

Day 2: Advanced Command line

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu
2022/03/30

Agenda

- Day 0: Preparation
- Day 1: Introduction to Unix Command line
 - Navigating through file system
 - Manipulating files and directories
- Day 2: Advanced **Command line**
 - Useful commands for file processing
 - **sed**, **awk**, **grep**, and **regular expression**
- Day 3: **Shell scripting** and running jobs on hoffman2



Notations of the slides

- Code chunk starts with "➤", e.g.
➤ echo 'Hello world!'
- Link is underlined
- Practice comes with



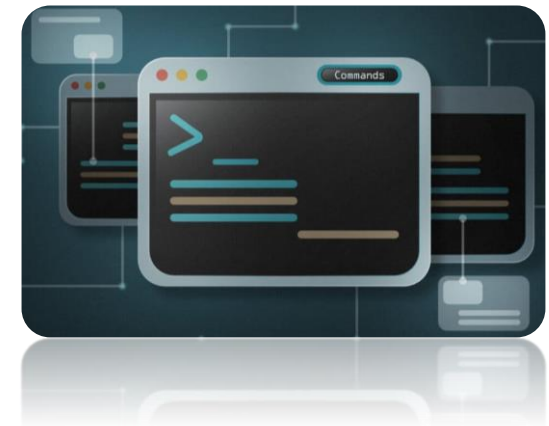
Overview

Time

- 3-hour workshop (45min + 45min + 30min + practices/Q&A)

Topics

- ☐ Request resources on hoffman2 cluster
- ☐ Useful commands for file processing
- ☐ sed, awk, grep
- ☐ Regular expression

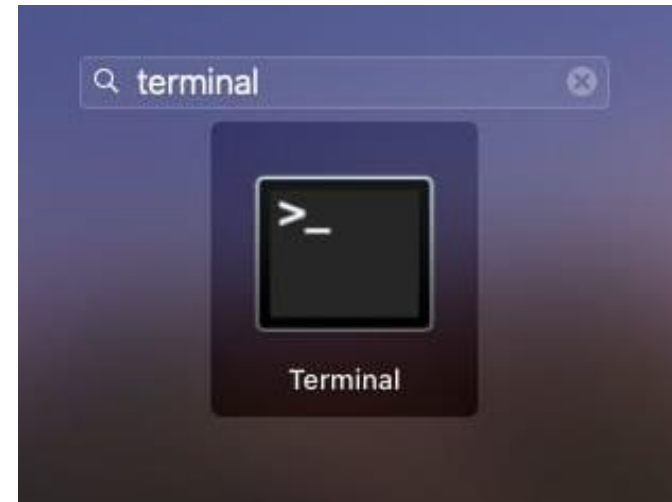
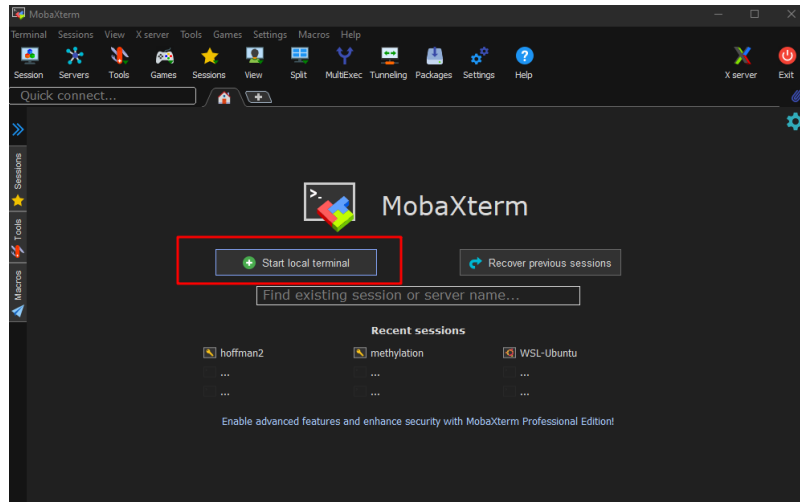


Questions can be put into this [Google doc](#)

Summary – Day 1

Connect to hoffman2 cluster through **command line interface**

- Windows: mobaXterm/WSL open local **terminal**
- macOS/Linux: open **terminal**



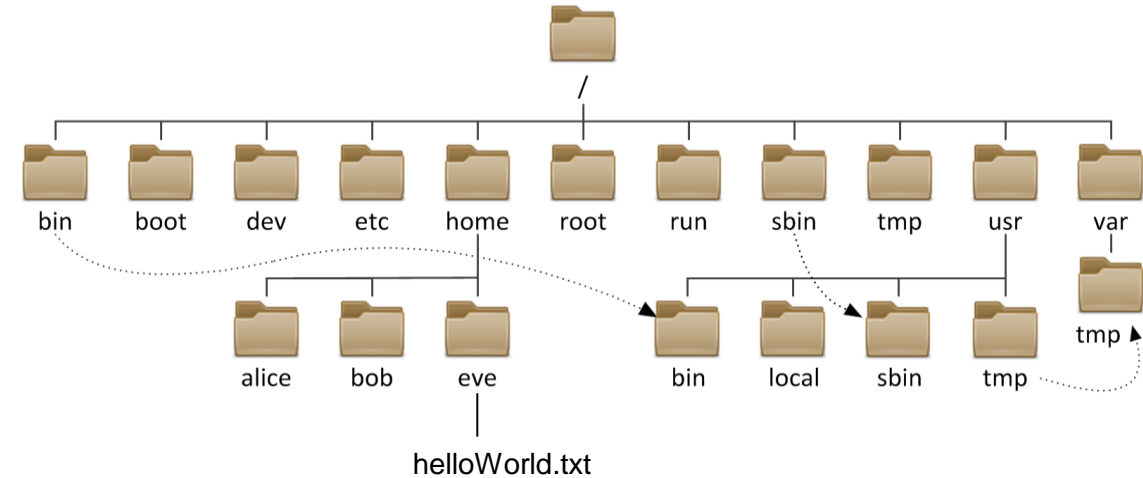
➤ `ssh username@hoffman2.idre.ucla.edu`
(replace **username** with your hoffman2 username, e.g. *wbguo*)

Summary – Day 1

Navigate through the file system

- **Absolute path** & **relative path**
- The meaning of “**~**”, “**.**”, “**..**”

Command	Function
ssh	Connect to hoffman2
exit	Close the remote connection
pwd	Print working directory
cd	Change directory
ls	List the information of files/folder (note the flag's usage)



Tips:

- Use up/down key to browse your command history
- Use Tab key for auto-completion
- Use clear to clean screen
- Use pwd and ls to check

Summary – Day 1

Manipulate with files/directories

Command	Function
mkdir	Create a directory
touch	Create a file / change the time stamp
rm	Delete file or directory
cp	Copy file or directory
mv	Move file or directory
cat	Print the file contents to standard output
vi	Modify a file / create a file
chmod	Change file permission
gzip	Compress/decompress a file
tar	Create a compressed archive
scp/sftp	File transfer
man/ --help	Get help

Note:

- command is **case-sensitive**
- **cat** file to show the contents

Let's fight with the dragon!



