



CURSO TEÓRICO-PRÁCTICO:

EPIDEMIOLOGÍA GENÓMICA Y DESCUBRIMIENTO DE PATÓGENOS MEDIANTE SECUENCIACIÓN DE PRÓXIMA GENERACIÓN / GENOMIC EPIDEMIOLOGY AND PATHOGEN DISCOVERY BY NEXT GENERATION SEQUENCING

Noviembre 8 - 12, 2021

Introducción a la línea de comandos de Linux

Los comandos se encuentran en letra Courier dentro de cajas.

1. Identifique su nombre de usuario y nombre del equipo en la terminal

Comandos: whoami

hostname

Este comando nos devuelve el nombre de usuario

```
ins@servidorInvestigacion:~$
```

usuario computador

```
ins@srvinvestigacon:~/taller$ whoami  
ins
```

2. ¿En qué carpeta estamos trabajando?

Con el comando `pwd` podemos ver en qué directorio estamos ubicados y conocer su PATH completo.

```
ins@srvinvestigacon:~$ pwd  
/home/ins
```

3. Cree un directorio nuevo

`mkdir` crea directorios. Podemos crear un directorio para el taller

```
mkdir taller
```

```
ins@srvinvestigacon:~$ mkdir taller
ins@srvinvestigacon:~$
```

También podemos intentar crear subdirectorios

```
mkdir taller/prueba/punto1
```

```
ins@srvinvestigacon:~$ mkdir taller/prueba/punto1
mkdir: cannot create directory 'taller/prueba/punto1': No such file or directory
```

Pero el comando no nos permite crear el directorio *punto1* dentro del directorio inexistente *prueba*. **Veamos la ayuda del comando**

4. ¿Cómo veo la ayuda o instrucciones de los comandos?

Opcion en los commands: --help

Cada comando tiene una opción para desplegar la ayuda. La mayoría tiene la opción ayuda agregando --help, o -h.

```
mkdir --help
```

```
ins@srvinvestigacon:~$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.
-m, --mode=MODE    set file mode (as in chmod), not a=rwx - umask
-p, --parents       no error if existing, make parent directories as needed
-v, --verbose       print a message for each created directory
-Z                 set SELinux security context of each created directory
                   to the default type
--context[=CTX]    like -Z, or if CTX is specified then set the SELinux
                   or SMACK security context to CTX
--help             display this help and exit
--version          output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
```

Viendo la ayuda del comando encontramos las opciones que podemos usar con el comando mkdir. Entre estas vemos la opción -p que se asegura de crear directorios parentales si no existen.

```
mkdir -p /taller/prueba/punto1
```

```
ins@srvinvestigacon:~$ mkdir -p taller/prueba/punto1
ins@srvinvestigacon:~$
```

5. Ingrese al directorio *taller* que acabamos de crear

Para entrar a los directorios en nuestra ubicación usamos el comando `cd`.

```
cd taller
```

```
ins@srvinvestigacon:~$ cd taller/  
ins@srvinvestigacon:~/taller$
```

Vemos que nuestra ubicación en la consola ha cambiado a `~/taller`

El símbolo `~` representa el directorio home.

También podemos ingresar al directorio que queramos definiendo el PATH completo.

```
cd /home/mi_usuario/taller
```

El nombre de usuario lo podemos ver en la terminal, o con el comando `whoami`

```
ins@srvinvestigacon:~$ cd /home/ins/taller  
ins@srvinvestigacon:~/taller$
```

6. Revise qué contiene el directorio actual

Ahora que estamos dentro del directorio queremos ver su contenido. El comando `ls` nos muestra una lista del contenido

```
ins@srvinvestigacon:~/taller$ ls  
prueba
```

7. Revise los permisos, tamaño y fecha de modificación de los archivos

```
Comando: ls -lh
```

Con la opción `--help` podemos ver las opciones para visualizar esta lista.

Si queremos activar varias opciones al tiempo podemos usar un solo guion y las letras correspondientes.

```
ins@srvinvestigacon:~/taller$ ls -lh  
total 4.0K  
drwxrwxr-x 3 ins ins 4.0K Nov  9 02:56 prueba
```

Pregunta: ¿Qué nos muestra el comando con las opciones `-lh`?

Pregunta: ¿Cómo podemos ver archivos ocultos?

