Sesión 1. Introducción al biocómputo en sistemas GNU/Linux

Pablo Vinuesa, Centro de Ciencias Genómicas - UNAM

2019-07-23

Contents

1 Sesión 1. Introducción al biocómputo en sistemas GNU/Linux - Primer contacto

Este apunte fue creado para el Taller 3 - Análisis comparativo de genomas microbianos: Pangenómica y filoinformática de los Talleres Internacionales de Bioinformática - TIB2019, celebrados en el Centro de Ciencias Genómicas de la Universidad Nacional Autónoma de México, del 29 de julio al 2 de agosto de 2019 por Pablo Vinuesa, CCG-UNAM

version: 2019-07-23

Una vez que domines los comandos básicos que se presentarán seguidamente, recomiendo revisar tutoriales mucho más detallados y completos como los siguientes:

- Bash Reference Manual
- Advanced Bash Scripting Guide

1.1 Conexión a un servidor y exploración de sus características básicas

1.1.1 ssh establecer sesion remota encriptada (segura) via ssh al servidor con número dado de IP

```
ssh -l $USER IP
```

1.1.2 hostname muestra el nombre del host (la máquina a la que estoy conectado) y la IP

```
hostname
hostname -i
```

```
## alisio
## 127.0.1.1
```

1.1.3 uname muestra el sistema operativo del host

```
uname
uname -a
```

```
## Linux
## Linux alisio 4.15.0-54-generic #58-Ubuntu SMP Mon Jun 24 10:55:24 UTC 2019 x86_64 x86_64 x86_64 GNU/
```

1.1.4 htop muestra los procesos en ejecución y los recursos que consumen

```
# sales con q o CTRL-c
htop
```

1.2 Exploración del sistema de archivos

1.2.1 pwd imprime la ruta absoluta del directorio actual

```
# dónde me encuentro en el sistema?
pwd
```

```
## /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux
```

1.2.2 ls lista contenidos del directorio

```
# Qué contiene el directorio actual?
ls

# mostrar todos (-a all) los archivos, incluidos los ocultos
ls -a
```

```
## assembly_summary.txt.gz
## empty_file
## fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## github_TIB-filoinfo_screenshot.png
## intro2genomics
```

```
## intro_biocomputo_Linux_pt1.odp
## Intro_biocomputo_Linux_pt1.pdf
## linux_basic_commands.tab
## linux_commands.tab
## linux_very_basic_commands_table.csv
## recA_Balpha.fna
## recA_Balpha.fnaedtab
## recA_Bbeta.fna
## recA_Bbeta.fnaedtab
## recA_Bcanariense.fna
## recA_Bcanariense.fnaedtab
## recA_Belkanii.fna
## recA_Belkanii.fnaedtab
## recA_Bjaponicum.fna
## recA_Bjaponicum.fnaedtab
## recA_Bliaoningense.fna
## recA_Bliaoningense.fnaedtab
## recA_Bradyrhizobium_vinuesa.fna
## recA_Bradyrhizobium_vinuesa.fnaed
## recA_Bradyrhizobium_vinuesa.fnaedtab
## recA_Bsp..fna
## recA_Bsp..fnaedtab
## recA_Byuanmingense.fna
## recA_Byuanmingense.fnaedtab
## sesion_local_capt_pantalla.png
## sesion_remota_bonampak_capt_pantalla1.png
## sesion_remota_bonampak_capt_pantalla2.png
## sesion_remota_bonampak_capt_pantalla.png
## Stenotrophomonas_complete_genomes_and_ftp_paths.txt
## TIB2019-T3
## working_with_linux_commands.code
## working_with_linux_commands.html
## working_with_linux_commands.pdf
## working_with_linux_commands.Rmd
## .
## ..
## assembly_summary.txt.gz
## empty_file
## fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## github_TIB-filoinfo_screenshot.png
## intro2genomics
## intro_biocomputo_Linux_pt1.odp
## Intro_biocomputo_Linux_pt1.pdf
## linux_basic_commands.tab
## linux_commands.tab
## .linux_commands.tab.swp
## linux_very_basic_commands_table.csv
## recA_Balpha.fna
## recA_Balpha.fnaedtab
## recA_Bbeta.fna
## recA_Bbeta.fnaedtab
## recA_Bcanariense.fna
## recA_Bcanariense.fnaedtab
## recA_Belkanii.fna
```

```
## recA_Belkanii.fnaedtab
## recA_Bjaponicum.fna
## recA_Bjaponicum.fnaedtab
## recA_Bliaoningense.fna
## recA_Bliaoningense.fnaedtab
## recA_Bradyrhizobium_vinuesa.fna
## recA_Bradyrhizobium_vinuesa.fnaed
## recA_Bradyrhizobium_vinuesa.fnaedtab
## recA_Bsp..fna
## recA_Bsp..fnaedtab
## recA_Byuanmingense.fna
## recA_Byuanmingense.fnaedtab
## .Rhistory
## sesion_local_capt_pantalla.png
## sesion_remota_bonampak_capt_pantalla1.png
## sesion_remota_bonampak_capt_pantalla2.png
## sesion_remota_bonampak_capt_pantalla.png
## Stenotrophomonas_complete_genomes_and_ftp_paths.txt
## TIB2019-T3
## working_with_linux_commands.code
## working_with_linux_commands.html
## working_with_linux_commands.pdf
## working_with_linux_commands.Rmd
```

1.2.2.1 Veamos el contenido del directorio raiz

```
ls /
```

```
## bin
## boot
## cdrom
## dev
## etc
## home
## initrd.img
## initrd.img.old
## lib
## lib32
## lib64
## lost+found
## media
## mnt
## opt
## proc
## root
## run
## sbin
## snap
## srv
## swapfile
## sys
## tmp
## usr
## var
```

```
## vmlinuz
## vmlinuz.old
```

1.2.2.2 Veamos las primeras 5 entradas y últimas 5 del directorio /bin

```
ls /bin | head -20
```

```
## bash
## brltty
## bunzip2
## busybox
## bzcat
## bzcmp
## bzdiff
## bzegrep
## bzexe
## bzfgrep
## bzgrep
## bzip2
## bzip2recover
## bzless
## bzmore
## cat
## chacl
## chgrp
## chmod
## chown
```

```
# idem, pero con detalles de permisos etc
ls -l /bin | tail -5
```

```
## -rwxr-xr-x 1 root root    2131 abr 27  2017 zforce
## -rwxr-xr-x 1 root root    5938 abr 27  2017 zgrep
## -rwxr-xr-x 1 root root    2037 abr 27  2017 zless
## -rwxr-xr-x 1 root root    1910 abr 27  2017 zmore
## -rwxr-xr-x 1 root root    5047 abr 27  2017 znew
```

```
# idem, pero ordenando los archivos por fechas de modificacion (-t), listando los mas recientes al fina
ls -ltr /bin | head -20
```

```
## total 12480
## -rwxr-xr-x 1 root root      89 abr 26  2016 red
## -rwxr-xr-x 1 root root   51512 abr 26  2016 ed
## -rwxr-xr-x 1 root root   14328 ago 11  2016 ulockmgr_server
## -rwsr-xr-x 1 root root   30800 ago 11  2016 fusermount
## -rwsr-xr-x 1 root root   64424 mar  9  2017 ping
## -rwxr-xr-x 1 root root   40056 abr 21  2017 efibootmgr
## -rwxr-xr-x 1 root root   18424 abr 21  2017 efibootdump
## -rwxr-xr-x 1 root root   35512 abr 21  2017 setfacl
## -rwxr-xr-x 1 root root   23160 abr 21  2017 getfacl
## -rwxr-xr-x 1 root root   14328 abr 21  2017 chacl
## -rwxr-xr-x 1 root root    5047 abr 27  2017 znew
## -rwxr-xr-x 1 root root    1910 abr 27  2017 zmore
## -rwxr-xr-x 1 root root    2037 abr 27  2017 zless
## -rwxr-xr-x 1 root root    5938 abr 27  2017 zgrep
## -rwxr-xr-x 1 root root    2131 abr 27  2017 zforce
```

```
## -rwxr-xr-x 1 root root     140 abr 27  2017 zfgrep
## -rwxr-xr-x 1 root root     140 abr 27  2017 zegrep
## -rwxr-xr-x 1 root root    5764 abr 27  2017 zdiff
## -rwxr-xr-x 1 root root    1777 abr 27  2017 zcmp
```