Request resources on hoffman2

- Connect to hoffman2

- `ssh username@hoffman2.idre.ucla.edu`

- Request a interactive node

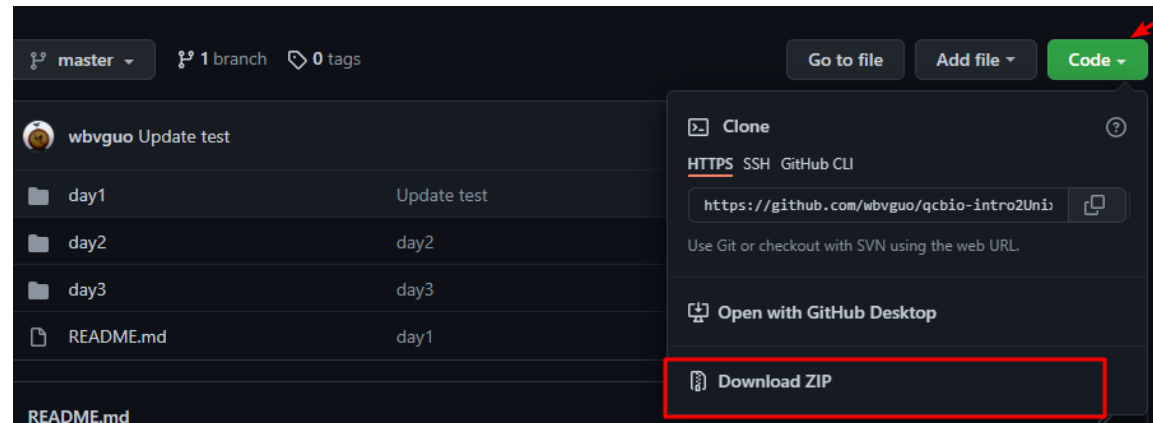
- `qrsh -l h_rt=03:00:00,h_data=256M`

Note:

- **No space** between h_rt and h_data (they are only **separated by comma**)
- Default h_rt (02:00:00), h_data (1G)
- Make sure to request enough resources for your computing jobs

Get working materials

- Option1: Clone the working materials to your directory
 - `git clone https://github.com/wbvguo/qcbio-intro2Unix.git`
(if meet “already exists” problem, try `rm -rf qcbio-intro2Unix` first)
- Option2: download the file and upload to hoffman2
 - Go to <https://github.com/wbvguo/qcbio-intro2Unix.git>
 - Download zip file



Get working materials

- Option1: Clone the working materials to your directory
 - `git clone https://github.com/wbvguo/qcbio-intro2Unix.git`
(if meet “already exists” problem, try `rm -rf qcbio-intro2Unix` first)
- Option2: download the file and upload to hoffman2
 - Go to <https://github.com/wbvguo/qcbio-intro2Unix.git>
 - Download zip file
 - Upload to hoffman2 using scp or Cyberduck/mobaXterm



Useful commands for file processing

- View/merge
- Sort
- Redirect/pipeline
- Other command lines (wc/cut/uniq/diff)

“Everything is a file in Unix”

View a file

- Regular file
 - cat
 - more/less
 - head/tail
- Compressed file
 - zcat
 - zmore

View (cat)

- **cat**: concatenate files and print on the standard output
- Syntax: `cat <file name>`

[illegible]

- It's **not suitable** to view very big file

View (more)

- **more**: view the file contents one screen at a time
- Syntax: `more <file name>`

Scroll option

- By line: enter
- By screen: space

Exit Option:

- q (quit)

```
[wbguo@login1 day2]$ ls
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10  sample-A01.gz  seq_file_list.txt
[wbguo@login1 day2]$ more hg38_chr.fai
chr1      248956422      8          60          61
chr2      242193529      253105712   60          61
chr3      198295559      499335808   60          61
chr4      190214555      700936301   60          61
chr5      181538259      894321107   60          61
chr6      170805979      1078885012  60          61
chr7      159345973      1252537766  60          61
chr8      145138636      1414539514  60          61
chr9      138394717      1562097136  60          61
chr10     133797422      1702798442  60          61
chr11     135086622      1838825832  60          61
chr12     133275309      1976163908  60          61
chr13     114364328      2111660483  60          61
chr14     107043718      2227930894  60          61
chr15     101991189      2336758684  60          61
chr16     90338345       2440449737  60          61
chr17     83257441       2532293732  60          61
chr18     80373285       2616938808  60          61
```

View (less)

- **less**: display the file in a paginated way
- Syntax: `less <file name>`

```
chr1 248956422      8      60      61
chr2 242193529     253105712    60      61
chr3 198295559     499335808    60      61
chr4 190214555     700936301    60      61
chr5 181538259     894321107    60      61
chr6 170805979     1078885012   60      61
chr7 159345973     1252537766   60      61
chr8 145138636     1414539514   60      61
chr9 138394717     1562097136   60      61
chr10 133797422    1702798442   60      61
chr11 135086622    1838825832   60      61
chr12 133275309    1976163908   60      61
chr13 114364328    2111660483   60      61
chr14 107043718    2227930894   60      61
chr15 101991189    2336758684   60      61
chr16 90338345     2440449737   60      61
chr17 83257441     2532293732   60      61
chr18 80373285     2616938808   60      61
chr19 58617616     2698651658   60      61
chr20 64444167     2758246245   60      61
hg38_chr.fai
```

Scroll option

- By line: up/down key
- By page: f or space / b (forward / backward);
PgDn/PgUp

Exit Option:

- q (quit)

Search option:

- /string: search string forward
- ?string: search string backward
- n: next occurrence
- N: previous occurrence

View (head)

- **head**: show the beginning of file
- Syntax: `head -<flag> <file name>`
- Flag option:
 - `-n`: show the first n line of file

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt
[wbguo@login1 day2]$ head -10 hg38_chr.fai
chr1      248956422      8      60      61
chr2      242193529      253105712      60      61
chr3      198295559      499335808      60      61
chr4      190214555      700936301      60      61
chr5      181538259      894321107      60      61
chr6      170805979      1078885012      60      61
chr7      159345973      1252537766      60      61
chr8      145138636      1414539514      60      61
chr9      138394717      1562097136      60      61
chr10     133797422      1702798442      60      61
```

View (tail)

- **tail**: show the end of the file
- Syntax: `tail -<flag> <file name>`
- Flag option:
 - `-n`: show the last n line of file