8. Sacar en pantalla un texto

Con echo podemos ver en pantalla el valor de variables o mostrar en el stdout.

Hay variables del sistema que podemos ver con echo, como la ubicación actual o el id del usuario \$UID

```
echo $UID
```

```
ins@srvinvestigacon:~/taller$ echo $UID
1000
```

9. Guarde la salida (stdout) en un archivo

La salida que veríamos en el stdout la podemos guardar con el símbolo >

```
echo "hello world" > hello.txt
```

```
ins@srvinvestigacon:~/taller$ echo "hello world" > hello.txt
ins@srvinvestigacon:~/taller$
```

Podemos agregar líneas al archivo hello.txt usando el símbolo >>

```
echo "hola mundo" >> hello.txt
```

```
echo "adios" >> hello.txt
```

```
ins@srvinvestigacon:~/taller$ echo "hola mundo" >> hello.txt
ins@srvinvestigacon:~/taller$ echo "adios" >> hello.txt
ins@srvinvestigacon:~/taller$
```

10. Ver el contenido de archivos

Los comandos cat, more, less, head, tail nos muestran el contenido de archivos de texto de diferentes formas.

```
more hello.txt
```

```
ins@srvinvestigacon:~/taller$ more hello.txt
hello world
hola mundo
adios
ins@srvinvestigacon:~/taller$
```

cat también puede ser usado para concatenar archivos de texto

con head podemos ver las primeras líneas de un archivo

con tail podemos ver las líneas finales

11. Clonar repositorios de GitHub

Para empezar, debemos descargar los archivos que usaremos en el taller. Los archivos están depositados en un repositorio de GitHub, desde donde los copiaremos

```
git clone https://github.com/dfbautista/Curso-Epidemiologia-Genomica-Practica-Bioinformatica/tree/main/Intro_LineaComandos_Linux
```

el contenido de este repositorio será clonado al directorio donde estemos ubicados.

12. Descargar archivos

Podemos usar el comando `wget` para descargar archivos.

```
wget https://github.com/dfbautista/Curso-Epidemiologia-Genomica-Practica-Bioinformatica/raw/main/Intro_LineaComandos_Linux/secuencias_virales.fasta
```

En la carpeta que descargamos hay un archivo fasta llamado **secuencias_virales.fasta**.

13. Pregunta: ¿Cuántas secuencias hay en este archivo?

Primero veamos cómo es el archivo:

```
head secuencias_virales.fasta
```

```
ins@srvinvestigacon:~/taller$ head secuencias_virales.fasta
>NC_055230.1 |Akhmeta virus isolate Akhmeta_2013-88, complete genome
TATTTTATTTAGTGTCTAGAAAAAATGTGTGACCACTACTGTCATAAACTCTAGAGGGT
AAGAAAATCAATCTCTTATAGAGACCATCGAAGAGAGAAAGGAGAAAGAATTTTTTTGT
AAAATTTTTTCACGACTCCATCAGAAAGAGGTTTAATTTTTTTGTGAGACCATCGAAGA
GAGAAAGGAGAAAGAATTTTTTTGTAAACTTTTTTCACGACTCCATCAGAAAGAGGTT
TAATTTTTTTGTGAGACCATCGAAGAGAGAAAGGAGAAAGAATTTTTTTGTAAACTTT
TTTCACGACTCCATCAGAAAGAGGTTTAATTTTTTTGTGAGACCATCGAAGAGAGAAAGG
AGAAAGAATTTTTTTGTAAACTTTTTTCACGACTCCATCAGAAAGAGGTTTAATTTT
TTGTGAGACCATCGAAGAGAGAAAGGAGAAAGAATTTTTTTGTAAACTTTTTTCACGA
CTCCATCAGAAAGAGGTTTAATTTTTTTGTGAGACCATCGAAGAGAGAAAGGAGAAAGAA
ins@srvinvestigacon:~/taller$
```

14. grep – Hacer una búsqueda dentro de un texto

Busca un patrón dado en el archivo o texto que indiquemos.

Podemos buscar en nuestro archivo `hello.txt` todas las veces que aparezca la letra `h`.

```
grep "h" hello.txt
```

```
ins@srvinvestigacon:~/taller$ grep "h" hello.txt
hello world
hola mundo
```

Este comando puede ser útil para explorar archivos de texto, por ejemplo archivos fasta.

