:
  - Este comando suma todos los valores en la tercera columna que son mayores a 50 y luego imprime el total.

Ejemplo 3

```
awk '$1 == "error"' archivo.log | wc -l
```

- **Explicación**:
  - `'$1 == "error"'`: Filtra líneas donde la primera columna es igual a "error".
  - `| wc -l`: Cuenta el número de líneas que cumplen con la condición.
- **Resultado**:
  - Imprime el número de veces que aparece "error" como la primera palabra en el archivo de registro `archivo.log`.

Ejemplo avanzado

Problema: Tienes un archivo con datos de expresión génica para diferentes condiciones experimentales, y deseas calcular el coeficiente de variación (CV = desviación estándar/media) para cada gen y marcar los que tienen un CV mayor a 0.5.

```
awk '{
  if(NR > 1) {
    sum=0; sumsq=0;
    for(i=2; i<=NF; i++) {
      sum += $i;
      sumsq += ($i)^2;
```

```
    }
    mean = sum / (NF-1);
    variance = (sumsq - (sum^2)/(NF-1)) / (NF-2);
    stdev = sqrt(variance);
    cv = stdev / mean;
    if(cv > 0.5) print $1, cv;
  }
}' datos_expresion.txt
```

Explicación:

- `NR > 1` : Omite la primera línea (asumiendo que es un encabezado).
- `sum=0; sumsq=0;` : Inicializa las variables para sumar los valores y las sumas al cuadrado.
- `for(i=2; i<=NF; i++) { ... }` : Recorre todas las columnas a partir de la segunda (asumiendo que la primera es el nombre del gen).
- `mean = sum / (NF-1);` : Calcula la media de la expresión.
- `variance = (sumsq - (sum^2)/(NF-1)) / (NF-2);` : Calcula la varianza.
- `stdev = sqrt(variance);` : Calcula la desviación estándar.
- `cv = stdev / mean;` : Calcula el coeficiente de variación (CV).
- `if(cv > 0.5) print $1, cv;` : Imprime el nombre del gen y su CV si el CV es mayor que 0.5.
- **Resultado**: Este script `awk` calcula el coeficiente de variación para cada gen y marca aquellos genes que tienen una variabilidad alta entre condiciones experimentales.