```
[wbguo@login1 day2]$ tail -10 hg38_chr.fai
KI270748.1      93321   3150532617    60    61
KI270749.1     158759   3150627517    60    61
KI270750.1     148850   3150788945    60    61
KI270751.1     150742   3150940299    60    61
KI270752.1      27745   3151093577    60    61
KI270753.1      62944   3151121808    60    61
KI270754.1      40191   3151185825    60    61
KI270755.1      36723   3151226709    60    61
KI270756.1      79590   3151264068    60    61
KI270757.1      71251   3151345008    60    61
```

View (zcat/zmore)

- **zcat**: Decompress and concatenate files to standard output
- Syntax: `zcat <file name>.gz`
- **zmore**: Decompress and view the file contents one screen at a time
- Syntax: `zmore <file name>.gz`

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt
[wbguo@login1 day2]$ zmore sample-A01.gz
-----> sample-A01.gz <-----
chr3    G      13080   CHH    CA      0.0    0      1
chr3    G      13083   CHH    CT      0.0    0      1
chr3    G      13089   CHG    CA      0.0    0      1
chr3    G      13090   CHH    CC      0.0    0      1
chr3    G      13091   CHH    CC      0.0    0      1
chr3    G      13095   CHG    CT      0.0    0      1
chr3    G      13096   CHH    CC      0.0    0      1
chr3    G      13098   CHH    CA      0.0    0      1
chr3    G      13101   CHG    CT      0.0    0      1
```

Merge (cat)

- **cat**: concatenate files and print on the standard output
- Syntax: `cat <file1> <file2>`

[illegible]

- **Question:** does the order matter?

Tips: using * as a wildcard in commands

- `cat chr*` concatenates files whose name starts with chr
 - Same results as `cat chr1_10 chr2_10`
- `ls *10` outputs files end with 10

```
[wbguo@login1 day2]$ ls
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10  sample-A01.gz  seq_file_list.txt
[wbguo@login1 day2]$ ls *10
chr1_10  chr2_10  mod_chr1_10
[wbguo@login1 day2]$
```

- `ls *chr*` outputs files that contains chr

```
[wbguo@login1 day2]$ ls
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10  sample-A01.gz  seq_file_list.txt
[wbguo@login1 day2]$ ls *chr*
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10
[wbguo@login1 day2]$
```

Practice:



- Try less command with file `hg38_chr.fai`
 - Scroll by line/page
 - Search string 'chr'
- Merge `mod_chr1_10` and `chr1_10`

Redirection

- Redirecting the standard output to a file
 - **>**: output to the file (if file exists, it will overwrite it)
 - **>>**: append the output to the end of a file (if file doesn't exist, it will create the file)

```
[wbguo@login1 day2]$ ls
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10  sample-A01.gz  seq_file_list.txt
[wbguo@login1 day2]$ echo "this is day2" > test.txt
[wbguo@login1 day2]$ ls
chr1_10  chr2_10  hg38_chr.fai  mod_chr1_10  sample-A01.gz  seq_file_list.txt  test.txt
[wbguo@login1 day2]$ cat test.txt
this is day2
[wbguo@login1 day2]$ echo "add another line" >> test.txt
[wbguo@login1 day2]$ cat test.txt
this is day2
add another line
[wbguo@login1 day2]$
```

Pipeline



- | (pipe): combine multiple commands together
- Syntax: `command1 | command2 | command3`
- The `command1`'s output will be redirected to `command2`'s input, and so on

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt test.txt
[wbguo@login1 day2]$ zcat sample-A01.gz | head -10
chr3 G 13080 CHH CA 0.0 0 1
chr3 G 13083 CHH CT 0.0 0 1
chr3 G 13089 CHG CA 0.0 0 1
chr3 G 13090 CHH CC 0.0 0 1
chr3 G 13091 CHH CC 0.0 0 1
chr3 G 13095 CHG CT 0.0 0 1
chr3 G 13096 CHH CC 0.0 0 1
chr3 G 13098 CHH CA 0.0 0 1
chr3 G 13101 CHG CT 0.0 0 1
chr3 G 13103 CHH CA 0.0 0 1
```

Enhance the efficiency:

- Avoid generating the intermediate files

Sort

- **sort**: sort lines in a file
- Syntax: `sort -<flag> <filename>`
- Flag options
 - **-k** col1,col2: Specify a **key** to do the sorting, col1 and col2 are used to indicate the starting field indices and ending field indices (col1 default is 1, col2 can be empty)
 - **-t**: specify the delimiter
 - **-n**: sort based on **n**umerical value (default: alphabetically)
 - **-r**: **r**everse order
 - **-u**: only unique lines
 - **-b**: Ignore blanks at the start of the line
 - **-o**: specify the output file (default is the standard output)

Sort

- **sort**: sort lines in a file
- Syntax: `sort -<flag> <filename>`

original

```
[wbguo@login1 day2]$ head -10 hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712  60      61
chr3    198295559     499335808  60      61
chr4    190214555     700936301  60      61
chr5    181538259     894321107  60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
```

sorted

```
[wbguo@login1 day2]$ sort -k 2 -n hg38_chr.fai | head -10
KI270394.1    970      3142567409    60      61
KI270392.1    971      3142565045    60      61
KI270423.1    981      3142587281    60      61
KI270385.1    990      3142554003    60      61
KI270539.1    993      3143532085    60      61
KI270312.1    998      3142429913    60      61
KI270336.1   1026     3142507546    60      61
KI270419.1   1029     3142582335    60      61
KI270329.1   1040     3142499492    60      61
KI270379.1   1045     3142543110    60      61
```

Count

- **wc**: **W**ord **c**ount for the file
- Syntax: `wc -<flag> <file name>`
- Flag options:
 - **-m**: show total number of characters
 - **-l**: show total number of **l**ines
 - **-w**: show total number of **w**ords

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt test.txt
[wbguo@login1 day2]$ wc -m chr1_10
557 chr1_10
[wbguo@login1 day2]$ wc -l chr1_10
10 chr1_10
[wbguo@login1 day2]$
```

Cut

- **cut**: show the specific parts of the file;
- Syntax:
 - `cut -c <char position> <filename>`
 - `cut -d <delimiter> -f <field position> <filename>`
- Flag options:
 - `-c <char position>`: select only these characters
 - `-d <delimiter>` : separate using delimiter (default tab)
 - `-f <field position>`: extract these fields