La manera más fácil de ver cuantas secuencias hay sería buscando el símbolo ">" de cada header en el fasta y contándolos

```
grep -c ">" secuencias_virales.fasta
```

```
ins@srvinvestigacon:~/taller$ grep -c ">" secuencias_virales.fasta
262
```

15. Escribir scripts y archivos de texto con nano

Podemos crear archivos de texto con diferentes herramientas como nano o vim. nano es sencilla de usar. Llamamos el comando nano y el nombre del archivo que vamos a crear o editar.

```
nano archivo.txt
```

```
ins@srvinvestigacon:~/taller$ nano archivo.txt
GNU nano 5.2                                archivo.txt
[ New File ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Acá podemos escribir el contenido de nuestro archivo.

Vamos a escribir el encabezado de un script de bash

```
#!/bin/bash
```

Para salir damos Ctrl+x, luego confirmamos que queremos guardar los cambios

16. Mover archivos y cambiarles el nombre

Estas dos acciones las podemos realizar con el comando mv

Sí queremos cambiar el archivo de ubicación es como si reescribiéramos su PATH

```

ins@srvinvestigacon:~/taller$ mv archivo.txt prueba/.
ins@srvinvestigacon:~/taller$ ls prueba/
archivo.txt  punto1
ins@srvinvestigacon:~/taller$ █

```

Acá hemos movido el archivo al directorio prueba/

También podemos cambiar el nombre, o si quisiéramos realizar las dos acciones simultáneamente.

```

ins@srvinvestigacon:~/taller/prueba$ mv archivo.txt script.txt
ins@srvinvestigacon:~/taller/prueba$ ls
punto1  script.txt
ins@srvinvestigacon:~/taller/prueba$ █

```

17. Copiar archivos con cp

Para copiar archivos, llamamos el comando y luego definimos el path del archivo que queremos copiar y luego el path de destino.

```

ins@srvinvestigacon:~/taller/prueba$ cp script.txt ..
ins@srvinvestigacon:~/taller/prueba$ cd ..
ins@srvinvestigacon:~/taller$ ls
hello.txt  prueba  script.txt  secuencias_virales.fasta
ins@srvinvestigacon:~/taller$ █

```

18. Copiar archivos con rsync (sincronización de directorios)

rsync es una herramienta que permite sincronizar el contenido de directorios. Esto quiere decir que podemos ir agregando archivos sin tener que copiar nuevamente todo un directorio. Además tiene opciones que nos permiten visualizar el proceso.

```

ins@srvinvestigacon:~/taller$ rsync -avhP secuencias_virales.fasta ~/taller/prueba/punto1/
sending incremental file list
secuencias_virales.fasta
   2.19M 100%  79.04MB/s   0:00:00 (xfr#1, to-chk=0/1)

sent 2.19M bytes  received 35 bytes  4.38M bytes/sec
total size is 2.19M  speedup is 1.00
ins@srvinvestigacon:~/taller$ ls prueba/punto1/
secuencias_virales.fasta
ins@srvinvestigacon:~/taller$ █

```

19. Eliminar archivos

Eliminar archivos en Linux es potencialmente peligroso.

El comando rm nos permite eliminar archivos sencillos.

Si queremos eliminar directorios vacíos podemos usar rm -d

Para eliminar directorios y su contenido podemos usar rm -r

Linux no tiene papelera de reciclaje, así que al eliminar, esto se hace de manera permanente.

20. Ver el espacio disponible en disco

Comando: df

```
ins@srvinvestigacon:~/taller$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
tmpfs                  5140080      1200    5138880   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 5159024400 1115488652 3833070376 23% /
tmpfs                  25700380      16    25700364   1% /dev/shm
tmpfs                   5120         0         5120   0% /run/lock
tmpfs                   4096         0         4096   0% /sys/fs/cgroup
/dev/sda2              999320      220768    709740   24% /boot
tmpfs                  5140076         4    5140072   1% /run/user/1000
ins@srvinvestigacon:~/taller$
```

21. Ver el tamaño de un directorio

du -sh

¿Qué tamaño tienen los directorios que creamos?

