Herramientas de Programación en Bioinformática y Biología Computacional

Resumen

Esta es una asignatura introductoria de los conceptos más importantes que se irán extendiendo a lo largo de todo el máster. Se divide en tres módulos: Linux, Python y Bases de Datos.

Sandra Mingo Ramírez

UAM - 2024/25

3 de septiembre de 2024 14:01

Universidad Autónoma de Madrid
Bioinformática y Biología Computacional

Código en Github

Índice general

Capítulo I

Linux

Todos los comandos se lanzan desde la terminal de Linux. Se pueden explorar los ficheros del árbol de directorios con un explorador que es equivalente al explorador de Windows. En cuanto se maneje con ficheros grandes, o se requieran acciones mínimas, las herramientas de la interfaz gráfica no van a ser suficientes - de ahí que tengamos que trabajar con la terminal.

man

Con el comando `man comando`, se accede al manual para buscar el comando con su nombre, descripción y posibles parámetros tanto en su versión larga como corta.

I.1. Tratamiento de ficheros y directorios

ls

El comando `ls` muestra la lista de la información del directorio en el que nos encontremos. Muestra tanto los ficheros como los directorios. En Ubuntu, distingue los directorios en azul oscuro y los ficheros en verde, pero el código de colores puede cambiar en la configuración y puede no ser igual siempre, por lo que no nos deberíamos fijar en eso. El comando se puede completar con una serie de **parámetros detrás de un guion que modifican el comportamiento del comando inicial**. `ls -l` lista el contenido del directorio con información adicional que incluye la localización del fichero, el tipo de fichero, el tamaño en bytes, los permisos, el inode y cuándo se modificó por última vez. Algunos tamaños pueden ser difíciles de leer a simple vista. Para ello, se emplea `ls -lh` que lista el contenido con el tamaño largo, de forma que indica las unidades de forma más comprensible a simple vista. De hecho, la h viene de "human". El parámetro `-t` ordena la lista por orden de creación o modificación de más reciente a más antiguo, y `-p` añade una barra invertida a los directorios para poder distinguirlos de los ficheros. Si se escribe `-lp`, la primera columna tiene una d al inicio de todos los directorios, mientras que los ficheros tienen un guion como primer caracter. Así, todos los parámetros se pueden concatenar en `ls -lhtp`. El parámetro `ls -a` lista el contenido del directorio en el que nos encontremos, incluyendo los ficheros ocultos cuyo nombre empieza por un punto (por ejemplo, ".config"). Estos ficheros no aparecen a no ser que se añada el parámetro a para protegerlos y que no se modifiquen sin querer. "Si no quieres que alguien meta la pata, no le enseñes dónde puede meter la pata". En función de los permisos que se tenga, luego se pueden modificar esos ficheros o no. El parámetro `-i` añade un inode que será importante cuando seamos administradores,

ya que es un índice que único para cada fichero que permite modificar un fichero sin modificar otro aunque se llamen igual. Aunque no se puedan tener dos ficheros en el mismo directorio que se llamen igual porque el path sería el mismo, sí puede haber dos ficheros con el mismo nombre en directorios distintos, ya que el path sería diferente, al igual que el inode.

mkdir

Para crear nuevas carpetas o directorios, en el explorador es click derecho y crear nueva carpeta. En la terminal, esto se traduce en el comando `mkdir nombre-carpeta-nueva`. El tamaño de un directorio no es el tamaño de lo que tenga dentro el directorio; al crear una carpeta nueva, ya tiene un tamaño de 4K, y si se tuvieran muchos ficheros muy pesados, no tiene por qué ser la suma de todos los tamaños que contenga ese directorio.

cd

Para cambiar de directorio, se emplea el comando `cd nuevo-directorio` Así, el siguiente prompt incluye el directorio en el que se encuentra. Al entrar en un directorio que acabamos de crear, con el comando `ls -lht` no aparece nada, pero con `ls -lhta` aparecen dos directorios ocultos llamados "." y "..". Los dos puntos son un **enlace al directorio anterior o padre**, de forma que con `cd ..` se va al directorio anterior. El punto simple es el **directorio actual**.

pwd

El comando `pwd` muestra dónde se encuentra el **directorio actual**, es decir, el **directorio de trabajo o working directory**. El primer símbolo es una barra (/) que indica el **directorio raíz**. La virgulilla (∼) lleva a la "carpeta personal", que es una traducción de **"home directory"**. Se puede mover entre directorios ya sea por path directo mediante el uso del directorio raíz o mediante el árbol del directorio con los dobles puntos o la virgulilla.