Cut

- **cut**: show the specific parts of the file;
- Syntax:
 - `cut -c <char position> <filename>`
 - `cut -d <delimiter> -f <field position> <filename>`

```
[wbguo@login1 day2]$ head -5 seq_file_list.txt
sample-A01_R1_trimmed.fastq.gz
sample-A01_R2_trimmed.fastq.gz
sample-A02_R1_trimmed.fastq.gz
sample-A02_R2_trimmed.fastq.gz
sample-A03_R1_trimmed.fastq.gz
[wbguo@login1 day2]$ cut -c 1-10 seq_file_list.txt | head -5
sample-A01
sample-A01
sample-A02
sample-A02
sample-A03
```

Deduplicate

- **uniq**: extract **unique** lines
- Syntax: `uniq -<flag> <filename>`
- Flag options:
 - **-c**: count how many times each line occurred
 - **-d**: show only duplicated lines

Deduplicate

- **uniq**: extract **unique** lines
- Syntax: `uniq -<flag> <filename>`

```
[wbguo@login1 day2]$ cut -c 1-10 seq_file_list.txt
sample-A01
sample-A01
sample-A02
sample-A02
sample-A03
sample-A03
sample-A04
sample-A04
sample-A05
sample-A05
sample-A06
sample-A06
sample-A07
sample-A07
sample-A08
sample-A08
sample-A09
sample-A09
sample-A10
sample-A10
sample-A11
sample-A11
sample-A12
sample-A12
```

```
[wbguo@login1 day2]$ cut -c 1-10 seq_file_list.txt | uniq
sample-A01
sample-A02
sample-A03
sample-A04
sample-A05
sample-A06
sample-A07
sample-A08
sample-A09
sample-A10
sample-A11
sample-A12
```

```
[wbguo@login1 day2]$ cut -c 1-10 seq_file_list.txt | uniq -c
 2 sample-A01
 2 sample-A02
 2 sample-A03
 2 sample-A04
 2 sample-A05
 2 sample-A06
 2 sample-A07
 2 sample-A08
 2 sample-A09
 2 sample-A10
 2 sample-A11
 2 sample-A12
```

Difference

- **diff**: show differences between files
- Syntax: `diff -<flag> <filename1> <filename2>`
- Flag options:
 - **-c**: output with copied context
 - **-u**: output with unified context
 - **-y**: side by side

[illegible]

Practice:



- Perform sorting based on column 3 (numerically decreasing) for file `sample-A01.gz`
- Using `cut` to extract the sample id for file `seq_file_list.txt`
 - e.g. if the file name is `sample-A01_R1_trimmed.fastq.gz`, sample id is `A01`

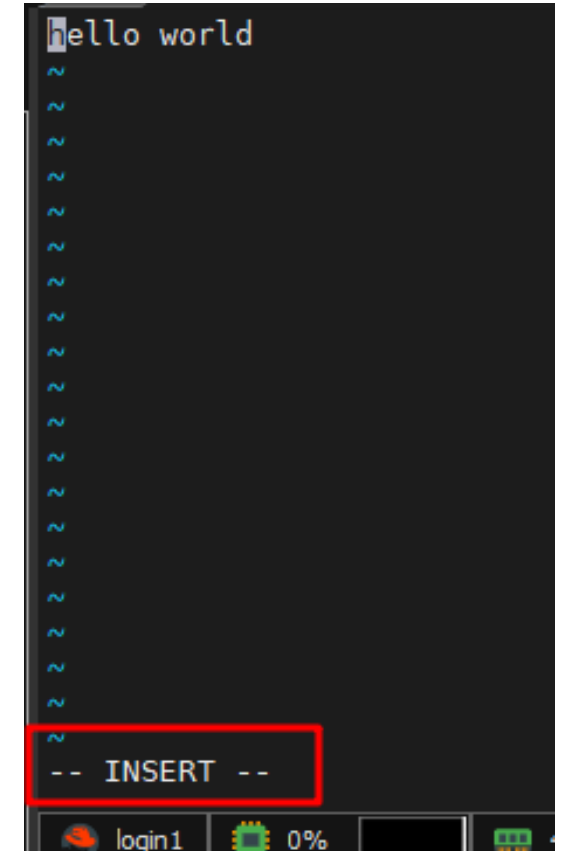
Advanced Command lines

- sed: non-interactively file editing
- awk: scripting language for text processing
- grep: search string/pattern in the file

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8        60      61
chr2    242193529      253105712 60      61
chr3    198295559      499335808 60      61
chr4    190214555      700936301 60      61
chr5    181538259      894321107 60      61
chr6    170805979      1078885012 60      61
chr7    159345973      1252537766 60      61
chr8    145138636      1414539514 60      61
chr9    138394717      1562097136 60      61
chr10   133797422      1702798442 60      61
chr11   135086622      1838825832 60      61
chr12   133275309      1976163908 60      61
chr13   114364328      2111660483 60      61
chr14   107043718      2227930894 60      61
chr15   101991189      2336758684 60      61
chr16   90338345       2440449737 60      61
chr17   83257441       2532293732 60      61
chr18   80373285       2616938808 60      61
chr19   58617616       2698651658 60      61
chr20   64444167       2758246245 60      61
chr21   46709983       2823764492 60      61
chr22   50818468       2871252985 60      61
chrX    156040895      2922918436 60      61
chrY    57227415       3081560021 60      61
chrM    16569          3139741236 60      61
GL000008.2 209709 3139758105 60      61
GL000009.2 201709 3139971333 60      61
GL000194.1 191469 3140176427 60      61
```

vi: an interactive editor

- Press "**i**" to type in new text (**i**nsertion mode)
- Press "**Esc**" to jump out the insertion mode
- Enter
 - **:w** (**w**rite)
 - **:q** (**q**uit)
 - **:wq** (**w**rite and **q**uit)
 - **:q!** (force **q**uit – the modification will not be saved)



vi: an interactive editor

Several scenarios **when vi is not suitable**

- Files are too large for interactive editing
 - It could be very slow to load them into memory
- Editing is too complicated and tedious
 - After all, you are editing file using hand

sed is very fast and uses little memory,
it can perform multiple editing in one pass

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529    253105712    60      61
chr3    198295559    499335808    60      61
chr4    190214555    700936301    60      61
chr5    181538259    894321107    60      61
chr6    170805979    1078885012    60      61
chr7    159345973    1252537766    60      61
chr8    145138636    1414539514    60      61
chr9    138394717    1562097136    60      61
chr10   133797422    1702798442    60      61
chr11   135086622    1838825832    60      61
chr12   133275309    1976163908    60      61
chr13   114364328    2111660483    60      61
chr14   107043718    2227930894    60      61
chr15   101991189    2336758684    60      61
chr16   90338345     2440449737    60      61
chr17   83257441     2532293732    60      61
chr18   80373285     2616938808    60      61
chr19   58617616     2698651658    60      61
chr20   64444167     2758246245    60      61
chr21   46709983     2823764492    60      61
chr22   50818468     2871252985    60      61
chrX    156040895     2922918436    60      61
chrY    57227415     3081560021    60      61
chrM    16569      3139741236      60      61
GL000008.2    209709    3139758105    60      61
GL000009.2    201709    3139971333    60      61
GL000194.1    191469    3140176427    60      61
```

sed

- **sed**: stream **e**ditor (parse and transform text)
- Syntax: `sed -<flag> 'sed commands' <filename>`
- Editing task: insert / append/ delete / substitute / modify / subset
- Flag options:
 - **-i**: in place (default is to output to standard output)
 - **-e**: expression (useful for using multiple sed commands at once)

sed - insert

- Syntax: `sed 'x,y i pattern' <filename>`
- Insert pattern **before** each line between x and y