```
ins@srvinvestigacon:~/taller$ du -sh
4.3M .
```

22. Realizar un conteo de palabras o líneas de un archivo

Podemos realizar un conteo del número de letras o palabras con el comando wc

Sí combinamos este comando con uno de búsqueda, por ejemplo grep, podemos obtener información de un archivo de texto.

Por ejemplo, ¿Cuántas bases tiene la secuencia del archivo secuencia_nc_055230_1.fasta?

Para conectar la salida de un comando con la entrada de otro podemos usar el símbolo del pipeline |.

```
| wc -l
```

```
ins@srvinvestigacon:~/taller$ grep ">" secuencias_virales.fasta | wc -l
262
ins@srvinvestigacon:~/taller$
```

23. Pausar la terminal por un tiempo

A veces es útil hacer una pausa entre comandos. Para esto podemos usar el comando sleep, y definir el tiempo que queremos que espere el sistema en horas (h), minutos (m), o segundos (s).

sleep

```
ins@srvinvestigacon:~/taller$ sleep 5s
```

24. Crear un script de bash con instrucciones

En muchas ocasiones debemos realizar una tarea o ejecutar un comando repetidas veces para distintos archivos de entrada. Para realizar esto debemos saber cómo se estructuran los scripts de bash.

En el editor de texto podemos agregar en la primera el término `#!/bin/bash`

Esto le indicará al sistema que lo que está escrito en el archivo son instrucciones en bash.

Probemos con el siguiente script:

```
GNU nano 5.2 script.txt Modified
#!/bin/bash

date
sleep 5s
echo "funciona"
sleep 3s
echo "prueba exitosa"
echo "la fecha es: "
date

Save modified buffer?
Y Yes
N No ^C Cancel
```

25. Cambiar permisos de lectura, escritura y ejecución de archivos

Para poder ejecutar nuestros scripts deben tener los permisos habilitados. Podemos usar `ls -lh` para revisar que permisos tenemos activos y el comando `chmod` para modificar los permisos.

```
chmod +x script.txt
```

```
ins@srvinvestigacon:~/taller$ ls -lh
total 2.2M
-rw-rw-r-- 1 ins ins 29 Nov 9 03:54 hello.txt
drwxrwxr-x 3 ins ins 4.0K Nov 9 04:38 prueba
-rw-rw-r-- 1 ins ins 100 Nov 9 05:01 script.txt
-rw-rw-r-- 1 ins ins 2.1M Nov 9 04:16 secuencias_virales.fasta
ins@srvinvestigacon:~/taller$
```

```

ins@srvinvestigacon:~/taller$ chmod +x script.txt
ins@srvinvestigacon:~/taller$ ls -lh
total 2.2M
-rw-rw-r-- 1 ins ins 29 Nov 9 03:54 hello.txt
drwxrwxr-x 3 ins ins 4.0K Nov 9 04:38 prueba
-rwxrwxr-x 1 ins ins 100 Nov 9 05:01 script.txt
-rw-rw-r-- 1 ins ins 2.1M Nov 9 04:16 secuencias_virales.fasta
ins@srvinvestigacon:~/taller$ █

```

Para ejecutar nuestro script es necesario llamarlo con su path completo. Lo más sencillo si está en nuestra ubicación actual es usar el símbolo .

```

ins@srvinvestigacon:~/taller$ ./script.txt
Tue Nov 9 05:07:19 AM UTC 2021
funciona
prueba exitosa
la fecha es:
Tue Nov 9 05:07:27 AM UTC 2021
ins@srvinvestigacon:~/taller$ █

```

26. ¿Cuánto se demora un comando en correr?

Si queremos saber cuánto tarda un comando en completar su función podemos agregar el termino time antes del comando o script. Al finalizar su ejecución obtendremos un reporte del tiempo que gastó.