1.2.3 Expansión de caracteres con * y ?

```
# lista los archivos en /bin que empiezan por las letras b y c
ls /bin/b*
ls /bin/c*
```

```
## /bin/bash
## /bin/brltty
## /bin/bunzip2
## /bin/busybox
## /bin/bzcat
## /bin/bzcmp
## /bin/bzdiff
## /bin/bzegrep
## /bin/bzexe
## /bin/bzfgrep
## /bin/bzgrep
## /bin/bzip2
## /bin/bzip2recover
## /bin/bzless
## /bin/bzmore
## /bin/cat
## /bin/chacl
## /bin/chgrp
## /bin/chmod
## /bin/chown
## /bin/chvt
## /bin/cp
## /bin/cpio
```

```
# lista los archivos en /bin que empiezan por la letra c seguida de uno o dos caracteres más
ls /bin/c?
ls /bin/c??
```

```
## /bin/cp
## /bin/cat
```

1.3 Moviéndonos por el sistema de archivos: comando cd

1.3.1 de nuevo, ¿dónde estoy?

```
pwd
```

```
## /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux
```

1.3.2 sube un directorio usando RUTA RELATIVA

```
cd ..
```

1.3.3 donde estoy?

```
pwd
```

```
## /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux
```

1.3.4 regresa a tu home

```
cd $HOME

# que es equivalente a:
cd
```

1.3.5 cd cambiar directorios con rutas absolutas (/ruta/completa/al/dir) y relativas ../../

```
# a dónde nos lleva este comando?
cd /
pwd
```

```
## /
```

- cambia de nuevo a tu home

```
cd
pwd
```

```
## /home/vinuesa
```

- sube al directorio home/ usando la ruta relativa

```
cd ../
```

1.4 Generación de directorios: comando mkdir

```
# vamos a $HOME y generamos el directorio TIB2019-T3
cd
if [ -d TIB2019-T3 ]; then
    echo "found dir TIB2019-T3"
else
    mkdir TIB2019-T3
fi
```

```
## found dir TIB2019-T3
```

- comprueba los **permisos** del nuevo directorio

```
ls -l
```

```
## total 13060
## -rw-r--r-- 1 vinuesa vinuesa 6780296 jul 21 19:26 assembly_summary.txt.gz
## -rw-r--r-- 1 vinuesa vinuesa       0 jul 23 21:48 empty_file
## -rw-r--r-- 1 vinuesa vinuesa  222602 jul 23 20:19 fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## -rw-r--r-- 1 vinuesa vinuesa   87065 jul 22 20:59 github_TIB-filoinfo_screenshot.png
## drwxr-xr-x 3 vinuesa vinuesa    4096 jul 22 21:23 intro2genomics
## -rw-r--r-- 1 vinuesa vinuesa 2186396 jul 22 21:11 intro_biocomputo_Linux_pt1.odp
## -rw-r--r-- 1 vinuesa vinuesa 1692567 jul 22 21:11 Intro_biocomputo_Linux_pt1.pdf
## -rwxr-xr-x 1 vinuesa vinuesa   10193 jul 21 11:21 linux_basic_commands.tab
## lrwxrwxrwx 1 vinuesa vinuesa      78 jul 23 21:48 linux_commands.tab -> /home/vinuesa/Cursos/TIB/TIB
## -rwxr-xr-x 1 vinuesa vinuesa    1705 jul 21 11:21 linux_very_basic_commands_table.csv
## -rw-r--r-- 1 vinuesa vinuesa    1115 jul 23 21:48 recA_Balpha.fna
## -rw-r--r-- 1 vinuesa vinuesa    1115 jul 23 21:48 recA_Balpha.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    2190 jul 23 21:48 recA_Bbeta.fna
## -rw-r--r-- 1 vinuesa vinuesa    2190 jul 23 21:48 recA_Bbeta.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   10135 jul 23 21:48 recA_Bcanariense.fna
## -rw-r--r-- 1 vinuesa vinuesa   10135 jul 23 21:48 recA_Bcanariense.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    9903 jul 23 21:48 recA_Belkanii.fna
## -rw-r--r-- 1 vinuesa vinuesa    9903 jul 23 21:48 recA_Belkanii.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   15606 jul 23 21:48 recA_Bjaponicum.fna
## -rw-r--r-- 1 vinuesa vinuesa   15606 jul 23 21:48 recA_Bjaponicum.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    8345 jul 23 21:48 recA_Bliaoningense.fna
## -rw-r--r-- 1 vinuesa vinuesa    8345 jul 23 21:48 recA_Bliaoningense.fnaedtab
## -rw-rw-r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fna
## -rw-r--r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fnaed
## -rw-r--r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    4318 jul 23 21:48 recA_Bsp..fna
## -rw-r--r-- 1 vinuesa vinuesa    4318 jul 23 21:48 recA_Bsp..fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   17768 jul 23 21:48 recA_Byuanmingense.fna
## -rw-r--r-- 1 vinuesa vinuesa   17768 jul 23 21:48 recA_Byuanmingense.fnaedtab
## -rw-rw-r-- 1 vinuesa vinuesa   51271 jul 22 18:55 sesion_local_capt_pantalla.png
## -rw-r--r-- 1 vinuesa vinuesa   40746 jul 22 18:56 sesion_remota_bonampak_capt_pantalla1.png
## -rw-r--r-- 1 vinuesa vinuesa    4468 jul 22 18:57 sesion_remota_bonampak_capt_pantalla2.png
## -rw-rw-r-- 1 vinuesa vinuesa  408580 jul 22 18:51 sesion_remota_bonampak_capt_pantalla.png
## -rw-r--r-- 1 vinuesa vinuesa   47651 jul 23 21:48 Stenotrophomonas_complete_genomes_and_ftp_paths.tx
## drwxr-xr-x 3 vinuesa vinuesa    4096 jul 23 18:41 TIB2019-T3
## -rwxr-xr-x 1 vinuesa vinuesa    6047 jul 21 11:21 working_with_linux_commands.code
## -rw-r--r-- 1 vinuesa vinuesa 1041164 jul 23 21:47 working_with_linux_commands.html
## -rw-r--r-- 1 vinuesa vinuesa  315982 jul 23 21:48 working_with_linux_commands.pdf
## -rw-r--r-- 1 vinuesa vinuesa   38374 jul 23 21:52 working_with_linux_commands.Rmd
```

- generemos un subdirectorio por debajo del que acabamos de crear:

```
mkdir -p TIB2019-T3/sesion1_linux && cd TIB2019-T3/sesion1_linux
```

1.4.1 permisos

- cambiamos a /home/vinuesa e intenta crear estos mismos directorios ahí

```
cd /home/vinuesa && mkdir -p TIB2019-T3/sesion1_linux
```

1.5 Copiar, mover, renombrar y borrar archivos con: cp, mv y rm

```
# cambia a tu home, y luego a TIB2019-T3/sesion1_linux
cd && cd TIB2019-T3/sesion1_linux
```

1.5.1 copia de archivo simple: cp file .

- copia el archivo /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/linux_basic_commands.tab al directorio actual

```
cp /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/linux_basic_commands.tab . # <<< vean el punto, s
```

- otra manera, usando rutas absolutas y la variable de ambiente $HOME

```
cp /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/linux_basic_commands.tab $HOME/TIB2019-T3/sesion1
```

1.5.2 copiado de directorio: cp -r dir .

- copiar el directorio /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/ a tu dir actual

```
# Noten el punto '.' y cp -r (recursively), necesario para copiar directorios completos
cp -r /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data .
```

1.5.3 Eliminar un directorio: rm -rf [recursively -r and force -f]

```
mkdir borrame

cp linux_basic_commands.tab borrame

ls borrame

rm -rf borrame
```

```
## linux_basic_commands.tab
```

Prueba ahora este comando

```
rm data
```

qué pasa?