[illegible]

sed - insert

- Syntax: `sed '/pattern1/ i pattern2' <filename>`
- Insert pattern2 **before** each line when pattern1 is found

[illegible]

sed - append

- Syntax: `sed 'x,y a pattern' <filename>`
- Insert pattern **after** each line between x and y

[illegible]

sed - append

- Syntax: `sed '/pattern1/ a pattern2' <filename>`
- Insert pattern2 **after** each line when pattern1 is found

[illegible]

sed - delete

- Syntax: `sed 'x,y d' <filename>`
- delete lines between x and y

[illegible]

sed - delete

- Syntax: `sed '/pattern/ d' <filename>`
- delete lines when pattern is found

[illegible]

sed - substitute

- Syntax: `sed 's/pattern1/pattern2/' <filename>`
- The substitution only apply to the first match

[illegible]

sed - substitute

- Syntax: sed '**s**/pattern1/pattern2/' <filename>
 - The substitution only apply to the first match
-
-
-
-
-
-
-
-
-
-
- Syntax: sed '**s**/pattern1/pattern2/**g**' <filename>
 - Use **g** (**g**lobally) to substitute all the matches

[illegible]

sed - substitute

- Syntax: `sed 'x,y s/pattern1/pattern2/' <filename>`
- Restrict the substitution between line x and y
 - If set y as **\$**, it means the end of the file

[illegible]

sed – modify a specific line

- Syntax: `sed 'xc\your_new_line' <filename>`
- Change the specific x-th line

[illegible]

sed – subset the data

- Syntax example: `sed -n 'x,y'p' <filename>`
- Print the file from x-th line to y-th line (both x,y included)

[illegible]

sed – execute multiple commands in one pass

- Syntax example:

- `sed -e 'x,y d' -e 's,t a pattern' <filename>`

```
[wbguo@login1 day2]$ sed -e '1,2 i new_line' -e '3c\Hello world!' mod_chr1_10
new_line
>chr1 chr1
new_line
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
Hello world!
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNN_UNIX_IS_COOL_NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

Suggestions

- Do not read and write in the same file

✗ • `sed 'command' file.txt > file.txt`

✓ • `sed -i 'command' file.txt`

- It's better to backup a file when you use '`sed -i`'

Practice:



- Copy chr1_10 file as chr1_test1, try the following command
 - `sed '1,2 i oops' chr1_test1 > chr1_test1`
 - `cat chr1_test1`
- Copy chr1_10 file as chr1_test2, try the following command
 - `sed -i '1,2 i oops' chr1_test2`
 - `cat chr1_test2`

What did you find?

awk

- **awk**: reads a file line by line, parses the data into fields and allows complex operations to be applied for each line of data
- Syntax: `awk -<flags> 'condition {operation}' <file name>`
- **Conditions:**
 - Search pattern: `/pattern/`
 - Comparison: `a>b`, `a==b`, `a!=b`, ...
 - Keyword: `BEGIN`, `END`
- **Operations:**
 - Output: e.g. `print`
 - Arithmetic operation: e.g. `c=a*b`
 - More complex logic: e.g. `if-else`

Flag options:

- `-F`: to specify a delimiter between columns (default is space/tab)
- `-v var=value`: To declare a variable.

awk – print out certain column

- Syntax: `awk '{operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

Diagram illustrating field numbering for the first line of the file:

\$1	\$2	...
\$0		

- Print column1
 - `awk '{print $1}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{print $1}' hg38_chr.fai | head -10
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
[wbguo@login1 day2]$
```

awk – print out certain column

- Syntax: `awk '{operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1      248956422      8      60      61
chr2      242193529      253105712      60      61
chr3      198295559      499335808      60      61
chr4      190214555      700936301      60      61
chr5      181538259      894321107      60      61
chr6      170805979      1078885012      60      61
chr7      159345973      1252537766      60      61
chr8      145138636      1414539514      60      61
chr9      138394717      1562097136      60      61
chr10     133797422      1702798442      60      61
chr11     135086622      1838825832      60      61
chr12     133275309      1976163908      60      61
chr13     114364328      2111660483      60      61
chr14     107043718      2227930894      60      61
chr15     101991189      2336758684      60      61
chr16     90338345      2440449737      60      61
chr17     83257441      2532293732      60      61
chr18     80373285      2616938808      60      61
chr19     58617616      2698651658      60      61
chr20     64444167      2758246245      60      61
chr21     46709983      2823764492      60      61
chr22     50818468      2871252985      60      61
chrX      156040895      2922918436      60      61
chrY      57227415      3081560021      60      61
chrM      16569      3139741236      60      61
GL000008.2      209709      3139758105      60      61
GL000009.2      201709      3139971333      60      61
GL000194.1      191469      3140176427      60      61
```

Diagram illustrating a timeline with three periods: \$1, \$2, and \$0. The first two periods are grouped by a bracket above them, and the last period is grouped by a bracket below it.

- Print column1 and column2
 - `awk '{print $1,$2}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{print $1,$2}' hg38_chr.fai | head -10
chr1 248956422
chr2 242193529
chr3 198295559
chr4 190214555
chr5 181538259
chr6 170805979
chr7 159345973
chr8 145138636
chr9 138394717
chr10 133797422
[wbguo@login1 day2]$
```

awk – print out certain column

- Syntax: `awk '{operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712  60      61
chr3    198295559     499335808  60      61
chr4    190214555     700936301  60      61
chr5    181538259     894321107  60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
chr11   135086622     1838825832 60      61
chr12   133275309     1976163908 60      61
chr13   114364328     2111660483 60      61
chr14   107043718     2227930894 60      61
chr15   101991189     2336758684 60      61
chr16   90338345      2440449737 60      61
chr17   83257441      2532293732 60      61
chr18   80373285      2616938808 60      61
chr19   58617616      2698651658 60      61
chr20   64444167      2758246245 60      61
chr21   46709983      2823764492 60      61
chr22   50818468      2871252985 60      61
chrX    156040895     2922918436 60      61
chrY    57227415      3081560021 60      61
chrM    16569 3139741236 60      61
GL000008.2 209709 3139758105 60      61
GL000009.2 201709 3139971333 60      61
GL000194.1 191469 3140176427 60      61
```

- Print “contig” and column1
 - `awk '{print "contig",$1}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{print "contig",$1}' hg38_chr.fai | head -10
contig chr1
contig chr2
contig chr3
contig chr4
contig chr5
contig chr6
contig chr7
contig chr8
contig chr9
contig chr10
```

awk – print out certain column

- Syntax: `awk '{operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712 60      61
chr3    198295559     499335808 60      61
chr4    190214555     700936301 60      61
chr5    181538259     894321107 60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
chr11   135086622     1838825832 60      61
chr12   133275309     1976163908 60      61
chr13   114364328     2111660483 60      61
chr14   107043718     2227930894 60      61
chr15   101991189     2336758684 60      61
chr16   90338345      2440449737 60      61
chr17   83257441      2532293732 60      61
chr18   80373285      2616938808 60      61
chr19   58617616      2698651658 60      61
chr20   64444167      2758246245 60      61
chr21   46709983      2823764492 60      61
chr22   50818468      2871252985 60      61
chrX    156040895     2922918436 60      61
chrY    57227415      3081560021 60      61
chrM    16569 3139741236 60      61
GL000008.2 209709 3139758105 60      61
GL000009.2 201709 3139971333 60      61
GL000194.1 191469 3140176427 60      61
```

- Print “contig”, column1, column2, and addition of column4 and column5
 - `awk '{print "contig",$1,$2,$4+$5}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{print "contig",$1,$2,$4+$5}' hg38_chr.fai | head -10
contig chr1 248956422 121
contig chr2 242193529 121
contig chr3 198295559 121
contig chr4 190214555 121
contig chr5 181538259 121
contig chr6 170805979 121
contig chr7 159345973 121
contig chr8 145138636 121
contig chr9 138394717 121
contig chr10 133797422 121
```


awk – built-in variables

- **NR**: Number of Records seen so far
- **NF**: Number of Fields of current record
- **FS**: input Field Separator
- **OFS**: Output Field Separator

awk – search pattern conditions

- Syntax: `awk 'condition {operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

- print the number of line where column1 starts with chr
 - `awk '/chr/ {print NR}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '/chr/ {print NR}' hg38_chr.fai | head -10
1
2
3
4
5
6
7
8
9
10
```

awk – search pattern conditions

- Syntax: `awk -<flags> 'condition {operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712 60      61
chr3    198295559     499335808 60      61
chr4    190214555     700936301 60      61
chr5    181538259     894321107 60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
chr11   135086622     1838825832 60      61
chr12   133275309     1976163908 60      61
chr13   114364328     2111660483 60      61
chr14   107043718     2227930894 60      61
chr15   101991189     2336758684 60      61
chr16   90338345      2440449737 60      61
chr17   83257441      2532293732 60      61
chr18   80373285      2616938808 60      61
chr19   58617616      2698651658 60      61
chr20   64444167      2758246245 60      61
chr21   46709983      2823764492 60      61
chr22   50818468      2871252985 60      61
chrX    156040895     2922918436 60      61
chrY    57227415      3081560021 60      61
chrM    16569         3139741236 60      61
GL000008.2 209709 3139758105 60      61
GL000009.2 201709 3139971333 60      61
GL000194.1 191469 3140176427 60      61
```

- print column1 and column2 if column1 starts with chr, columns are separated by comma
 - `awk -v FS='\t' -v OFS=',' ' /chr/ {print $1,$2}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk -v FS='\t' -v OFS=',' ' /chr/ {print $1,$2}' hg38_chr.fai
chr1,248956422
chr2,242193529
chr3,198295559
chr4,190214555
chr5,181538259
chr6,170805979
chr7,159345973
chr8,145138636
chr9,138394717
chr10,133797422
chr11,135086622
chr12,133275309
chr13,114364328
chr14,107043718
chr15,101991189
chr16,90338345
chr17,83257441
chr18,80373285
chr19,58617616
chr20,64444167
chr21,46709983
chr22,50818468
chrX,156040895
chrY,57227415
chrM,16569
```

awk – search pattern conditions

- Syntax: `awk 'condition {operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

- print the line where column2 is less than 100,000
- `awk '$2 < 100000 {print $0}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '$2 < 100000 {print $0}' hg38_chr.fai | more
chrM 16569 3139741236 60 61
GL000208.1 92689 3140745787 60 61
GL000226.1 15008 3142392679 60 61
KI270302.1 2274 3142407961 60 61
KI270303.1 1942 3142410296 60 61
KI270304.1 2165 3142412294 60 61
KI270305.1 1472 3142414519 60 61
KI270310.1 1201 3142416039 60 61
KI270311.1 12399 3142417284 60 61
KI270312.1 998 3142429913 60 61
KI270315.1 2276 3142430951 60 61
KI270316.1 1444 3142433288 60 61
KI270317.1 37690 3142434780 60 61
KI270320.1 4416 3142473122 60 61
KI270322.1 21476 3142477635 60 61
KI270329.1 1040 3142499492 60 61
```

Comparison symbol:

- Equal: `==`
- Greater than: `>`
- Less than: `<`
- Greater equal: `>=`
- Less equal: `<=`

awk – search pattern **combined** conditions

- Syntax: `awk 'condition {operation}' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

- print the line where column2 is **less than** 100,000, but **greater than** 10,000
 - `awk '$2 < 100000 && $2 > 10000 {print $0}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '$2 < 100000 && $2 > 10000 {print $0}' hg38_chr.fai | more
chrM 16569 3139741236 60 61
GL000208.1 92689 3140745787 60 61
GL000226.1 15008 3142392679 60 61
KI270311.1 12399 3142417284 60 61
KI270317.1 37690 3142434780 60 61
KI270322.1 21476 3142477635 60 61
KI270435.1 92983 3142593847 60 61
KI270512.1 22689 3143241452 60 61
KI270538.1 91309 3143439231 60 61
KI270579.1 31033 3143536013 60 61
KI270589.1 44474 3143598397 60 61
KI270707.1 32032 3143835450 60 61
KI270709.1 66860 3143997873 60 61
KI270710.1 40176 3144065871 60 61
```

Combine the conditions:

- And: **&&**
- Or: **||**

awk – if condition

- Syntax: `awk '{if(condition) action}' <filename>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

- Print column1 and If column 2 is greater than 100,000, print 100,000 as column2
 - `awk '{if ($2 > 100000) print $1, 100000}' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{if ($2 > 100000) print $1, 100000}' hg38_chr.fai | more
chr1 100000
chr2 100000
chr3 100000
chr4 100000
chr5 100000
chr6 100000
chr7 100000
chr8 100000
chr9 100000
chr10 100000
chr11 100000
chr12 100000
chr13 100000
chr14 100000
chr15 100000
chr16 100000
chr17 100000
chr18 100000
chr19 100000
chr20 100000
chr21 100000
chr22 100000
chrX 100000
chrY 100000
GL000008.2 100000
GL000009.2 100000
GL000194.1 100000
```