```

ins@srvinvestigacon:~/taller$ time ./script.txt
Tue Nov 9 05:08:47 AM UTC 2021
funciona
prueba exitosa
la fecha es:
Tue Nov 9 05:08:55 AM UTC 2021

real    0m8.013s
user    0m0.012s
sys     0m0.002s
ins@srvinvestigacon:~/taller$ █

```

27. Ver qué procesos están corriendo

htop

```
1 [ 0.0%] 4 [ 0.0%]
2 [ 2.0%] 5 [ 0.0%]
3 [ 0.0%] 6 [ 0.7%]
Mem[|||||||||||||||||||||||||||||||||||||640M/49.0G]
Swp[| 30.1M/8.00G] Tasks: 51, 84 thr; 1 running
Load average: 0.29 0.20 0.11
Uptime: 38 days, 11:50:55

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
2020002 ins       20   0  8152   4324   3408 R   0.7  0.0  0:00.23 htop
901510 ins       20   0  6880   3200   2952 S   0.0  0.0  6:21.69 bash -c while [ -d /proc/$PPID ]; do sleep 1;head -v -n
876654 ins       20   0 14944   6276   4564 S   0.0  0.0  2:54.96 sshd: ins@pts/0
804 root       20   0  82724   3876   3568 S   0.0  0.0  3:17.98 /usr/sbin/irqbalance --foreground
683 root       RT    0  274M  18828   9032 S   0.0  0.0  0:29.79 /sbin/multipathd -d -s
897 root       20   0 1455M  25164  11500 S   0.0  0.0  1:40.52 /usr/bin/containerd
1 root       20   0  166M  13216   8460 S   0.0  0.0  44:18.39 /sbin/init
788 root       20   0  231M  8656   6476 S   0.0  0.0  7:33.77 /usr/lib/accountsservice/accounts-daemon
795 messagebu 20   0  8612   4744   3680 S   0.0  0.0  29:57.81 /usr/bin/dbus-daemon --system --address=systemd: --nofor
825 root       20   0 19252   9424   7412 S   0.0  0.0  20:25.33 /lib/systemd/systemd-logind
904 root       20   0 1455M  25164  11500 S   0.0  0.0  1:37.05 /usr/bin/containerd
1129473 ins       20   0 14948   6144   4564 S   0.0  0.0  0:01.48 sshd: ins@pts/2
433 root       19  -1 86080  39864  37100 S   0.0  0.1  22:09.69 /lib/systemd/systemd-journald
464 root       20   0 22472   5608   4024 S   0.0  0.0  0:07.49 /lib/systemd/systemd-udev
684 root       RT    0  274M  18828   9032 S   0.0  0.0  0:00.00 /sbin/multipathd -d -s
685 root       RT    0  274M  18828   9032 S   0.0  0.0  0:04.22 /sbin/multipathd -d -s
686 root       RT    0  274M  18828   9032 S   0.0  0.0  3:40.85 /sbin/multipathd -d -s
687 root       RT    0  274M  18828   9032 S   0.0  0.0  0:00.00 /sbin/multipathd -d -s
688 root       RT    0  274M  18828   9032 S   0.0  0.0  0:00.00 /sbin/multipathd -d -s
681 root       RT    0  274M  18828   9032 S   0.0  0.0  5:53.49 /sbin/multipathd -d -s
772 systemd-n 20   0 19824   7996   7016 S   0.0  0.0  0:36.38 /lib/systemd/systemd-networkd
774 systemd-r 20   0 25240  13472   9228 S   0.0  0.0  0:09.57 /lib/systemd/systemd-resolved
708 root       20   0  231M  8656   6476 S   0.0  0.0  2:20.74 /usr/lib/accountsservice/accounts-daemon
```

28. Detener un comando en ejecución

Si un comando o script no nos responde o simplemente queremos detenerlo podemos presionar las teclas Ctrl y la letra C simultáneamente para detener la ejecución.

29. Imprimir el historial de comandos que hemos ejecutado

Si necesitamos un registro de los comandos que hemos ejecutado podemos recurrir al comando history

```
history
```

30. Caracteres especiales de bash

Linux es sensible a lo que escribamos en mayúscula y minúscula. Adicionalmente, algunos caracteres están reservados por el sistema y son de uso frecuente. Los comodines, atajos a directorio home y parental, y otros símbolos nos ayudan a ser más productivos.

Caracter	Descripción	Caracter	Descripción
~	Directorio home	.	Directorio actual
..	Directorio parental	#	Comentario
\$	Variable	&	Trabajo en background
*	Comodín	?	Comodín de una letra
(Inicio sub-shell (anidar comandos))	Fin sub-shell (anidar comandos)
\	Comillas al siguiente caracter		Pipe (el stdout será entrada del siguiente comando)
;	Separador de comandos	!	Negación (No lógico)
>	Redirigir stdout	>>	Agregar stdout a archivo