cp

Para copiar un archivo, se emplea el comando `cp archivo-original archivo-nuevo`. Hay comandos que no necesitan **argumentos**, y hay comandos (como este) que sí necesitan. En este caso, requiere de dos argumentos: el archivo que se quiere copiar y el nombre que se le quiere dar a la copia. El orden es siempre **comando -parámetro argumento**. Si se quiere copiar un directorio con todo su contenido a otro directorio, es necesario añadir el parámetro `cp -r directorio-original directorio-nuevo` (r de recursivo, es decir, todo lo que haya dentro). La fecha de los ficheros copiados es de cuando se crea la copia, no la del fichero original. Solo se podrán copiar archivos en aquellos directorios en los que tengamos los permisos apropiados por el administrador. Antes de realizar el comando para copiar, la terminal realiza una serie de comprobaciones, de forma que si el nombre del archivo a copiar tiene alguna errata, salta un error al no poder realizarse el comando stat, que forma parte de las comprobaciones previas. Si se copia un directorio en otro preexistente, se copia el primer directorio en el segundo. Esto resulta en que el segundo directorio incluye los ficheros que ya estaban y un nuevo directorio nuevo dentro - no se sobreescribe todo el directorio ni se borra lo que hubiese ahí. Aun así, sigue siendo recomendable proteger los directorios sensibles añadiendo un punto antes del nombre o directamente cambiando los permisos.

mv

Para mover un archivo, se utiliza el comando `mv`. Se utiliza de forma similar a cp, pudiendo mover un fichero a otro directorio o al mismo con el mismo u otro nombre siempre que el nuevo directorio ya exista previamente. Si se mueve un directorio entero, se mueve con todo el contenido que tenga en el nuevo directorio, y también se le puede cambiar el nombre.

rm
rmdir

A la hora de querer borrar, se utiliza `rm` para los archivos y `rmdir` para los directorios. Para esto último, es necesario que el directorio esté vacío. En caso de que no, se puede utilizar `rm -r directorio/` o `rm -r ./directorio/` para borrar tanto el directorio como su contenido.

ln

En lugar de copiar un fichero y tenerlo doble, se puede crear un link simbólico. Esto quiere decir que se crea un fichero que no tiene el contenido del fichero original (y por tanto pesa menos, solo 11 bytes), si no que redirige a ese archivo, permitiendo enlazar distintos directorios o ficheros en ordenadores en remoto. Para crear un link simbólico, el comando es `ln -s fichero-original fichero-simbolico`. Si se borra el fichero original, el enlace simbólico ya no tiene utilidad. Para borrar un enlace, se puede borrar como un fichero cualquiera con `rm`.

> Todos los ficheros tienen un nombre y una extensión, y siempre se debe escribir completo. Los directorios no llevan nunca extensión. Se recomienda no utilizar espacios en blanco en los nombres tanto de los ficheros como de los directorios porque dificulta la navegación.

I.2. Mostrar contenidos en pantalla

clear

El comando `clear` borra los comandos de la pantalla para dejarla vacía.

cat

Para mostrar en pantalla el contenido de un fichero, se puede emplear el comando `cat fichero`. No obstante, este comando no diferencia cambio de página, si no que muestra todo el contenido por pantalla aunque ocupe más. También se puede utilizar este comando con varios ficheros para que se muestren uno detrás de otro.

less

Para mostrar el contenido de un fichero en la pantalla por partes, se emplea `less`. De esa forma no hay que subir y bajar en la pantalla, si no que se desplaza mediante el uso del espacio para ir por páginas enteras, enter para ir línea a línea, y la tecla q para salir (quit).

head
tail

El comando `head` muestra en pantalla las primeras 10 líneas de un fichero por defecto. Se le puede añadir un parámetro para indicar el número de líneas que se desea ver: `head -15 fichero` muestra las primeras 15 líneas. De forma similar, `tail` muestra las últimas 10 líneas de un fichero.

I.3. Buscar contenidos de ficheros

grep

El comando `grep` es fundamental al utilizarse mucho con los pipes y las redirecciones (véase más abajo). Se utiliza `grep cadena-buscada fichero`, y el resultado son todas las líneas que incluyen la cadena de caracteres que se ha buscado, tanto como palabra suelta como dentro de otra palabra. Utilizando `grep -w`, se filtran aquellas líneas en las que la cadena forma palabras individuales (si se busca with, descarga within, por ejemplo). Este comando es sensible a las mayúsculas y minúsculas. Para ignorar eso, se añade el parámetro `-i`. El parámetro `-v` muestra aquellas líneas

que no incluyan la cadena, `-n` incluye al inicio de cada línea el número de dicha línea, y `-c` el número de líneas que se deberían mostrar.

wc

El comando `wc` cuenta distintas cosas en un fichero: líneas, palabras, y bytes. Para obtener solo una de esas mediciones, se pueden añadir los parámetros `-l, -w, -c` respectivamente. Tradicionalmente, la cantidad de bytes indica la cantidad de caracteres, pero puede haber pequeñas diferencias por la codificación. Para obtener la cantidad de caracteres, se emplea `grep -m`.