awk – if-else condition

- Syntax: `awk '{if(condition) action; else (condition) action}' <filename>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712 60      61
chr3    198295559     499335808 60      61
chr4    190214555     700936301 60      61
chr5    181538259     894321107 60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
chr11   135086622     1838825832 60      61
chr12   133275309     1976163908 60      61
chr13   114364328     2111660483 60      61
chr14   107043718     2227930894 60      61
chr15   101991189     2336758684 60      61
chr16   90338345      2440449737 60      61
chr17   83257441      2532293732 60      61
chr18   80373285      2616938808 60      61
chr19   58617616      2698651658 60      61
chr20   64444167      2758246245 60      61
chr21   46709983      2823764492 60      61
chr22   50818468      2871252985 60      61
chrX    156040895     2922918436 60      61
chrY    57227415      3081560021 60      61
chrM    16569        3139741236 60      61
GL000008.2 209709    3139758105 60      61
GL000009.2 201709    3139971333 60      61
GL000194.1 191469    3140176427 60      61
```

- Print column1 and If column 2 is greater than 100,000, print “long”, else print “short” as column2
 - `awk '{if ($2 > 100000) print $1,"long"; else print $1,"short" }' hg38_chr.fai`

```
[wbguo@login1 day2]$ awk '{if ($2 > 100000) print $1,"long"; else print $1,"short" }' hg38_chr.fai
chr1 long
chr2 long
chr3 long
chr4 long
chr5 long
chr6 long
chr7 long
chr8 long
chr9 long
chr10 long
chr11 long
chr12 long
```


awk – calculation

- How to calculate the sum of column2 using awk?
 - Define a new variable, and add \$2 to it for each line

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

Command:

➤ `awk '{sum+=$2} END {print sum}' hg38_chr.fai`

Variable

`sum=sum+$2`

When you reach to the
end of file, then print

```
[wbguo@login1 day2]$ awk '{sum+=$2} END {print sum}' hg38_chr.fai
3099750718
[wbguo@login1 day2]$
```


Practice:



- Use `sample-A01.gz`
 - Print column1 and column2, they are separated by hyphen (-)
- Filter sites if column4 is not CG
- calculate the sum of last column

grep

- **grep**: **g**lobal **r**egular **e**xpression **p**rint (search for pattern in the file or standard output)
- Syntax: `grep -<flag> 'Pattern' <file name>`

- Flag options

- **-v**: print lines that **do not contain** the string (**invert-match**)
- **-n**: line **numbers** where the string occurs
- **-c**: **count** number of lines containing the string
- **-l**: list files that contain the string
- **-i**: case-**i**nsensitive
- **-E**: **E**xtended Regular Expression
- **-B <num>**: **b**efore context, show <num> lines before the match
- **-A <num>**: **a**fter context, show <num> lines after the match
- **--color**: color matched string

grep – substring

- Syntax: `grep -<flag> 'Pattern' <file name>`

- Find lines with “chr”

➤ `grep 'chr' hg38_chr.fai`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
GL000008.2 209709 3139758105 60 61
GL000009.2 201709 3139971333 60 61
GL000194.1 191469 3140176427 60 61
```

```
[wbguo@login1 day2]$ grep 'chr' hg38_chr.fai
chr1 248956422 8 60 61
chr2 242193529 253105712 60 61
chr3 198295559 499335808 60 61
chr4 190214555 700936301 60 61
chr5 181538259 894321107 60 61
chr6 170805979 1078885012 60 61
chr7 159345973 1252537766 60 61
chr8 145138636 1414539514 60 61
chr9 138394717 1562097136 60 61
chr10 133797422 1702798442 60 61
chr11 135086622 1838825832 60 61
chr12 133275309 1976163908 60 61
chr13 114364328 2111660483 60 61
chr14 107043718 2227930894 60 61
chr15 101991189 2336758684 60 61
chr16 90338345 2440449737 60 61
chr17 83257441 2532293732 60 61
chr18 80373285 2616938808 60 61
chr19 58617616 2698651658 60 61
chr20 64444167 2758246245 60 61
chr21 46709983 2823764492 60 61
chr22 50818468 2871252985 60 61
chrX 156040895 2922918436 60 61
chrY 57227415 3081560021 60 61
chrM 16569 3139741236 60 61
```

grep – substring

- Syntax: `grep -<flag> 'Pattern' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8        60        61
chr2    242193529      253105712 60        61
chr3    198295559      499335808 60        61
chr4    190214555      700936301 60        61
chr5    181538259      894321107 60        61
chr6    170805979      1078885012 60        61
chr7    159345973      1252537766 60        61
chr8    145138636      1414539514 60        61
chr9    138394717      1562097136 60        61
chr10   133797422      1702798442 60        61
chr11   135086622      1838825832 60        61
chr12   133275309      1976163908 60        61
chr13   114364328      2111660483 60        61
chr14   107043718      2227930894 60        61
chr15   101991189      2336758684 60        61
chr16   90338345       2440449737 60        61
chr17   83257441       2532293732 60        61
chr18   80373285       2616938808 60        61
chr19   58617616       2698651658 60        61
chr20   64444167       2758246245 60        61
chr21   46709983       2823764492 60        61
chr22   50818468       2871252985 60        61
chrX    156040895      2922918436 60        61
chrY    57227415       3081560021 60        61
chrM    16569          3139741236 60        61
GL000008.2 209709 3139758105 60        61
GL000009.2 201709 3139971333 60        61
GL000194.1 191469 3140176427 60        61
```

- Find lines without “chr”

➤ `grep -v 'chr' hg38_chr.fai`

```
[wbguo@login1 day2]$ grep -v 'chr' hg38_chr.fai | more
GL000008.2 209709 3139758105 60        61
GL000009.2 201709 3139971333 60        61
GL000194.1 191469 3140176427 60        61
GL000195.1 182896 3140371111 60        61
GL000205.2 185591 3140557079 60        61
GL000208.1 92689 3140745787 60        61
GL000213.1 164239 3140840044 60        61
GL000214.1 137718 3141007044 60        61
GL000216.2 176608 3141147081 60        61
GL000218.1 161147 3141326656 60        61
GL000219.1 179198 3141490512 60        61
GL000220.1 161802 3141672720 60        61
GL000221.1 155397 3141837242 60        61
GL000224.1 179693 3141995252 60        61
GL000225.1 211173 3142177963 60        61
GL000226.1 15008 3142392679 60        61
KI270302.1 2274 3142407961 60        61
KI270303.1 1942 3142410296 60        61
```

grep – substring

- Syntax: `grep -<flag> 'Pattern' <file name>`

