

Introduction to Unix command-line tools – Part 2

Gladstone Institutes

Leandro Lima

Bioinformatics Core

Institute of Data Science and Biotechnology

Finkbeiner Lab

Center for Systems and Therapeutics

November 5, 2019

tar - archiving

Create an archive:

```
tar -cvf newfile.tar file1 file2 dir1 dir2
```

```
tar -cvf BLs.tar bla.txt ble.txt blo.txt
```

```
tar -cvzf BLs.tar.gz bla.txt ble.txt blo.txt
```

Parameters: c (create), v (verbose), z (gzip), f (file)

tar - archiving

Extract from an archive:

```
tar -xvzf data.tar.gz
```

```
tar -xvjf XHMM_results.tar.bz2
```

Parameters: x (extract), v (verbose), f (file),
z (gzip), j (bzip2)

Let's download some data to play with

```
cd
```

```
mkdir workshop
```

```
cd workshop
```

```
wget finkbeiner-biowww.gladstone.org/data.tar.gz
```

```
curl -O finkbeiner-biowww.gladstone.org/data.tar.gz
```

```
tar -xvzf data.tar.gz
```

```
cd data
```

less - file visualization

```
less VCP.txt
```

- Use arrows (←↑→↓) to navigate the file
- Type / to search

```
less -S VCP.txt
```

file slicing - head, tail, cut



head - first lines

first 20 lines

head -n ATXN2.txt

all lines, excluding last 2

(on Linux, not Mac)

head -n -2 ATXN2.txt

tail - last lines

last 20 lines

```
tail -n 20 ATXN2.txt
```

last lines, starting from line 2

```
tail -n +2 ATXN2.txt
```


cut - get specific columns of file

```
# fields 1 to 3, 5 and 6
```

```
cut -f 1-3,5,6 ATXN2.txt
```

```
# other examples
```

```
cut -f1,2 -d, var_counts.csv # delimiter = comma
```

```
# other delimiters: space, semi-colon, etc.
```

```
cut -d' ' -f1-2 ...
```

```
cut -d';' -f5,7,9 ...
```

Using "|" (pipe) to join commands

```
cut -f 1-3,5,6 TBK1.txt | head -n 5
```

```
cut -f 1-3,5,6 UNC13A.txt | less
```

```
tail -n +2 OPTN.txt | cut -f3 | sort | uniq -c
```

column - columnate lists

using white spaces to separate

and fill columns

column -t # print columns

column -s # choose separator

cut -f1-3 -d, var_counts.csv |

column -t -s,

sort - sort lines of text files

```
sort file.txt
```

```
sort -k : choose specific field
```

```
sort -n : numeric-sort
```

```
sort -r : reverse
```

```
# Exercise: show 3 first columns of  
# ATXN2.txt and sort by 3rd column
```

uniq - report or filter out repeated lines in a file

```
cut -f3 ATXN2.txt | sort | uniq
```

```
# reporting counts of each line
```

```
cut -f3 ATXN2.txt | sort | uniq -c
```

```
# Exercise: find the most common variant
```

```
# annotation (column 3) in ATXN2.txt
```

wc - word, line, character and byte count

wc -l : number of lines

wc -w : number of words

wc -m : number of characters

wc FUS.txt

wc -l *.txt

cut -f3 ATXN2.txt | sort | uniq | wc -l

grep - finds words/patterns in a file (i)

grep [word] [file.txt]

Options:

grep -w : find the whole word

grep -c : returns the number of lines found

grep -f : specifies a file with a list of words

grep -o : returns only the match

grep - finds words/patterns in a file (ii)

grep -A 2 : also show 2 lines after

grep -B 3 : also show 3 lines before

grep -v : shows lines without pattern

grep --color : colors the match

grep - finds words/patterns in a file (iii)

Exercises:

- Show the chromosome and position (cols. 5 and 6) of any nonsynonymous variants in FUS
- Show the same in all the genes (*.txt)
- Show the chromosome and position (cols. 5 and 6) of any variant not matching “nonsynonymous” FUS

awk - a powerful way to check conditions and show specific columns

Examples:

```
# Finding all variants with
```

```
# REF = 'A' (col 7) and ALT = 'G' (col 8)
```

```
awk ' $7 == "A" && $8 == "G" ' *txt | less -S
```

```
# Finding all variants in chrms. 1 to 5
```

```
cat *txt | awk '$5 < 6' | cut -f1-3,5,6
```

awk - different ways to do the same thing

```
cat *txt | awk '$5 < 6' | cut -f5,6
```

```
# same effect 1
```

```
cat *txt | awk '$5 < 6 {print $5" "$6}'
```

```
# same effect 2
```

```
cat *txt | awk 'if ($5 < 6) {print $5" "$6}'
```

Exercises

1. How many variants are located on chrom. 1?
2. How many splicing variants are there?
3. Which gene has more “stopgain” variants?

awk - more options on if statement and using external variables

```
# Showing first row and the variant functions with more  
# than 100 and less than 1000 variants
```

```
awk -F',' 'NR == 1 || ($2 > 100 && $2 < 1000) \  
          {print $1}' var_counts.csv
```

```
# Using external variables
```

```
awk -F',' -v k=1000 \  
  'NR == 1 || ($2 > 0.1*k && $2 < k) \  
    {print $1}' var_counts.csv
```

in-line Perl/sed to find and replace (i)

```
cut -f1-3 UBQLN2.txt | perl -pe 's/exonic/Exonic/g'
```

```
cut -f1-3 UBQLN2.txt | perl -pe 's/_/ /g'
```

```
cut -f1-3 UBQLN2.txt | perl -pe 's/\t/,/g'
```

```
cut -f1-3 UBQLN2.txt | perl -pe 's/\t/,/g; s/_/ /g'
```

Other possibilities

```
cut -f1-3 UBQLN2.txt | perl -pe 's!_! !g'
```

```
cut -f1-3 UBQLN2.txt | perl -pe 's|_| |g'
```

Creating a file with genomics locations

```
cut -f1-3 UBQLN2.txt | perl -pe 's/\t/:/g; s/^/chr/g'
```

in-line Perl/sed to find and replace (ii)

```
# "s" means substitute
```

```
# "g" means global (replace all matches, not only first)
```

```
# See the difference...
```

```
cut -f5,6 SOD1.txt | sed 's/0/zero/g'
```

```
cut -f5,6 SOD1.txt | sed 's/0/zero/'
```

copy from terminal to clipboard/ paste from clipboard to terminal

This is like Ctrl+V in your terminal

pbpaste

This is like Ctrl+C from your terminal

cut -f1 -d, var_counts.csv | pbcopy

Then, Ctrl+V in other text editor

On Linux, you can install "xclip"

<http://sourceforge.net/projects/xclip/>

Variables (i)

```
i=1
```

```
name=Leandro
```

```
count=`wc -l OPTN.txt`
```

```
echo $i
```

```
echo $name
```

```
echo $count
```

Variables (ii)

Examples

bwa=/home/users/llima/tools/bwa

hg19=/references/hg19.fasta

Do not run

\$bwa index \$hg19

System variables

```
echo $HOME
```

```
echo $USER
```

```
echo $PWD
```

```
# directory where bash looks for your programs
```

```
echo $PATH
```

Running a bash script (i)

```
# using cat as a text editor
```

```
cat > arguments.sh
```

```
echo Your program is $0
```

```
echo Your first argument is $1
```

```
echo Your second argument is $2
```

```
echo You entered $# parameters.
```

```
# Finally, press Ctrl+C to exit "cat"
```

Running a bash script (ii)

```
bash arguments.sh
```

```
bash arguments.sh A B C D E
```

chmod - set permissions (i)

```
ls -lh arguments.sh
```

```
-rw-r--r--
```

First character

b	Block special file.
c	Character special file.
d	Directory.
l	Symbolic link.
s	Socket link.
p	FIFO.
-	Regular file.

chmod - set permissions (ii)

Next characters

user, group, others | **read, write, execute**

ls -lh arguments.sh

-rw-r--r--

Everybody can read

Only user can write/modify

chmod - set permissions (iii)

Add writing permission to group

```
chmod g+w arguments.sh
```

```
ls -lh arguments.sh
```

Remove writing permission from group

```
chmod g-w arguments.sh
```

```
ls -lh arguments.sh
```

Add execution permission to all

```
chmod a+x arguments.sh
```

```
ls -lh arguments.sh
```


Run your program again

```
# Add writing permission to group
./arguments.sh
./arguments.sh A B C D E
# change the name
mv arguments.sh arguments
# Send to your PATH (showing on Mac)
sudo cp arguments /usr/local/bin/
# Go to another directory
# Type argu<Tab>, and "which arguments"
```

vi/vim (text editor)

(i)

vi text_file.txt (open "text_file.txt")

i - start edition mode (remember "insert")

ESC - stop edition mode

:w - save file ("write")

:q - quit

:x - save (write) and quit

vi/vim (text editor)

(ii)

u - undo

:30 - go to line number 30

:syntax on - syntax highlighting

^ - go to beginning of line

\$ - go to end of line

vi/vim (text editor)

(iii)

dd - delete current line

d2↓ - delete current line and 2 lines below

yy - copy current line

y3↓ - copy current line and 3 lines below

pp - paste lines below current line

Survey

<https://bioinformatics-course-feedback.questionpro.com/>

End