I.4. Redireccionamientos y pipes

Tras ejecutar un comando, el resultado que sale en pantalla es la **salida estándar**. Sin embargo, se puede **redireccionar la salida** a otro lugar mediante el símbolo >.

> > >

Linux trata igual la salida estándar y un fichero, pero la salida estándar es la opción por defecto, por lo que nosotros debemos redireccionar la salida cuando lo queremos guardar en un fichero. Hay que tener cuidado porque si se redirecciona una salida a un fichero que ya existe, se sobreescribe el contenido del mismo. Si lo que queremos es que el contenido nuevo se añada detrás del contenido ya presente en un fichero, se deben emplear los dos símbolos > >.

También existe un **redireccionamiento de entrada**, siendo la entrada estándar

<

lo que se escribe por teclado. Para redireccionar la entrada, se emplea el símbolo <.

gedit

El comando `gedit fichero` abre un editor de texto plano que nos permite crear un fichero y escribir su contenido. Para guardar el contenido que hemos escrito, pulsa `ctrl s` y para cerrar `ctrl q`. Aunque el editor de texto también muestre el contenido de un fichero que ya existe previamente, si el tamaño del fichero es muy grande, no es recomendable leer su contenido mediante este comando, si no mediante otros mencionados previamente como `less`.

sort

El comando `sort` ordena una entrada por teclado o un fichero alfanuméricamente. Si un fichero está separado en líneas, utiliza el primer caracter de cada línea. A este comando se le puede pasar directamente ficheros o se le puede redireccionar por entrada.

|

Hay ocasiones que para obtener un resultado paso a paso, hay que crear ficheros temporales que modificar y a los que acceder. Esto se puede evitar mediante el uso de **pipes** o barras verticales (|), que conectan los comandos antes y después del pipe. De esta forma, se omite el paso intermedio y se utiliza la salida del primer comando como entrada del segundo.

I.5. Wildcards y convenciones en nombres de archivo

*
?

Los wildcards en informática son como comodines.Existen dos tipos: el asterisco y el símbolo de interrogación. El **asterisco** va a representar cualquier número de caracteres en el nombre de un fichero o directorio. Por ejemplo, si ponemos `prueba*`, puede resultar en el directorio pruebas y en los ficheros prueba.txt y prueba.pdf. Por el

contrario, el **interrogante** representa exactamente un caracter. Así, al poner `prueba?`, el resultado será exclusivamente el directorio pruebas, pero no los ficheros.

A la hora de nombrar ficheros y directorios, se deben **evitar los símbolos especiales** tales como /, *, & y %. También es muy recomendable **evitar espacios entre caracteres**. En resumen, a la hora de poner un nombre a un archivo, se deberían usar solo caracteres alfanuméricos junto a barras bajas y puntos. Tradicionalmente, los nombres empiezan en minúcula y pueden terminar en un punto seguido de un grupo de letras que indican el contenido de un archivo (por ejemplo, poner al final de todos los archivos en código Python .py).

I.6. Seguridad del sistema de archivos (permisos de acceso

Cada archivo y directorio tiene **permisos de acceso asociados**, que se pueden comprobar en la primera columna al realizar `ls -lht`. Se trata de una cadena de 10 símbolos d, r, w, x, -. La d solo estará presente en primera posición e indica que se trata de un directorio. Si se trata de un fichero, en primera posición hay un guion. Los 9 símbolos restantes se agrupan en grupos de 3 en 3 y representan los **permisos del usuario, del grupo al que pertenece el usuario (y no es el usuario) y de todos los demás** respectivamente. Las opciones son:

- r (read): permisos de lectura y copiado de un archivo y de listar el contenido de un directorio.
- w (write): permisos de escritura y modificado de un archivo y de crear y borrar los archivos del directorio o mover archivos a él.
- x (execution): permisos de ejecución de un archivo cuando sea apropiado, es decir, cuando sea ejecutable. Por ejemplo, los comandos de linux como ls o mv son ficheros que se encuentra en /usr/bin/ y que se pueden ejecutar por todos a la hora de escribir esos comandos en la terminal.

