Básicos UNIX y conda

Ph D. Stephanie Hereira Pacheco

2023-02-13

¿Qué necesitaremos para la clase de hoy?

Para la clase de hoy vamos a necesitar:

- Tener acceso a una terminal, los usuarios de linux y macOS ya cuentan con ella, para los usuarios de windows preferiblemente VirtualBox.
- Tener instalado conda, para esto depende de tu sistema operativo:

```
#macOS
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh -O ~/miniconda.sh
```

```
bash ~/miniconda.sh -b -p $HOME/miniconda
```

```
#Linux
bash Miniconda3-latest-Linux-x86_64.sh
```

```
bash Anaconda-latest-Linux-x86_64.sh
```

¿Qué es UNIX? - Generalidades

¿Qué es UNIX?

- Es un sistema operativo desarrollado por Ken Thompson y Dennis Ritchie en 1969.
- Fue desarrollado con el objetivo de escribir programas para una sola tarea y que colaboren entre sí además de manejar un flujo de texto que pueda funcionar en una interfaz universal.
- Hasta 1991 era reservado para las supercomputadoras. Luego se implementó un equivalente de libre acceso para Linux (GNU/Linux) y otro basado en él que fue el de macOS.
- Algunos conceptos:
  - Kernel: núcleo o corazón del sistema operativo, y se encarga principalmente de mediar entre los procesos de usuario y el hardware disponible en la máquina, es decir, concede el acceso al hardware, al software que lo solicite, de una manera segura. Encargado de controlar el acceso a todos los componentes.
  - Shell: Es la interfaz para que el usuario pueda interaccionar con los diferentes elementos y poder hacer uso de una computadora y sus programas. Encargo de gestionar los comandos y sus respuestas (Intérprete de comandos): Bourn shell (sh), bash shell (shell) y Z shell (zsh).

Sistemas de archivos, terminal y códigos.

Sistemas de archivos

- UNIX propone una organización de datos vía archivos y **directorios**.
- La raíz del sistema es "/" llamado "root". Cualquier directorio o archivo será refernciado a partir de esta raíz.
- Varios ejemplos de directorios comunes son: >- `/etc` que contiene los archivos referentes a las configuraciones de los programas >- `/bin` y `/usr/bin/` contienen los archivos binarios de uso común. >- `/home` que contiene los directorios del usuario. >- `./` ó `.` la ubicación de nuestro directorio actual. >- `../` la ubicación del directorio anterior. >- `../../` la ubicación del directorio dos ubicaciones anteriores.
- Si queremos saber donde estamos usamos el comando `pwd`
- Bash shell es amigable, con las flechas permite ver códigos anteriores y con TAB autocompletar códigos, además de permitir trabajos de fondo.

Terminal

- La Terminal es una consola que nos permite ejecutar todo tipo de **códigos** mediante **comandos** (secuencias de palabras o instrucciones).
- Nos muestra en qué directorio nos encontramos y si estamos en un ambiente de conda o no (Esto lo veremos más adelante con mpas detalle).

Terminal

- `PATH` es una variable del entorno que almacena una lista de directorios en los cuales Linux debe buscar un comando para poder ejecutarlo.
- Una variable de entorno es una variable que es definida en todo el entorno de ejecución de la shell.
- Esto es muy importante porque a veces no sabemos que debemos colocar los programas en el `PATH` para poder ejecutarlos o ubicarnos en el directorio para poder correrlos.

Terminal - Cambiando el PATH

##Cambio temporal

```
export PATH="$HOME/bin:$PATH"
```

```
nano ~/.bashrc
#colocamos la línea dentro del archivo
export PATH="$HOME/bin:$PATH"
```

#Comandos

Los comandos se ejecutan de la siguiente manera:

```
comando [opciones] [argumentos]
```

- **Opciones**: Cambia el funcionamiento orifinal del comando.
- **Argumento**: Información que necesita el comando para realizar su tarea.

Corramos el siguiente comando:

```
ls -l
```

Ahora este:

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

En caso de pánico podemos usar `Ctrl` + C

Otros comandos...

cd

Este comando nos permite movernos entre directorios "change directory".

```
cd ../ #Referencia relativa
```

```
cd /home/usr/Documents #referencia absoluta
```

```
cd #regresamos al HOME
```

Otros comandos...

ls, mkdir, rmdir

```
ls  #lista el contenido de nuestro directorio
```

```
mkdir Nuevo_directorio #crear un directorio
```

```
rmdir Nuevo_directorio #crear un directorio
```

```
rm -r  Nuevo_directorio #crear un directorio
```

Otros comandos...

COMANDO	FUNCIÓN
`cp`	Hacer una copia del directorio o archivo
`mv`	Mover un directorio o archivo / renombrar
`cat`	Visualizar o listar e imprimir elementos de archivo
`zcat`	Visualizar o listar e imprimir de archivo de extensión .gz
`less`	Visualizar arhivo de a poco
`head`	Visualizar las primeras 10 líneas
`tail`	Visualizar las últimas 10 líneas
`wc`	Cuenta palabras
`sudo`	Permite correr programas con permisos de superusario
`df`	Espacio de disco en el sistema
`du`	Espacio que ocupa un archivo en el sistema
`unique`	Elimina líneas sucesivas repetidas

Otros comandos...

COMANDO	FUNCIÓN
`history`	Despliega el historial de comandos
`ln -s`	Crear una liga simbólica
`clean`	Borra comandos en tu ventana
`wget`	Descagar archivos de internet
`ps/ps aux`	Procesos activos del sistema
`kill`	Matar un proceso activo
`gzip/gunzip`	Comprimir o descomprimir un archivo .gz
`tar`	Comprimir y descomprimir archivos de extensión .tar
`grep/awk`	Buscar un patrón dado en un archivo
`find -name`	Buscar un archivo en mis directorios
`sort`	Ordena alfabéticamente líneas de un archivo
`cut`	Suprime partes de líneas de un archivo

Permisos de archivo

Al poner este comando en nuestra terminal podemos ver :

```
ls -l
```

Un archivo puede tener derechos de: - lectura (r) - escritura (w)

Para un directorio puede tener derechos de: - lectura (r) - escritura (w) - entrada (x) / ejecución (programas)

#Cambiando permisos : chmod

- Añadir permisos de ejecución para el usuario

```
chmod u+x
```

- Añadir permisos de ejecución para el grupo y demás

```
chmod g+w
chmod o-x
```

- Añadir permiso a programa de ejecutarse

```
chmod +x script.sh
```

- Añadir permisos de lectura total y totales

```
chmod 755 archivo
chmod 777 archivo
```

```
0 =  = sin acceso
1 = --x = ejecución
2 = -w- = escritura
3 = -wx = escritura y ejecución
4 = r-- = lectura
5 = r-x = lectura y ejecución
6 = rw- = lectura y escritura
7 = rwx = lectura, escritura y ejecución
```

WILCARD *

El wildcard o comodín es el más conocido y útil, representa cualquier combinación posible, es decir, **todo**. Por ejemplo si ponemos:

```
cp * /home/
```

Estamos indicando que queremos copiar todos los elementos del directorio donde nos encontramos para el HOME.

REDIRECCIÓN > Y »

Los caracteres > y » permiten redirigir la salida estándar. Podemos redirigir la saida a un archivo de la siguiente manera:

```
ls > mi_lista.txt
```

Permite hacer un nuevo archivo o reescribir uno existente:

```
ls >> mi_lista.txt
```

USO DE PIPE |

- El concepto Pipe procede de la palabra inglesa **"pipeline"**, cuya traducción es **"tubería"**.
- En informática, un pipe es un flujo de datos que circula entre dos procesos que, o bien están estrechamente vinculados, o no tienen un origen común. Esto quiere decir que el resultado arrojado por un programa servirá como entrada para otro programa.

```
Comando-1 | Comando-2 | ...| Comando-N
```

Ejemplo:

```
cat archivo.txt |wc -l
```

```
grep "^>" dna-sequences.fasta | wc -l
```

¿Qué es conda?

De acuerdo a la definición oficial:

  - conda: A package manager helps you find and install packages.
  - conda: Un administrador de paquetes que ayuda a encontrar e instalar paquetes.

- Se obtiene mediante Anaconda o Miniconda.
- La instalación dependerá del sistema operativo a utilizar.
- Aunque conda está disponible para Windows, macOS y Linux, no todos los paquetes disponibles para bioinformática son compatibles entre los tres sistemas, siendo la prioridad de paquetes Linux > macOS y en algunos casos, al final, Windows.
- Funciona para casi todos los lenguajes.
- originalmente orientado a usuarios de Python

¿Por qué Conda?

- Es fácil de usar
- Es multiplataforma
- Se lleva bien con Python

- Maneja entornos o ambientes

Entornos virtuales o ambientes

- Es una ubicación del PATH, aislada del sistema, que previene conflictos de versiones.

Conda

- Instalando conda

```
bash Miniconda3-latest-Linux-x86_64.sh
```

- Revisando versión

```
conda info
```

- Ambientes creados

```
conda env list
```

Conda

- Agregar canales

```
conda config --add channels r
conda config --add channels conda-forge
conda config --add channels bioconda
```

- Creando entorno

```
conda create -n test3.6 python=3.6 pandas=1.0
```

Removiendo un entorno

```
conda remove -n qiime2 --all
```

Conda

mamba

```
conda install -yc conda-forge mamba

mamba create -y -n testpy3.6 python=3.6 pandas=1.0

mamba create -y -n testR3.6 r-base=3.6

mamba create -n picrust2 -c bioconda -c conda-forge picrust2=2.5.0
```

Conda

- Activar un ambiente o conda

```
conda activate qiime2
```

- Desactivar un ambiente o conda

```
conda deactivate qiime2
```

Referencias

- https://mamba.readthedocs.io/en/latest/user_guide/mamba.html
- https://comunidadbioinfo.github.io/cdsb2022/introducci%C3%B3n-a-conda..html
- https://omicstutorials.com/getting-started-in-linux-bioinformatics/
- https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html