¿Cómo tengo que borrar un directorio? rm -rf directorio

```
rm -rf data
```

1.6 Generación de ligas simbólicas a archivos: comando ln -s /ruta/al/archivo/fuente nombre_la_liga

Esto es muy importante, ya que permite ahorrar mucho espacio en disco al evitar la multiplicación de copias fisicas en el disco duro del mismo archivo en el $HOME de uno o más usuarios

```
hostn=$(hostname)
if [ "$hostn" == "Tenerife" ]; then
   ln -s /home/vinuesa/Cursos/OMICAS_UAEM_genomica/clase1_intro2linux/linux_basic_commands.tab comandos
elif [ "$hostn" == "buluc" ]; then
     ln -s /home/vinuesa/cursos/TIB2019-T3/sesion1_linux/data/linux_basic_commands.tab comandos_de_linu
elif [ "$hostn" == "alisio" ]; then
  ln -s /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux/linux_basic_commands.tab comandos_de_linu
elif [ "$hostn" == "bonampak" ]; then
   ln -s /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/linux_basic_commands.tab comandos_de_linux.tab
   ln -s /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/assembly_summary.txt.gz .
fi

# confirmamos que se generaron las ligas
ls -l
```

```
## total 13064
## -rw-r--r-- 1 vinuesa vinuesa 6780296 jul 21 19:26 assembly_summary.txt.gz
## lrwxrwxrwx 1 vinuesa vinuesa      78 jul 23 21:52 comandos_de_linux.tab -> /home/vinuesa/Cursos/TIB/
## -rw-r--r-- 1 vinuesa vinuesa       0 jul 23 21:48 empty_file
## -rw-r--r-- 1 vinuesa vinuesa  222602 jul 23 20:19 fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## -rw-r--r-- 1 vinuesa vinuesa   87065 jul 22 20:59 github_TIB-filoinfo_screenshot.png
## drwxr-xr-x 3 vinuesa vinuesa    4096 jul 22 21:23 intro2genomics
## -rw-r--r-- 1 vinuesa vinuesa 2186396 jul 22 21:11 intro_biocomputo_Linux_pt1.odp
## -rw-r--r-- 1 vinuesa vinuesa 1692567 jul 22 21:11 Intro_biocomputo_Linux_pt1.pdf
## -rwxr-xr-x 1 vinuesa vinuesa   10193 jul 21 11:21 linux_basic_commands.tab
## lrwxrwxrwx 1 vinuesa vinuesa      78 jul 23 21:48 linux_commands.tab -> /home/vinuesa/Cursos/TIB/TIB
## -rwxr-xr-x 1 vinuesa vinuesa    1705 jul 21 11:21 linux_very_basic_commands_table.csv
## -rw-r--r-- 1 vinuesa vinuesa    1115 jul 23 21:48 recA_Balpha.fna
## -rw-r--r-- 1 vinuesa vinuesa    1115 jul 23 21:48 recA_Balpha.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    2190 jul 23 21:48 recA_Bbeta.fna
## -rw-r--r-- 1 vinuesa vinuesa    2190 jul 23 21:48 recA_Bbeta.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   10135 jul 23 21:48 recA_Bcanariense.fna
## -rw-r--r-- 1 vinuesa vinuesa   10135 jul 23 21:48 recA_Bcanariense.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    9903 jul 23 21:48 recA_Belkanii.fna
## -rw-r--r-- 1 vinuesa vinuesa    9903 jul 23 21:48 recA_Belkanii.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   15606 jul 23 21:48 recA_Bjaponicum.fna
## -rw-r--r-- 1 vinuesa vinuesa   15606 jul 23 21:48 recA_Bjaponicum.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    8345 jul 23 21:48 recA_Bliaoningense.fna
## -rw-r--r-- 1 vinuesa vinuesa    8345 jul 23 21:48 recA_Bliaoningense.fnaedtab
## -rw-rw-r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fna
## -rw-r--r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fnaed
## -rw-r--r-- 1 vinuesa vinuesa   69380 jul 23 21:48 recA_Bradyrhizobium_vinuesa.fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa    4318 jul 23 21:48 recA_Bsp..fna
## -rw-r--r-- 1 vinuesa vinuesa    4318 jul 23 21:48 recA_Bsp..fnaedtab
## -rw-r--r-- 1 vinuesa vinuesa   17768 jul 23 21:48 recA_Byuanmingense.fna
## -rw-r--r-- 1 vinuesa vinuesa   17768 jul 23 21:48 recA_Byuanmingense.fnaedtab
## -rw-rw-r-- 1 vinuesa vinuesa   51271 jul 22 18:55 sesion_local_capt_pantalla.png
## -rw-r--r-- 1 vinuesa vinuesa   40746 jul 22 18:56 sesion_remota_bonampak_capt_pantalla1.png
## -rw-r--r-- 1 vinuesa vinuesa    4468 jul 22 18:57 sesion_remota_bonampak_capt_pantalla2.png
## -rw-rw-r-- 1 vinuesa vinuesa  408580 jul 22 18:51 sesion_remota_bonampak_capt_pantalla.png
## -rw-r--r-- 1 vinuesa vinuesa   47651 jul 23 21:48 Stenotrophomonas_complete_genomes_and_ftp_paths.tx
## drwxr-xr-x 3 vinuesa vinuesa    4096 jul 23 18:41 TIB2019-T3
## -rwxr-xr-x 1 vinuesa vinuesa    6047 jul 21 11:21 working_with_linux_commands.code
## -rw-r--r-- 1 vinuesa vinuesa 1041164 jul 23 21:47 working_with_linux_commands.html
```

```
## -rw-r--r-- 1 vinuesa vinuesa  315982 jul 23 21:48 working_with_linux_commands.pdf
## -rw-r--r-- 1 vinuesa vinuesa   38374 jul 23 21:52 working_with_linux_commands.Rmd
```

1.6.1 renombramos la liga (o cualquier archivo o directorio)

```
mv comandos_de_linux.tab linux_commands.tab
```

1.7 Visualización de contenidos de archivos: comando head, tail, cat, less, more

1.7.1 uso de head y tail para desplegar la cabecera y cola de archivos

```
head linux_commands.tab
```

```
## IEEE Std 1003.1-2008 utilities Name  Category    Description     First appeared
## admin    SCCS    Create and administer SCCS files    PWB UNIX
## alias    Misc    Define or display aliases
## ar   Misc    Create and maintain library archives    Version 1 AT&T UNIX
## asa  Text processing     Interpret carriage-control characters   System V
## at   Process management  Execute commands at a later time    Version 7 AT&T UNIX
## awk  Text processing     Pattern scanning and processing language    Version 7 AT&T UNIX
## basename     Filesystem  Return non-directory portion of a pathname; see also dirname    Version 7 A
## batch    Process management  Schedule commands to be executed in a batch queue
## bc   Misc    Arbitrary-precision arithmetic language     Version 6 AT&T UNIX
```

```
tail linux_commands.tab
```

```
## val  SCCS    Validate SCCS files     System III
## vi   Text processing     Screen-oriented (visual) display editor     1BSD
## wait     Process management  Await process completion    Version 4 AT&T UNIX
## wc   Text processing     Line, word and byte or character count  Version 1 AT&T UNIX
## what     SCCS    Identify SCCS files     PWB UNIX
## who  System administration   Display who is on the system    Version 1 AT&T UNIX
## write    Misc    Write to another user's terminal    Version 1 AT&T UNIX
## xargs    Shell programming   Construct argument lists and invoke utility     PWB UNIX
## yacc     C programming   Yet another compiler compiler   PWB UNIX
## zcat     Text processing     Expand and concatenate data     4.3BSD
```

```
# le podemos indicar el numero de lineas a desplegar
head -3 linux_commands.tab
```

```
## IEEE Std 1003.1-2008 utilities Name  Category    Description     First appeared
## admin    SCCS    Create and administer SCCS files    PWB UNIX
## alias    Misc    Define or display aliases
```

```
tail -1 linux_commands.tab
```

```
## zcat     Text processing     Expand and concatenate data     4.3BSD
```

1.7.2 cat despliega uno o más archivos, concatenándolos

```
cat linux_commands.tab | head
```

```
## IEEE Std 1003.1-2008 utilities Name  Category    Description     First appeared
## admin    SCCS    Create and administer SCCS files    PWB UNIX
## alias    Misc    Define or display aliases
## ar   Misc    Create and maintain library archives    Version 1 AT&T UNIX
## asa  Text processing     Interpret carriage-control characters   System V
## at   Process management  Execute commands at a later time    Version 7 AT&T UNIX
## awk  Text processing     Pattern scanning and processing language    Version 7 AT&T UNIX
## basename     Filesystem  Return non-directory portion of a pathname; see also dirname    Version 7 A
## batch    Process management  Schedule commands to be executed in a batch queue
## bc   Misc    Arbitrary-precision arithmetic language     Version 6 AT&T UNIX
```

cat -n nos permite añadir números de línea a los archivos desplegados

```
cat -n linux_commands.tab | head
```

```
##      1   IEEE Std 1003.1-2008 utilities Name     Category    Description     First appeared
##      2   admin   SCCS    Create and administer SCCS files    PWB UNIX
##      3   alias   Misc    Define or display aliases
##      4   ar  Misc    Create and maintain library archives    Version 1 AT&T UNIX
##      5   asa     Text processing     Interpret carriage-control characters   System V
##      6   at  Process management  Execute commands at a later time    Version 7 AT&T UNIX
##      7   awk     Text processing     Pattern scanning and processing language    Version 7 AT&T UNIX
##      8   basename    Filesystem  Return non-directory portion of a pathname; see also dirname    Vers
##      9   batch   Process management  Schedule commands to be executed in a batch queue
##     10   bc  Misc    Arbitrary-precision arithmetic language     Version 6 AT&T UNIX
```

1.7.3 el paginador less despliega archivos página a página

```
less linux_commands.tab | head
```

```
## IEEE Std 1003.1-2008 utilities Name  Category    Description     First appeared
## admin    SCCS    Create and administer SCCS files    PWB UNIX
## alias    Misc    Define or display aliases
## ar   Misc    Create and maintain library archives    Version 1 AT&T UNIX
## asa  Text processing     Interpret carriage-control characters   System V
## at   Process management  Execute commands at a later time    Version 7 AT&T UNIX
## awk  Text processing     Pattern scanning and processing language    Version 7 AT&T UNIX
## basename     Filesystem  Return non-directory portion of a pathname; see also dirname    Version 7 A
## batch    Process management  Schedule commands to be executed in a batch queue
## bc   Misc    Arbitrary-precision arithmetic language     Version 6 AT&T UNIX
```

Nota: con *q* salimos del paginador less

1.8 Edición de archivos con los editores vim o [g|n]edit

vim (vi improved) es un poderoso editor programable presente en todos los sistemas UNIX. La principal característica tanto de Vim como de Vi consiste en que disponen de diferentes modos entre los que se alterna para realizar ciertas operaciones, lo que los diferencia de la mayoría de editores comunes, que tienen un solo modo en el que se introducen las órdenes mediante combinaciones de teclas (o interfaces gráficas). Se controla por completo mediante el teclado desde un Terminal, por lo que puede usarse sin problemas a través de conexiones remotas ya que no carga el sistema al no desplegar un entorno gráfico.

Es muy recomendable aprender a usar VIM, pero no tenemos tiempo de hacerlo en el TIB, por lo que les recomiendo este tutorial de uso de VIM en español, o directamente en su terminal tecleando el comando

```
vimtutor

# para salir de vim,

<ESC> # para estar seguros que estamos en modo ex
:q
```

En el taller usaremos generalmente el editor con ambiente gráfico gedit, de uso muy sencillo y simiar al block de notas de Windows o similar

```
# noten el uso de & al final de la sentencia para enviar el proceso al fondo
# para evitar que bloquee la terminal
gedit linux_commands.tab &
```

1.9 Edición de archivos con el editor de flujo sed (stream editor)

sed (stream editor) es un editor de flujo, una potente herramienta de tratamiento de texto para el sistema operativo Unix que acepta como entrada un archivo, lo lee y modifica línea a línea de acuerdo a un script, mostrando el resultado por salida estándar (normalmente en pantalla, a menos que se realice una redirección). Sed permite manipular flujos de datos, como por ejemplo cortar líneas, buscar y reemplazar texto (con soporte de xpresiones regulares), entre otras cosas. Posee muchas características de ed y ex.

La sintaxis general de la orden sed es:

```
$ sed [-n] [-e'script'] [-f archivo] archivo1 archivo2 ...
```

donde:

```
-n indica que se suprima la salida estándar.
-e indica que se ejecute el script que viene a continuación. Si no se emplea la opción -f se puede omit
-f indica que las órdenes se tomarán de un archivo
```

1.9.1 Ejemplos de uso básico de sed:

- Cambia todas las minúsculas a mayúsculas de archivo:

```
$ sed 'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/' archivo
```

- Borra la 1ª línea de archivo:

```
$ sed '1d' archivo
```

- Elimina las líneas en blanco. Nótese el uso de expresiones regulares, done:
  - // delimitan la expresión regular. Noten que hay que escaparla entre commillas sencillas.
  - ^ indica el inicio de la línea
  - $ indica el término de la línea

```
$ sed '/^$/d' archivo
```

- Genera una lista numerada de los nombres de campos o cabeceras del archivo linux_commands.tab
  - // delimitan la expresión regular. Noten que hay que escaparla entre commillas sencillas.
  - \t representa al tabulador
  - \m representa el salto de línea
  - //g la g indica que se reemplacen todas las instancias

```
head -1 linux_commands.tab | sed 's/\t/\n/g' | cat -n
```

```
##      1   IEEE Std 1003.1-2008 utilities Name
##      2   Category
```

```
##      3   Description
##      4   First appeared
```

1.10 Uso de tuberias de herramientas UNIX/Linux para filtrado de texto con cut, grep, sort, uniq, wc y |

UNIX y Linux ofrecen una gran cantidad de herramientas para todo tipo de trabajos, cada una generalmente con muchas opciones. En bioinformática y genómica, los archivos de texto plano (ASCII) son los más comunes. Por ello es muy útil dominar algunas de las herramientas de filtrado de texto más comunes. Como ejemplo, trabajaremos con el archivo assembly_summary.txt, que contiene los datos de ensambles genómicos de la división RefSeq de GenBank. Lo descargué y comprimí con los siguientes comandos:

```
wget -c ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/assembly_summary.txt
gzip assembly_summary.txt
```

- Exploremos el archivo comprimido (con compresión gnu zip) con usando los comandos zless o zcat

```
zless assembly_summary.txt.gz
```

```
# veamos las 5 primeras líneas del archivo
zcat assembly_summary.txt.gz | head -5
```

```
## #   See ftp://ftp.ncbi.nlm.nih.gov/genomes/README_assembly_summary.txt for a description of the colu
## # assembly_accession bioproject  biosample   wgs_master  refseq_category taxid   species_taxid   orga
## GCF_000010525.1  PRJNA224116 SAMD00060925        representative genome   438753  7   Azorhizobium cau
## GCF_000007365.1  PRJNA224116 SAMN02604269        representative genome   198804  9   Buchnera aphidi
## GCF_000007725.1  PRJNA224116 SAMN02604289        representative genome   224915  9   Buchnera aphidi
```

1.10.1 Ejemplos de herramientas de filtrado de texto en acción

- cut corta líneas de texto/tablas por delimitadores de campo (-d) específicos (TAB por defecto), extrayendo los campos indicados con -f (cut -d' ' -f1-3,5,9)
- sort ordena (sor -u; sort -nrk2; sort -dk1)
- wc cuenta líneas, palabras y caracteres (wc -l)
- uniq regresa listas de valores únicos (uniq -c)
- grep Filtra las lineas de un archivo que contienen (o no) caracteres o expresiones regulares (grep -E '^XXX|YYY|zzz$'; grep -v '^#')
- el pipe '|' conecta la salida de un comando con la entrada >STDIN> de otro
- ¿cuántas líneas tiene el archivo assembly_summary.txt.gz?

```
# ¿cuántas líneas tiene el archivo assembly_summary.txt.gz?
zcat assembly_summary.txt.gz | wc
zcat assembly_summary.txt.gz | wc -l
```

```
##  161297 3788695 48497020
## 161297
```

- la columna assembly_level (#12) indica el estado del ensamble. ¿Cuáles son los niveles de la variable categórica assembly_level (valores únicos de la misma?

```
# la columna assembly_level (#12) indica el estado del ensamble. ¿Cuáles son los niveles de la variable
zcat assembly_summary.txt.gz | grep -v "^#" | cut -f 12 | sort -u
```

```
## Chromosome
## Complete Genome
## Contig
## Scaffold
```

- ¿cuántos genomas hay por nivel de la variable categórica assembly_level?

```
# ¿cuántos genomas hay por nivel de la variable categórica assembly_level?
zcat assembly_summary.txt.gz | grep -v "^#" | cut -f 12 | sort | uniq -c
```

```
##    2018 Chromosome
##   13983 Complete Genome
##   82755 Contig
##   62539 Scaffold
```

- asocia cada nombre de columna de la cabecera con el número de la columna correspondiente

```
# asocia cada nombre de columna de la cabecera con el número de la columna correspondiente
zcat assembly_summary.txt.gz | head -2 | sed '1d; s/\t/\n/g' | cat -n
```

```
##      1   # assembly_accession
##      2   bioproject
##      3   biosample
##      4   wgs_master
##      5   refseq_category
##      6   taxid
##      7   species_taxid
##      8   organism_name
##      9   infraspecific_name
##     10   isolate
##     11   version_status
##     12   assembly_level
##     13   release_type
##     14   genome_rep
##     15   seq_rel_date
##     16   asm_name
##     17   submitter
##     18   gbrs_paired_asm
##     19   paired_asm_comp
##     20   ftp_path
##     21   excluded_from_refseq
##     22   relation_to_type_material
```

- genera una estadística del número de genomas por especie (columna # 8), y muestra sólo las 10 especies con más genomas secuenciados!

```
# genera una estadística del número de genomas por especie (columna # 8), y muestra sólo las 10 especie
zcat assembly_summary.txt.gz | grep -v "^#" | cut -f8 | sort | uniq -c | sort -nrk1 | head -10
```

```
##   14089 Escherichia coli
##    8039 Streptococcus pneumoniae
##    6398 Klebsiella pneumoniae
##    5924 Staphylococcus aureus
##    4556 Mycobacterium tuberculosis
##    4358 Pseudomonas aeruginosa
##    3164 Acinetobacter baumannii
##    2789 Listeria monocytogenes
##    2173 Salmonella enterica subsp. enterica serovar Typhi
```

```
##    1792 Clostridioides difficile
```

- ¿Cuántos genomas completos hay del género Acinetobacter?

```
# ¿Cuántos genomas completos hay del género Acinetobacter?
zcat assembly_summary.txt.gz | grep Acinetobacter | grep Complete | wc -l

# también puedes usar zgrep para evitar la llamada primero a zcat
zgrep Acinetobacter assembly_summary.txt.gz | grep Complete | wc -l
```

```
## 220
## 220
```

```
# ojo: Linux es sensible a mayúsculas y minúsculas: prueba este comando para comprobarlo
zgrep acinetobacter assembly_summary.txt.gz | grep Complete | wc -l # no encuentra nada
```

```
# grep -i lo hace insensible a la fuente
zgrep -i acinetobacter assembly_summary.txt.gz | grep Complete | wc -l
```

```
## 220
```

- filtra y cuenta las lineas que contienen Acinetobacter o Stenotrophomonas

```
# filtra y cuenta las lineas que contienen Acinetobacter o Stenotrophomonas
zgrep -E 'Acinetobacter|Stenotrophomonas' assembly_summary.txt.gz | wc -l
```

```
## 5170
```

- Cuenta los genomas de Acinetobacter, Pseudomonas y Klebsiella (por género) y presenta una lista ordenada por número decreciente de genomas

```
# Cuenta los genomas de Acinetobacter, Pseudomonas y Klebsiella (por género) y presenta una lista orden
zgrep -E 'Acinetobacter|Pseudomonas|Klebsiella' assembly_summary.txt.gz | cut -f 8 | cut -d' ' -f1 |sor
```

```
##    8951 Pseudomonas
##    8515 Klebsiella
##    4747 Acinetobacter
##       7 [Pseudomonas]
##       1 Candidatus
```

- Cuenta los genomas de Acinetobacter, Pseudomonas y Klebsiella (por género), con salida ordenada alfabéticamente por género

```
# filtra las lineas que contienen Filesystem o Text processing y ordénalas alfabéticamente según las en
# eliminando las entradas de Candidatus y [Pseudomonas]
zgrep -E 'Acinetobacter|Pseudomonas|Klebsiella' assembly_summary.txt.gz | cut -f 8 | cut -d' ' -f1 | gr
```

```
##    4747 Acinetobacter
##    8515 Klebsiella
##    8951 Pseudomonas
```

Veremos la gran utilidad y versatilidad de combinaciones de estos comandos para el procesamiento de archivos de secuencias en un ejercicio más adelante.

1.11 Manual de cada comando: man command

```
# mira las opciones de cut y sort en la manpage
man cut | head -20

man sort | head -20
```

```
## CUT(1)                                                                  User Commands
##
## NAME
##        cut - remove sections from each line of files
##
## SYNOPSIS
##        cut OPTION... [FILE]...
##
## DESCRIPTION
##        Print selected parts of lines from each FILE to standard output.
##
##        With no FILE, or when FILE is -, read standard input.
##
##        Mandatory arguments to long options are mandatory for short options too.
##
##        -b, --bytes=LIST
##               select only these bytes
##
##        -c, --characters=LIST
##               select only these characters
## SORT(1)                                                                 User Commands
##
## NAME
##        sort - sort lines of text files
##
## SYNOPSIS
##        sort [OPTION]... [FILE]...
##        sort [OPTION]... --files0-from=F
##
## DESCRIPTION
##        Write sorted concatenation of all FILE(s) to standard output.
##
##        With no FILE, or when FILE is -, read standard input.
##
##        Mandatory arguments to long options are mandatory for short options too.  Ordering options:
##
##        -b, --ignore-leading-blanks
##               ignore leading blanks
##
##        -d, --dictionary-order
```

1.12 redireccionado de la salida STOUT a un archivo con el comando >

```
zgrep Stenotrophomonas assembly_summary.txt.gz | cut -f8,20 > Stenotrophomonas_complete_genomes_and_ftp
```

1.13 Inicios de programación en Bash

Vermos aquí unas pocas construcciones muy básicas de probramación Shell

1.13.1 Asignación de variables

- La sintaxis básica de asignación es:

```
varName=VALUE
```

- para recuperar el valor de una varialbe, le añadimos el prefijo $. Para imprimir el valor asignado a la variable, usamos echo $varName

```
archivo_de_comandos_linux=linux_commands.tab
echo "$archivo_de_comandos_linux"
```

```
## linux_commands.tab
```

- para capturar la salida de un comando usammos $(comando)

```
wkdir=$(pwd)
date=$(date | awk '{print $3,$2,$6}' | sed 's/ //g')
h=$(hostname)
echo ">>> working in: $wkdir at <$h> on <$date>"
```

```
## >>> working in: /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux at <alisio> on <23jul2019>
```

- Modificación de variables y operaciones con ellas

```
wkdir=$(pwd)
echo "wkdir: $wkdir"

# 1. cortemos caracteres por la izquierda (todos los caracteres por la izquierda, hasta llegar a último
basedir=${wkdir##*/}
echo "basedir: $basedir  # \${wkdir##*/}"

# 2. cortemos caracteres por la derecha (cualqier caracter hasta llegar a /)
echo "path to basedir: ${wkdir%/*}  # \${wkdir%/*}"

# 3. contar el número de caracteres (longitud) de la variable
echo "basedir has ${#basedir} characters  # \${#basedir}"
```

```
## wkdir: /home/vinuesa/Cursos/TIB/TIB19-T3/sesion1_intro2linux
## basedir: sesion1_intro2linux  # ${wkdir##*/}
## path to basedir: /home/vinuesa/Cursos/TIB/TIB19-T3  # ${wkdir%/*}
## basedir has 19 characters  # ${#basedir}
```

1.13.2 Condicionales

- La sintaxis básica de un condicional simple en formato de una línea es así

if [condición]; then orden1; orden2 . . . ; fi

- también hay una versión más corta para test siples

[condición] && setecia1 && sentencia2

- En un script, lo escibimos generalmente como un bloque indentado, para mejor legibilidad

```
if [ condición ]; then
    orden1
    orden2
fi
```

1.13.2.1 Comparación de íntegros en condicionales

```
i=5
j=3

if [ "$i" -lt "$j" ]; then
   echo "$i < $j"
elif [ "$i" -gt "$j" ]; then
   echo "$i > $j "
fi
```

```
## 5 > 3
```

1.13.2.2 Comparación de cadenas de caracteres en condicionales

```
c=carla
j=juan

if [ "$c" == "$j" ]; then
   echo "$c = $j"
elif [ "$c" != "$j" ]; then
   echo "c:$i != j:$j "
fi
```

```
## c: != j:juan
```

1.13.2.3 Comprobación de la existencia de un archivo de tamaño > 0 bytes

```
touch empty_file
ls -l empty_file
ls -l *gz
f=$(ls *gz)

if [ -e empty_file  ]; then
   echo "empty_file file exists"
fi

if [ ! -s empty_file  ]; then
   echo "empty_file file exists but is empty"
fi

if [ -s "$f" ]; then
   echo "$f exists and is non-empty"
fi
```

```
## -rw-r--r-- 1 vinuesa vinuesa 0 jul 23 21:52 empty_file
## -rw-r--r-- 1 vinuesa vinuesa 6780296 jul 21 19:26 assembly_summary.txt.gz
## empty_file file exists
## empty_file file exists but is empty
## assembly_summary.txt.gz exists and is non-empty
```

1.13.2.4 La versión corta de test [condición] && ejecuta orden1 && ejecuta orden2 ...

```
# también podemos usar la versión corta del test:
f=$(ls *gz)
[ -s "$f" ] && echo "$f exists and is non-empty"
```

```
## assembly_summary.txt.gz exists and is non-empty
```

1.13.2.5 if; elif; else

```
  if [[ "$OSTYPE" == "linux-gnu" ]]
  then
    OS='linux'
    no_cores=$(awk '/^processor/{n+=1}END{print n}' /proc/cpuinfo)
    host=$(hostname)
    echo "running on $host under $OS with $no_cores cores :)"
  elif [[ "$OSTYPE" == "darwin"* ]]
  then
    OS='darwin'
    no_cores=$(sysctl -n hw.ncpu)
    host=$(hostname)
    echo "running on $host under $OS with $no_cores cores :)"
  else
       OS='windows'
       echo "oh no! another windows box :( ... you should better change to linux :) "
  fi
```

```
## running on alisio under linux with 12 cores :)
```

1.13.3 Bucles for

la sintaxis general de un bucle for en Bash es:

for ALIAS in LIST; do CMD1; CMD2; done

donde el usuario tiene que cambiar los términos en mayúsculas por opciones concretas. ALIAS es el nombre de una variable a la que se asigna secuencialmente cada valor de LIST.

Así por ejemplo, si tuviéramos muchos archivos de secuencias homólogas con la extensión *.faa en un directorio, podríamos alinearlas secuencialmente con un comando como el siguiente:

for file in *.faa; do clustalo -i $file -o ${file%.faa}_cluoAln.faa; done

donde ALIAS=file, LIST=*.faa y CMD1 es una llamada al programa de alineamientos múltiples clustalo que veremos más adelante en este taller.

1.13.3.1 Ejemplo de bucle for, acoplado a las herramientas de filtrado y de manipulación de variables

La idea del ejercicio es generar archivos a partir de linux_basic_commands.tab que contengan sólo los comandos de cada clase, nombrando a los archivo resultantes con el valor de dicha clase, almacenados en la segunda columna de la tabla

```
# veamos la cabecera y cola del archivo linux_basic_commands.tab
head linux_basic_commands.tab
echo '-------------------------------------------------'
tail linux_basic_commands.tab
```

```
## IEEE Std 1003.1-2008 utilities Name  Category    Description     First appeared
## admin    SCCS    Create and administer SCCS files    PWB UNIX
## alias    Misc    Define or display aliases
```

```
## ar   Misc    Create and maintain library archives    Version 1 AT&T UNIX
## asa  Text processing     Interpret carriage-control characters   System V
## at   Process management  Execute commands at a later time    Version 7 AT&T UNIX
## awk  Text processing     Pattern scanning and processing language    Version 7 AT&T UNIX
## basename     Filesystem  Return non-directory portion of a pathname; see also dirname    Version 7 A
## batch    Process management  Schedule commands to be executed in a batch queue
## bc   Misc    Arbitrary-precision arithmetic language     Version 6 AT&T UNIX
## ------------------------------------------------
## val  SCCS    Validate SCCS files     System III
## vi   Text processing     Screen-oriented (visual) display editor     1BSD
## wait     Process management  Await process completion    Version 4 AT&T UNIX
## wc   Text processing     Line, word and byte or character count  Version 1 AT&T UNIX
## what     SCCS    Identify SCCS files     PWB UNIX
## who  System administration   Display who is on the system    Version 1 AT&T UNIX
## write    Misc    Write to another user's terminal    Version 1 AT&T UNIX
## xargs    Shell programming   Construct argument lists and invoke utility     PWB UNIX
## yacc     C programming   Yet another compiler compiler   PWB UNIX
## zcat     Text processing     Expand and concatenate data     4.3BSD
```

Antes de correr el bucle, lista los archivos en el directorio de trabajo

```
# veamos el contenido del directorio antes de correr el bucle
ls
```

```
## assembly_summary.txt.gz
## empty_file
## fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## github_TIB-filoinfo_screenshot.png
## intro2genomics
## intro_biocomputo_Linux_pt1.odp
## Intro_biocomputo_Linux_pt1.pdf
## linux_basic_commands.tab
## linux_commands.tab
## linux_very_basic_commands_table.csv
## recA_Balpha.fna
## recA_Balpha.fnaedtab
## recA_Bbeta.fna
## recA_Bbeta.fnaedtab
## recA_Bcanariense.fna
## recA_Bcanariense.fnaedtab
## recA_Belkanii.fna
## recA_Belkanii.fnaedtab
## recA_Bjaponicum.fna
## recA_Bjaponicum.fnaedtab
## recA_Bliaoningense.fna
## recA_Bliaoningense.fnaedtab
## recA_Bradyrhizobium_vinuesa.fna
## recA_Bradyrhizobium_vinuesa.fnaed
## recA_Bradyrhizobium_vinuesa.fnaedtab
## recA_Bsp..fna
## recA_Bsp..fnaedtab
## recA_Byuanmingense.fna
## recA_Byuanmingense.fnaedtab
## sesion_local_capt_pantalla.png
## sesion_remota_bonampak_capt_pantalla1.png
```

```
## sesion_remota_bonampak_capt_pantalla2.png
## sesion_remota_bonampak_capt_pantalla.png
## Stenotrophomonas_complete_genomes_and_ftp_paths.txt
## TIB2019-T3
## working_with_linux_commands.code
## working_with_linux_commands.html
## working_with_linux_commands.pdf
## working_with_linux_commands.Rmd
```

Ahora el bucle. En este caso ALIAS=type y LIST corresponde a la lista de valores únicos almacenados en la segunda columna de la tabla: $(cut -f2 linux_basic_commands.tab | sort -u)

```
#>>> Ejemplo integrativo: usa un bucle for, acoplado a las herramientas de filtrado arriba mostradas,
#    para generar archivos que contengan solo los comandos de las diferentes categorias
#    nombrando a los archivos por estas

# for type in $(cut -f2 linux_basic_commands.tab | sort -u); do grep "$type" linux_basic_commands.tab >
for type in $(cut -f2 linux_basic_commands.tab | sort -u); do
    grep "$type" linux_basic_commands.tab > ${type}.cmds
done
```

Y voilà:

```
# veamos el contenido del directorio después de correr el bucle
ls
```

```
## administration.cmds
## assembly_summary.txt.gz
## Batch.cmds
## Category.cmds
## C.cmds
## empty_file
## fetch_recA_bradys_vinuesa_nuccore_screenshot.png
## Filesystem.cmds
## FORTRAN77.cmds
## github_TIB-filoinfo_screenshot.png
## intro2genomics
## intro_biocomputo_Linux_pt1.odp
## Intro_biocomputo_Linux_pt1.pdf
## linux_basic_commands.tab
## linux_commands.tab
## linux_very_basic_commands_table.csv
## management.cmds
## Misc.cmds
## Network.cmds
## Process.cmds
## processing.cmds
## programming.cmds
## Programming.cmds
## recA_Balpha.fna
## recA_Balpha.fnaedtab
## recA_Bbeta.fna
## recA_Bbeta.fnaedtab
## recA_Bcanariense.fna
## recA_Bcanariense.fnaedtab
## recA_Belkanii.fna
```

```
## recA_Belkanii.fnaedtab
## recA_Bjaponicum.fna
## recA_Bjaponicum.fnaedtab
## recA_Bliaoningense.fna
## recA_Bliaoningense.fnaedtab
## recA_Bradyrhizobium_vinuesa.fna
## recA_Bradyrhizobium_vinuesa.fnaed
## recA_Bradyrhizobium_vinuesa.fnaedtab
## recA_Bsp..fna
## recA_Bsp..fnaedtab
## recA_Byuanmingense.fna
## recA_Byuanmingense.fnaedtab
## SCCS.cmds
## sesion_local_capt_pantalla.png
## sesion_remota_bonampak_capt_pantalla1.png
## sesion_remota_bonampak_capt_pantalla2.png
## sesion_remota_bonampak_capt_pantalla.png
## Shell.cmds
## Stenotrophomonas_complete_genomes_and_ftp_paths.txt
## System.cmds
## Text.cmds
## TIB2019-T3
## utilities.cmds
## working_with_linux_commands.code
## working_with_linux_commands.html
## working_with_linux_commands.pdf
## working_with_linux_commands.Rmd
```

```
# veamos el contenido de uno de los nuevos archivos generados
cat programming.cmds
```

```
## cc/c99   C programming   Compile standard C programs     IEEE Std 1003.1-2001
## cflow    C programming   Generate a C-language call graph    System V
## command  Shell programming   Execute a simple command
## ctags    C programming   Create a tags file  3BSD
## cxref    C programming   Generate a C-language program cross-reference table     System V
## echo     Shell programming   Write arguments to standard output  Version 2 AT&T UNIX
## expr     Shell programming   Evaluate arguments as an expression     Version 7 AT&T UNIX
## false    Shell programming   Return false value  Version 7 AT&T UNIX
## fort77   FORTRAN77 programming   FORTRAN compiler    XPG4
## getopts  Shell programming   Parse utility options
## lex  C programming   Generate programs for lexical tasks     Version 7 AT&T UNIX
## logger   Shell programming   Log messages    4.3BSD
## nm   C programming   Write the name list of an object file   Version 1 AT&T UNIX
## printf   Shell programming   Write formatted output  4.3BSD-Reno
## read     Shell programming   Read a line from standard input
## sh   Shell programming   Shell, the standard command language interpreter    Version 7 AT&T UNIX (in
## sleep    Shell programming   Suspend execution for an interval   Version 4 AT&T UNIX
## strings  C programming   Find printable strings in files     2BSD
## strip    C programming   Remove unnecessary information from executable files    Version 1 AT&T UNIX
## tee  Shell programming   Duplicate the standard output   Version 5 AT&T UNIX
## test     Shell programming   Evaluate expression     Version 7 AT&T UNIX
## true     Shell programming   Return true value   Version 7 AT&T UNIX
## xargs    Shell programming   Construct argument lists and invoke utility     PWB UNIX
## yacc     C programming   Yet another compiler compiler   PWB UNIX
```

```
# finalmente borremos los nuevos archivos generados
rm *.cmds
```

1.14 El lenguaje de procesamiento de patrones AWK

AWK es un lenguaje de programación diseñado para procesar datos basados de texto, ya sean ficheros o flujos de datos. El nombre AWK deriva de las iniciales de los apellidos de sus autores: Alfred Aho, Peter Weinberger, y Brian Kernighan. *awk*, cuando está escrito todo en minúsculas, hace referencia al programa de Unix que interpreta programas escritos en el lenguaje de programación AWK. Es decir AWK es un lenguaje interpretado por el intérprete de comando *awk*.

AWK fue creado como un reemplazo a los algoritmos escritos en C para métodos de análisis de texto. Fue una de las primeras herramientas en aparecer en Unix (en la versión 3), que ganó popularidad rápidamente como una manera de añadir funcionalidad a las tuberías de Unix. Por ello se considera como una de las utilidades necesarias de todo sistema operativo Unix.

Debido a su densa notación, todos estos lenguajes son frecuentemente usados para escribir programas de una línea, como veremos seguidamente.

1.14.1 Estructura de los programas AWK

En general, a *awk* se le dan dos piezas de datos: un fichero de órdenes y un archivo primario de entrada.

Un fichero de órdenes (que puede ser un fichero real, o puede ser incluido en la invocación de *awk* desde la línea de comandos) contiene una serie de sentencias que le indican a *awk* cómo procesar el fichero de entrada.

El fichero primario de entrada es normalmente texto formateado de alguna manera, particularmente archivos con campos separados por tabuladores (tablas), puede ser en un fichero real, o puede ser leído por *awk* de la entrada estándar (teclado).

1.14.1.1 Estructura básica

- Un programa AWK típico consiste en una serie de líneas, cada una de la forma:

/patrón/ { acción }, donde la acción por defecto es imprimir {print}

donde patrón es una expresión regular y acción es una orden. La mayoría de las implementaciones de AWK usan expresiones regulares extendidas por defecto. AWK mira a lo largo del fichero de entrada; cuando encuentra una línea que coincide con el "patrón", ejecuta la (s) orden (es) indicadas en "acción".

- Para llamar a *awk* desde la línea de comandos, usaríamos una sintaxis de este tipo:

awk 'CODIGO AWK' ARCHIVO_A_PROCESAR

- para usarlo en una tubería de UNIX, conecta el STDOUT de un programa al STDIN de *awk* mediante |:

STDOUT_programaX | awk 'CODIGO AWK' > output_file.txt

1.14.1.2 Formas alternativas del código AWK:

BEGIN { acción }
Ejecuta las órdenes acción al comienzo de la ejecución, antes de que los datos comiencen a ser procesados.
END { acción }
Similar a la forma previa pero ejecuta las órdenes acción después de que todos los datos sean procesados.
/patrón/ Imprime las líneas acordes al patrón. { acción } Ejecuta acción por cada línea en la entrada.

Cada una de estas formas pueden ser incluidas varias veces en un archivo. El fichero es procesado de manera progresiva, entonces si hubiera dos declaraciones “BEGIN”, sus contenidos serán ejecutados en orden de aparición. Las declaraciones “BEGIN” y “END” no necesitan estar en forma ordenada.

1.14.1.3 Sintaxis condensada de AWK

- AWK, al ser un lenguaje de programación completo, contiene sintaxis para escribir:
  - condicionales y bucles for$ y $wile
  - operadores aritméticos +, -, *, /, %, =, ++, –, +=, -=, . . .)
  - operadores boleanos ||, &&
  - operadores relacionales <, <=, == !=, >=, >
  - funciones integradas: length(str); int(num); index(str1, str2); split(str,arr,del); sprintf(fmt,args); substr(str,pos,len); tolower(str); toupper(str)
  - funciones escritas por el usuario function FUNNAME (arg1, arg1){code}
  - estructuras de datos como arreglos asociativos (hashes o diccionarios): array[string]=value, entre otros.
- AWK Maneja también una serie de variables propias, de las que les resalto sólo las más usadas:

```
$0       guarda el valor de la fila actual en memoria de un archivo de entrada
$1-$n    guarda los contenidos de los campos de una fila
FILENAME nombre del archivo de entrada actualmente en procesamiento
FS       separador de campos (por defecto SPACE or TAB)
NR       guarda el número de campos delimitados por FS en registro o fila actual
OFS      separador de campo de la salida (SPACE por defecto)
ORS      separador de registro de la salida (\n por defecto)
```

1.14.2 Ejemplos básicos pero muy útiles de uso de AWK

No podemos aprender aquí más que algunos idiomas de AWK muy útiles, como veremos seguidamente

1.14.2.1 filtrado de archivos con AWK

- Cuenta el número de procesadores de tu sistema:

```
# el archivo /proc/cpuinfo contiene la información sobre las cpus del sistema, incluyendo los cores/pro
head -5 /proc/cpuinfo
```

```
## processor    : 0
## vendor_id    : GenuineIntel
## cpu family   : 6
## model        : 158
## model name   : Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
```

Recordemos la sintaxis general de código AWK: /patrón/ { acción }

```
# usamos el patrón '/^processor/, seguido de la acción {cuanta instancias} y terminamos con un bloque E
# este código de AWK se lo pasamos directamente al intérprete de comandos awk como un una cadena entre
# awk 'CODIGO AWK' ARCHIVO_A_PROCESAR
awk '/^processor/{n++} END{ print "This computer has", n, "processors"}' /proc/cpuinfo
```

```
## This computer has 12 processors
```

- Imprime líneas con 12 o menos caracteres de entre las primeras 20 líneas del archivo /proc/cpuinfo :

```
# con head -20 filtramos las primeras 20 líneas, las cuales pasamos a awk con |
# recuerda: la acción por defecto de awk es imprimir, en este caso las líneas que satisfagan la condici
head -20 /proc/cpuinfo | awk 'length <= 12'
```

```
## model        : 158
## core id      : 0
## apicid       : 0
## fpu      : yes
## wp       : yes
```

- Imprime líneas con 30 o más caracteres de entre las primeras 20 líneas del archivo /proc/cpuinfo :

```
head -20 /proc/cpuinfo | awk 'length >= 30'
```

```
## model name   : Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
## flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts a
```

1.14.2.2 filtrado de archivos separados por tabuladores (tablas) con AWK

- volvamos a asociar cada nombre de columna de la tabla assembly_summary.txt.gz con su número de campo, para recordarlos

```
# asocia cada nombre de columna de la cabecera con el número de la columna correspondiente
zcat assembly_summary.txt.gz | head -2 | sed '1d; s/\t/\n/g' | cat -n
```

```
##      1   # assembly_accession
##      2   bioproject
##      3   biosample
##      4   wgs_master
##      5   refseq_category
##      6   taxid
##      7   species_taxid
##      8   organism_name
##      9   infraspecific_name
##     10   isolate
##     11   version_status
##     12   assembly_level
##     13   release_type
##     14   genome_rep
##     15   seq_rel_date
##     16   asm_name
##     17   submitter
##     18   gbrs_paired_asm
##     19   paired_asm_comp
##     20   ftp_path
##     21   excluded_from_refseq
##     22   relation_to_type_material
```

- cuenta aquellas entradas de la tabla que tienen un número de accesión revisado v2, publicados en 2019 para genomas en estado Scaffold

```
# >>> ojo, es importante definer FS="\t", para que tome como campos sólo a aquellos separados por tabul
# ejemplo de código AWK con la estructura: BEGIN_BLOCK, condición, acción, END_BLOCK
# recuerden, como assembly_summary.txt.gz está comprimido, necesitamos zcat para poderlo leer y enviar
zcat assembly_summary.txt.gz | awk 'BEGIN{FS="\t"} $16 ~ /v2$/ {n++} END{print n}'
```

```
## 4488
```

- cuenta aquellas entradas de la tabla que tienen un número de accesión revisado v2, publicados en 2019 para genomas en estado Scaffold

```
# ejemplo de código AWK con la estructura: BEGIN_BLOCK, condición1 && condición2 && condición3, acción,
zcat assembly_summary.txt.gz | awk 'BEGIN{FS="\t"} $16 ~ /v2$/ && $15 ~ /2019/ && $12 == "Scaffold" {n+
```

```
## 31
```

- veamos las entradas de la tabla que tienen un número de accesión revisado v2, publicados en 2019 para genomas en estado Scaffold, pero imprime sólo los campos organism_name y ftp_path en formato tabla (OFS="\t"), imprimiendo sólo las primeras 3 líneas

```
# ejemplo de código AWK con la estructura: BEGIN_BLOCK, condición1 && condición2 && condición3, acción
zcat assembly_summary.txt.gz | awk 'BEGIN{FS="\t"; OFS="\t"} $16 ~ /v2$/ && $15 ~ /201./ && $12 == "Sca
```

```
## Leptospira interrogans   ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/002/370/085/GCF_002370085.2_ASM23
## Helicobacter pylori GAM100Ai ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/310/005/GCF_000310005.2_
## Helicobacter pylori GAM101Biv    ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/344/945/GCF_00034494
```

- veamos las entradas de la tabla que tienen un número de accesión revisado v2, publicados en 2019 para genomas en estado Scaffold, pero imprime sólo los campos organism_name y ftp_path separados por ' ~~~ ', imprimiendo sólo las primeras 3 líneas

```
# ejemplo de código AWK con la estructura: BEGIN_BLOCK, condición1 && condición2 && condición3, acción
zcat assembly_summary.txt.gz | awk 'BEGIN{FS="\t"; OFS=" ~~~ "} $16 ~ /v2$/ && $15 ~ /201./ && $12 == "
```

```
## Leptospira interrogans ~~~ ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/002/370/085/GCF_002370085.2_ASM
## Helicobacter pylori GAM100Ai ~~~ ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/310/005/GCF_00031000
## Helicobacter pylori GAM101Biv ~~~ ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/344/945/GCF_0003449
```

Felicidades, ya estás aprendiendo a programar. No es tan difícil, ¿verdad?

2 Ejercicios de exploración y parseo de archivos FASTA

Te propongo el siguiente ejercicio con un archivo de secuencias de DNA en formato FASTA para practicar algunos aspectos de lo aprendido en esta primera sesión.

Para correr los ejercicios, asegúrate de tener el archivo recA_Bradyrhizobium_vinuesa.fna en el directorio actual de trabajo.

El archivo recA_Bradyrhizobium_vinuesa.fna contiene secuencias del gen *recA* de bacterias del género *Bradyrhizobium* depositadas en GenBank por P. Vinuesa.

2.1 Búsqueda y descarga de secuencias en GenBank usando el sistema ENTREZ

Este bloque muestra el comando que usé para descargarlas usando el sistema ENTREZ de NCBI. El comando debe pegarse en la ventana superior del sistema ENTREZ.

```
# pega esta sentencia en la ventana de captura para interrogar la base de datos
# de nucleótidos de NCBI mediante el sistema ENTREZ
'Bradyrhizobium[orgn] AND vinuesa[auth] AND recA[gene]'
```

No hace falta que las descargues de NCBI. Para facilitar el acceso a las mismas, usa el siguiente código

2.1.1 Acceso a las secuencias

En primer lugar, debes estar en tu directorio $HOME/TIB2019-T3/sesion1_linux, y desde ahí generar una liga simbólica al archivo FASTA con las secuencias

```
cd $HOME/TIB2019-T3/sesion1_linu
ln -s /space31/PIG/vinuesa/TIB2019-T3/sesion1_Linux/data/recA_Bradyrhizobium_vinuesa.fna .

ls recA_Bradyrhizobium_vinuesa.fna
```

2.1.2 Inspección y estadísticas básicas de las secuencias descargadas

1. ¿Cuántas secuencias hay en el archivo recA_Bradyrhizobium_vinuesa.fna?
2. Explora la cabecera y cola del archivo con head y tail
3. Despliega las 5 primeras lineas de cabeceras fasta usando **grep** y **head** para explorar su estructura en detalle
4. Calcula el número de generos que contiene el archivo FASTA
5. Calcula el número de especies que contiene el archivo FASTA
6. Imprime una lista ordenada de mayor a menor, del numero de especies que contiene el archivo FASTA

2.1.3 Edición de las cabeceras FASTA mediante herramientas de filtrado de UNIX

1. Explora nuevamente todas las cabeceras FASTA del archivo recA_Bradyrhizobium_vinuesa.fna usando **grep** y less
2. Simplifica las cabeceras FASTA usando el comando **sed** (stream editor)

El objetivo es eliminar redundancia y los campos gb|no.de.acceso, así como todos los caracteres '(, ; :)' que impedirían el despliegue de un árbol filogenético, al tratarse de caracteres reservados del formato NEWICK. Dejar solo el numero GI, así como el género, especie y cepa indicados entre corchetes.

Es decir vamos a: - reducir Bradyrhizobium a 'B.' - eliminar ' RNA poly . . . ' y reemplazarlo por ']' - eliminar 'genosp.' - sustituir espacios por guiones bajos

Nota: hagan uso de expresiones regulares como '.*' y '[[:space:]]'

3. Cuando estén satisfechos con el resultado, guarden la salida del comando en un archivo llamado recA_Bradyrhizobium_vinuesa.fnaed