chmod

Si en lugar del caracter aparece un guion, significa que ese permiso no está dado. Si eres el propietario de un fichero, se pueden cambiar los permisos mediante el comando `chmod`. Para indicar a quién se le quiere cambiar el permiso, se pone u (usuario), g (grupo), o (otros), a (todos). Para quitar un permiso, se escribe un guion (-) seguido del permiso que se quiere quitar y del fichero; para dar un permiso, se escribe un más (+) y el permiso seguido del nombre del fichero. Un igual (=) expresa los permisos que se quieran. Para poner permisos diferentes, se escriben separados por comas (pero sin espacios), por ejemplo `chmod g+wx,o+w fichero`. También se puede ejecutar el comando `chmod 777 fichero` para darle todos los permisos a todos. Es importante tener en cuenta también los permisos de los directorios, ya que para poder modificar un fichero, no solo se deben tener los permisos para modificarlo, si no que a su vez debe estar en un directorio para el que se tengan los permisos para modificarlo.

I.7. Procesos y trabajos

Un **proceso** es un programa que se ejecuta y que recibe un **identificador (PID)** por parte del sistema operativo. Hay procesos que los lanzamos nosotros, pero hay otros que funcionan por detrás como por ejemplo buscar conexiones Wifi.

top

El comando `top` muestra todos los procesos que se están llevando a cabo en tiempo real. La terminal es un proceso, y cada vez que nosotros lanzamos un proceso desde la terminal, por lo que se genera un proceso con un proceso padre (la terminal). Los procesos no pueden ejecutarse a la vez (van secuencialmente), pero el sistema operativo va gestionando los procesos y sus tiempos para que parezca que van en paralelo. Los sistemas operativos modernos pueden contener varios núcleos o kernel que permite su funcionamiento a la vez, pero dentro de cada núcleo o kernel los procesos van de forma secuencial. Para salir de top, se debe ejecutar `ctrl c`.

ps

El comando `ps` muestra todos los procesos que se han lanzado desde esa terminal. Añadiendo el parámetro `-ef`, se incluyen los procesos padres (lanzados por root) con el usuario que lanzó el proceso, el PID y el PID padre.

sleep

El proceso `sleep` está parado el tiempo que se le indique. En realidad, va a estar más tiempo debido a la gestión del tiempo del sistema operativo que va alternando distintos procesos de forma intermitente. Por ello, aunque ese proceso sí dure el tiempo determinado, el usuario tiene que esperar algo más. Si se escribe `sleep 10s &`, se estará ejecutando de fondo (en el background), y lo que se devuelve a la terminal es el PID. Esto permite seguir utilizando la terminal para otros procesos mientras tanto. Si se va enviado un proceso en el foreground cuando se quería en el background, se puede abortar (con ctrl c) y volver a ejecutar bien o se puede detener (con ctrl z) y poner bg para enviarlo al background. Si por el contrario se quiere reiniciar un proceso suspendido, se pone `fg`. Esto reinicia el último proceso suspendido. Para especificar uno concreto, se debe poner su número de trabajo (no el PID): `fg $1`.

kill

Para matar un proceso de forma eficiente, se puede usar el comando `kill -9` seguido del identificador del proceso. De esta forma se asegura que también se eliminen los subprocesos generados por ese proceso y se cierren todos los recursos. Sin embargo, no es posible matar los procesos de otros usuarios.

I.8. Otros comandos útiles de UNIX

- **df**: informa del espacio libre del sistema.
- **du**: saca los kilobytes utilizados por cada subdirectorio. Esto puede ser útil cuando se ha superado el almacenamiento para ver qué directorio tiene más archivos.
  - -s: muestra solo un resumen
- **gzip y gunzip**: `gzip` reduce el tamaño de un fichero utilizando el compresor Zip, resultando en un fichero con extensión .gz. `gunzip` descomprime el fichero .gz. Para esto, es necesario tener permisos de lectura.
- **tar**: combina varios archivos en uno único que puede o no estar comprimido.

  - -c: crea un archivo.
  - -x: extrae un archivo.
  - -v: muestra el progreso de un archivo.
  - -f: nombre de archivo.
  - -t: ver el contenido del archivo.
  - -j: comprime el archivo mediante bzip2.
  - -z: comprime el archivo mediante gzip.
  - -r: añade o actualiza archivos o directorios a archivos existentes.
  - -C: directorio especificado
  - –exclude=: excluye los archivos que vayan a continuación de la orden principal.
- **zcat**: lee archivos de extensión .gz sin la necesidad de descomprimirlos previamente.
- **cut**: extrae porciones de texto de un fichero al seleccionar columnas o caracteres.
  - -d: delimitador en comillas simples
  - -f: campo
- **tr**: reemplaza o borra caracteres.
- **date**:
- **file**:
- **stat**:
- **basename y dirname**:
- **diff**:
- **sort**:
- **find**:
- **xargs**:
- **history**:
- **ssh**:
- **nohup**: