Instalación de aplicaciones

Estándar de jerarquía del sistema de archivos

Antes de instalar software no administrado por nuestra distribución conviene que tengamos una idea de como se organiza el sistema de archivos en Linux para que entendamos dónde se instala el software. Linux sigue una norma estándar llamada Estándar de jerarquía del sistema de archivos que define los directorios del sistema y la localización de los distintos tipos de archivos.

En Unix todos los archivos y directorios aparecen bajo el directorio raíz (/). Es habitual que dentro del directorio raíz existen tres jerarquías en las que se distribuyen los archivos y directorios de los programas:

- /,**jerarquía primaria**. De ella cuelgan el resto de jerarquías. Los archivos incluidos directamente en esta jerarquía son sólo los esenciales para el sistema, como por ejemplo los comandos: *cp*, *ls* o *mkdir*.
- /usr/, **jerarquía secundaria**. Contiene la mayoría de aplicaciones del sistema. En las distribuciones Linux es la jerarquía en las que los programas de uso común son instalados, como por ejemplo el LibreOffice o el entorno de usuario Gnome.
- /usr/local/, **jerarquía terciaria**. Contiene la mayoría de las aplicaciones que instalamos sin la mediación de la distribución.

Dentro de cada una de las jerarquías hay varios directorios en los que se distribuyen los archivos de las aplicaciones dependiendo del tipo de archivo. Por ejemplo, los ejecutables se encuentran en los directorios *bin* y las librerías en *lib*. En mi ordenador el ejecutable *cp*, que es esencial para el sistema se encuentra en */bin/cp*, el editor de textos *gedit* que ha sido instalado por la distribución y no es esencial para el sistema se encuentra en */usr/bin/gedit* y el alineador de secuencias *bwa* que yo he instalado manualmente sin utilizar el gestor de paquetes se encuentra en */usr/local/bin/bwa*.

A diferencia de otros sistemas operativos en los sistemas Linux las aplicaciones normalmente no están contenidas en un sólo directorio. Los ejecutables están en *bin*, las librerías de los que dependen en *lib*, etc. Cuando se instalan programas estáticos, que incluyen todas sus librerías, suelen instalarse en */opt/*. *opt/* sigue un modelo similar al *Archivos de programa* de *Windows*. No es muy común que los programas se instalen de esta forma en *Linux*, pero algunos programas como el *IDE java eclipse* sí suelen instalarse en /opt

Instalando programas

README e INSTALL

Cuando queramos instalar aplicaciones que no esten en repositorios o que la aplicacion que queremos está pero necesitamos una version más nueva, tendremos que ir programa por programa instalandolo. No existe una sola forma de instalarlos, pero si que todos los programas suelen tener algun tipo de documento en el que se explica como instalar el programa. Normalmente este documento suele llamarse README o INSTALL, aunque no hay ninguna regla que diga que esos son los nombres, por lo que lo primero que tenemos que hacer cuando nos descargemos la aplicacion es buscar este archivo o alguno en el que intuyamos en el que pueda estar la informacion acerca de como instalarlo.

Instalando programas compilados

Una vez hemos visto dónde debemos instalar los programas que administremos sin la ayuda de la distribución (en */usr/local*) vamos a ver como instalaríamos el mapeador de secuencias cortas bowtie. En su página de descargas encontramos los siguientes ficheros:

- bowtie-1.1.2-src.zip
- bowtie-1.1.2-macos-x86_64.zip
- bowtie-1.1.2-linux-x86_64.zip
- bowtie-1.1.2-mingw-x86_64.zip

Los ficheros de interés en un sistema Linux son los marcados como linux-x86_64 y src. El primeros incluyen el programa compilado y el segundo el código fuente listo para compilar. Normalmente si se nos ofrece el programa precompilado para nuestra arquitectura podemos simplemente copiar el programa a */usr/local*. En este caso el programa lo han compilado para la arquitecturas x86_64. x86_64 se refiere a los microprocesadores *Intel* y *AMD* de 64 bits.

Esta arquitectura no tiene que coincidir con nuestro procesador sino con la versión de la distribución que tenemos instalada. Por ejemplo, Ubuntu y Debian tienen versiones para 32 y 64 bits, pero en un ordenador con un microprocesador de 64 bits podemos optar por instalar la versión de la distribución de 32 bits. La arquitectura del programa compilado debe coincidir con la de la distribución que tengamos instalada, no con la de nuestro microprocesador. Para poder saber la arquitectura de nuestro sistema operativo, podemos usar el comando `uname`:

```
~$ uname -a
  Linux txindoki 4.6.0-1-amd64 #1 SMP Debian 4.6.4-1 (2016-07-18) x86_64 GNU/Li
```

Supongamos que queremos instalar la versión de 64 bits del programa precompilado. Una vez que hemos descargado el fichero zip (que en muchas ocasiones será en realidad un tar.gz), lo descomprimimos. En el directorio descomprimido encontramos los ejecutables, en este caso el *bowtie* y otros archivos adicionales, en este caso *scrips* en el directorio *scripts* y ficheros de ejemplos en los directorios *indexes*, *genomes* y *reads*. Es posible que entre todos estos ficheros haya un *README* o un *INSTALL* si es así debemos leerlos antes de continuar.

La forma más sencilla de instalar el programa es mover el directorio completo a */usr/local/bowtie/*. Una vez hecho deberemos añadir los directorios en los que hay ejecutables a nuestro $PATH. Sino lo hacemos no podremos ejecutar el programa a no ser que incluyamos la ruta completa al ejecutables */usr/local/bowtie/bowtie*.

Para incluir el directorio con los ejecutables en la variable *PATH* debemos modificar la variable con la siguiente orden::

```
~$ export PATH=$PATH:/usr/local/bowtie/:/usr/local/bowtie/scripts/
```

Una vez ejecutada esta orden ya podríamos ejecutar el bowtie en la terminal. El problema de este método es que cada vez que nos salimos de la sesión del terminal esta modificación del $PATH se pierde. Para que el $PATH sea el que deseamos siempre que entremos al sistema el comando anterior debe ser ejecutado, esto podemos conseguirlo incluyéndolo en el fichero *.bashrc* situado en nuestra $HOME. *.bashrc* se ejecuta automáticamente cada vez que entramos en un *shell* y se utiliza para adaptar las variables de entorno, como por ejemplo $PATH, a nuestras necesidades.

Ejercicions: 1. Instala la ultima version de ncbi-blast+ 2. Instala la ultima version de tophat

Instalando un programa desde el código fuente

Cuando no tenemos la posibilidad de instalarnos el programa ni desde la distribuicion ni usando binarios compilados, podemos optar por instalar un programa partiendo desde el código fuente, para ello primer tenemos que compilarlo. Compilar significa traducir el codigo en texto escrito por humanos en un codigo que el ordenador entienda; codigo binario. Este suele ser un caso bastante habitual en las aplicaciones bioinformáticas, para los casos en los que los desarrolladores no han creado binarios compilados para nuestra arquitectura.

Normalmente los ficheros con el código fuente con ficheros comprimidos *tar.gz*, pero en el ejemplo que nos ocupa es un fichero comprimido *zip*. Una vez descomprimido normalmente encontraremos un fichero *INSTALL* o *README* si es así debemos leerlos antes de continuar.

El caso más habitual es que entre los archivos descomprimidos haya un *shell script* ejecutable llamado *configure*. Si es así el procedimiento a seguir suele ser ejecutar la secuencia de comandos *./configure*, *make*, *make install*. *configure* verificará que nuestro sistema dispone de todos las librerías y utilidades necesarias para instalar el programa y creará una serie de ficheros *Makefile*. Una vez terminado el *configure* y creados los *Makefile* satisfactoriamente podemos ejecutar el comando *make* y el programa se compilará para nuestra arquitectura. El *configure* también suele ser el encargado de determinar en qué jerarquía del sistema de archivos se instalará el programa. Lo habitual es que estos programas compilados manualmente se instalen en la tercera jerarquía, */usr/local/*.

Para poder compilar los programas necesitamos tener instaladas una serie de aplicaciones. En Ubuntu por defecto estas aplicaciones no se instalan en el sistema. Para instalarlas debemos instalar el paquete build-essential::

```
~$ sudo apt-get install build-essential
```

Este comando instalará, entre otras cosas el compilador *gcc* y los *headers* de las librerías básicas del sistema. Los *headers* se encuentran en los paquetes *dev* (*devel* en *RedHat* y derivados) y son necesarios para poder compilar programas que requieran las librerías. Para compilar un programa que utilice una librería no es suficiente con tener instalada la librería, necesitamos instalar el paquete *dev* con los *headers*. Si el *configure* se queja de que alguna librería no está instalada lo más habitual es que no tengamos el correspondiente paquete *dev*instalado.

Una vez tenemos todas las librerías requeridas instaladas el compilador podrá compilar el programa cuando ejecutemos el *make*. Este proceso debe terminar sin errores, si algo falla debemos leer la salida del *make* y buscar el fallo. Normalmente el fallo suele deberse a la falta de alguna librería. Si el fallo es más complicado lo más recomendable es buscar primero en *google* y si no encontramos la solución preguntar a los desarrolladores del software por el problema.

Una vez compilado soló queda ejecutar *make install* para que el software se instale. Como lo habitual es que el programa esté destinado a ser instalado en */usr/local/* para pode ejecutar el *make install* deberemos tener privilegios de administrador.

```
~$ sudo make install
```

Con esto el programa debe estar listo para ser ejecutado puesto que el binario ejecutable debe haber sido copiado por el comando *make install* a algún directorio del $PATH, normalmente a */usr/local/bin*.

Ejercicios 1. Instala samtools 2. Instala bwa 3. Instala freebayes 4. Instala RAxML 5. Instala la última version de PyMOL

Instalando programas Java.

Java es un lenguaje de programacion de proposito general. Es un lenguaje muy usado en programas de bioinformatica porque es un lenguaje multiplataforma. Escribes una vez el codigo y lo puedes usar en los distintos sistemas operativos.

Cual es el truco? Tienes que tener instalado en el sistema el interprete de java. Este interprete será especial para cada sistema operativo, pero podra ejecutar cualquier codigo java. Existen diferentes interpretes de java, algunos son software libre, y suelen estar en la distribucion y luego tenemos el java de Oracle.

Para poder ejecutar un programa de java, necesitaremos un *Java runtime enviroment(JRE)*. Podemos instalar varios:

Java libres

Una forma de tener java en nuestro sistema es mediante las versines libres que nos ofrecen las distribuciones. Hay muchas variantes y distintas versiones para cada variante, por lo que si no sabemos que version instalar lo mejor es instalar la que ellos ponen por defecto.

```
~$ java
El programa «java» puede encontrarse en los siguientes paquetes:
* default-jre
* gcj-5-jre-headless
* openjdk-8-jre-headless
* gcj-4.8-jre-headless
* gcj-4.9-jre-headless
* openjdk-9-jre-headless
Intente: sudo apt install <paquete seleccionado>

~$ sudo apt install  default-jre
```

Java de oracle

Oracle es la empresa dueña de java. Ellos son los que lo desarrollan y por lo tanto su interprete de java suele ser el de referencia. No es software libre y por lo tanto las distrbuciones no lo pueden distribuir, pero existen formas sencilla de instalarlos. En ubuntu tenemos un PPA en el que podemos instalar un instalador que nos instalará el java de oracle. Para instalarlo:

```
~$ sudo add-apt-repository ppa:webupd8team/java
~$ sudo apt-get update
~$ sudo apt-get install oracle-java8-installer
```

Como siempre lo primero que tenemos que hacer es leernos la documentacion donde nos diga como ejecutarlo. Lo más usual es que el programa se distribuya como un fichero .jar. P.ej trimmomatic:

```
~$ java -jar Trimmommatic.jar
```

Otros programas en cambio se distribuyes con un ejecutable que se puede directamente ejecutar: p.ej: Fastqc:

```
~$ fastqc
```

Ejercicios: 1. Instala trimmomatic

1. Instala VarScan
2. Instala FastQC¿ COmo es el ejecutable para lanzar el fastq?¿Podrias modificarlo?

Python

Python es otro lenguaje de programacion de proposito general. Es un lenguaje interpretado, por lo que no hay que traducir el codigo fuente en codigo binario. De eso ya se encargara el interprete de python cada vez que ejecutamos algo.

Normalmente las distribuciones de linux suelen instalar python por defecto. Ahora mismo hay dos versiones de python conviviendo juntas; python2 y python3. Se pueden instalar las dos simultaneamente y tendremos que usar la version que nuestro programs/libreria necesite.Cuando instalamos una aplicacion en python solo lo instalaremos para la version que estemos usando en ese momento. Las librerias son unicas para esa version. Para saber cual de las 2 versiones estamos usando podemos usar el comando `which`

Tenemos varios metodos para instalar aplicaciones de python. 1. Instalarlos desde los repositorios.

1. Python tiene un repositorio de software llamado pypi donde muchos desarroladores “cuelgan” sus programas. Para poder instalar estos programas necesitamos instalar un programa que se llama `pip` . Con pip podremos administrar los paquetes del repositorio pypi. `~$ sudo apt-get install python-pip ... ~$ pip install nombre_programa`
2. Utilizando el script instalador que suelen tener los programas. Una vez descomprimido, entradoms en el directorio y ejecutamos la siguiente orden: `~$ tar zxvf programa_python.tar.gz ... ~$ cd programa_python ~$ python setup.py install`
3. Si el programa es un solo ejecutable, lo podemos instalar como cualquier otro ejecutable.

Virtualenv

Python nos permite, en aquellos entornos en los que no tenemos permisos para instalar fuera de nuestra HOME, crear un entorno virtual en el directorio que le digamos. Así, podremos instalar las apliaciones que neceitemos sin hacer cambios en el sistema. Con el comando `virtualenv` crearemos un entorno virtual en python2. Y con el comando `pyvenv` crearemos un entorno virtual en python3:

```
user@virtual:~$ virtualenv pyenv2
Running virtualenv with interpreter /usr/bin/python2
New python executable in /home/user/pyenv2/bin/python2
Also creating executable in /home/user/pyenv2/bin/python
Installing setuptools, pkg_resources, pip, wheel...done.
```

Una vez creados el entorno virtual, vemos que tenemos 1 nuevos directorio: pyenv2. Para user el entorno virtual tendremos que activarlo:

```
user@virtual:~$ source pyenv2/bin/activate
user@virtual:~$ source pyenv2/bin/activate
(pyenv2) user@virtual:~$ which python
/home/user/pyenv2/bin/python
```

Ejercicios: 1. Instala Biopython en el sistema

1. Instala pandas en el sistema
2. Instala ngs_crumbs en el sistema.
3. Crea un entorno virtual 3en python2 y repite los ejercicios 1 y 3 pero instaladolo en el entorno virtual
4. Instala la ultima version de python disponible

R

R es un entorno y lenguaje de programación con un enfoque al análisis estadístico. Para poder ejecutar cualquier libreria o progrma en R lo primero que tenemos que instalar es el interprete de R. La mejor forma es hacerlo desde un paquete de la distribución:

```
~$ sudo apt-get install r-base
```

R utilizan un repositorio llamado cran para distribuir los paquetes. Estos paquetes se pueden instalar desde la propia *shell* de R utilizando el commando install.packages(nombre_paquete):

```
~$ R

 >install.packages('ggplot2')
```

En R versiones concretas de librerias dependen de versiones concretas del interprete de R, por lo que si necesitamos una version concreta de una libreria de R, tendremos que instalar la version de R de la que dependa. En el caso que necesitemos dos versiones de R diferentes, lo tendremos que instalar a mano, ya que las distribuciones no suelen permitir la instalacion de diferentes versiones simultaneamente.

Ejercicios: 1. Instalar la libreria XML de R

1. Instala [bioconductor]
2. Instalar [la ultima version de R] manteniendo la que hemos instalado con la distribucion.

Programas que se pueden instalar

- C programs
  - bwa
  - samtools
  - bedtools
  - freebayes
  - Blast
  - bowtie y bowtie2
  - valvet
  - trinity
  - repeatmasker -> estos son chungos pero puede servir como ejercicio final
  - repeatexplorer -> estos son chungos pero puede servir como ejercicio final
  - phylip
  - RAxML -> compilarlo parece divertido . Hay que saber el prcesador que tienes.
  - phyml
- java:
  - fastqc
  - picard-tools
  - trimmomatic
  - igv
- python
  - Biopython
  - matplotlib
- repeatmasker
- emboos
- staden
- hmmer
- clustalw