```
[wbguo@login1 day2]$ more hg38_chr.fai
chr1    248956422      8      60      61
chr2    242193529     253105712 60      61
chr3    198295559     499335808 60      61
chr4    190214555     700936301 60      61
chr5    181538259     894321107 60      61
chr6    170805979     1078885012 60      61
chr7    159345973     1252537766 60      61
chr8    145138636     1414539514 60      61
chr9    138394717     1562097136 60      61
chr10   133797422     1702798442 60      61
chr11   135086622     1838825832 60      61
chr12   133275309     1976163908 60      61
chr13   114364328     2111660483 60      61
chr14   107043718     2227930894 60      61
chr15   101991189     2336758684 60      61
chr16   90338345      2440449737 60      61
chr17   83257441      2532293732 60      61
chr18   80373285      2616938808 60      61
chr19   58617616      2698651658 60      61
chr20   64444167      2758246245 60      61
chr21   46709983      2823764492 60      61
chr22   50818468      2871252985 60      61
chrX    156040895     2922918436 60      61
chrY    57227415      3081560021 60      61
chrM    16569 3139741236 60      61
GL000008.2 209709 3139758105 60      61
GL000009.2 201709 3139971333 60      61
GL000194.1 191469 3140176427 60      61
```

- Count the number of lines that has “chr”
 - `grep -c 'chr' hg38_chr.fai`
 - `grep 'chr' hg38_chr.fai | wc -l`

```
[wbguo@login1 day2]$ grep -c 'chr' hg38_chr.fai
25
[wbguo@login1 day2]$ grep 'chr' hg38_chr.fai | wc -l
25
```

grep – substring in multiple files

- Find lines that contain 'chr' in multiple files

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt test.txt
[wbguo@login1 day2]$ grep -n 'chr' chr1_10 mod_chr1_10
chr1_10:1:>chr1 1
mod_chr1_10:1:>chr1 chr1
[wbguo@login1 day2]$
```

- Find files that have 'chr' in their contents

```
[wbguo@login1 day2]$ ls
chr1_10 chr2_10 hg38_chr.fai mod_chr1_10 sample-A01.gz seq_file_list.txt test.txt
[wbguo@login1 day2]$ grep -l 'chr' *
chr1_10
chr2_10
hg38_chr.fai
mod_chr1_10
```

Practice:



- Count the number of lines whose contig contain 'GL' in hg38_chr.fai file
- Count the number of lines whose contig doesn't contain 'KL' in hg38_chr.fai file

Practice:



- Count the number of lines whose contig contain 'GL' in hg38_chr.fai file

```
[wbguo@login1 day2]$ grep -c 'GL' hg38_chr.fai  
16
```

- Count the number of lines whose contig doesn't contain 'KL' in hg38_chr.fai file

```
[wbguo@login1 day2]$ grep -vc 'KL' hg38_chr.fai  
194
```


Regular expression

- A regular expression (regex) is a **sequence of characters** that **specifies a search pattern**

Example of Patterns

- UID: 123456789, 987654321,
 - 9 digits
- Phone number: (123)-456-7890, (987)-654-3210, ...
 - 3 digits in parenthesis, followed by hyphen, 3 digits, hyphen, 4 digits
- Address: 611 Charles E Young Dr E, 529 Boyer Hall, Los Angeles, CA, 90095
 - 5 segments separated by comma,
 - 1st and 2nd are a mixture of digits and characters
 - 3rd is a string
 - 4th is a string with 2 capitalized characters
 - 5th is a 5-digit number

Regular expression:

describe the **pattern** in a formula
(a sequence of characters)

Regular expression can match multiple places

regular expression →

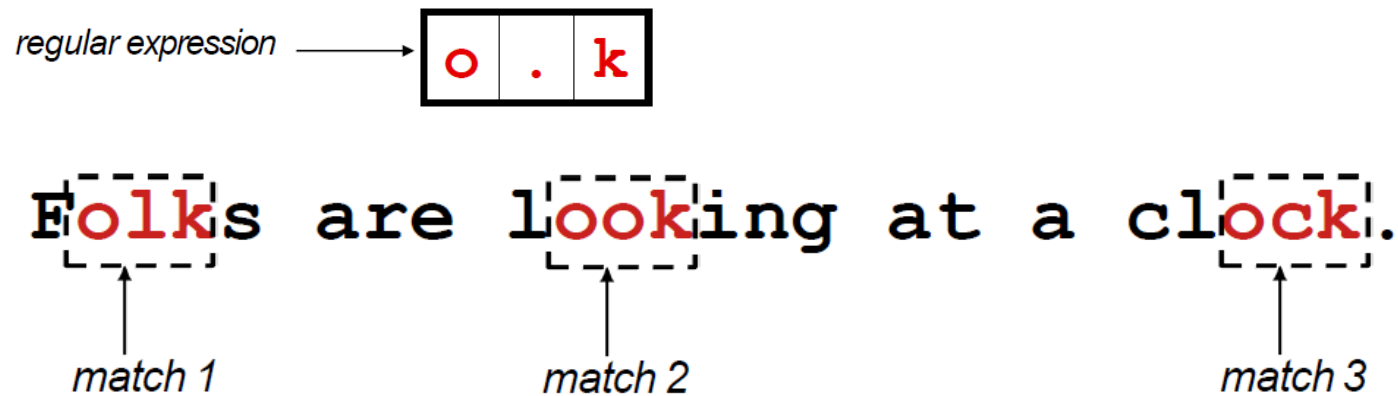
c	k	s
----------	----------	----------

UNIX Tools su**cks**.
 ↑
 match

UNIX Tools ro**cks**.
 ↑
 match

Rules for writing regular expression

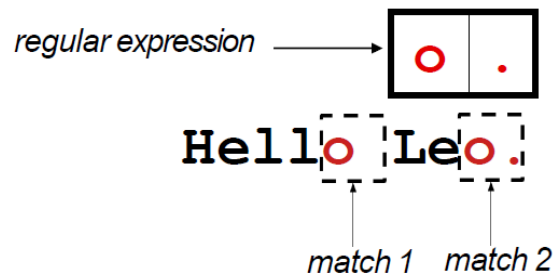
- “.” can be used to match any character



```
[wbguo@login1 day2]$ echo "Folks are looking at a clock" | grep --color 'o.k'
Folks are looking at a clock
```

Rules for writing regular expression

- “.” can be used to **match any character**
 - If you want to match a “.” itself, use “\” (escape) which removes the specific meaning of the character



```
[wbguo@login1 day2]$ echo "Hello Leo." | grep --color 'o.'
```

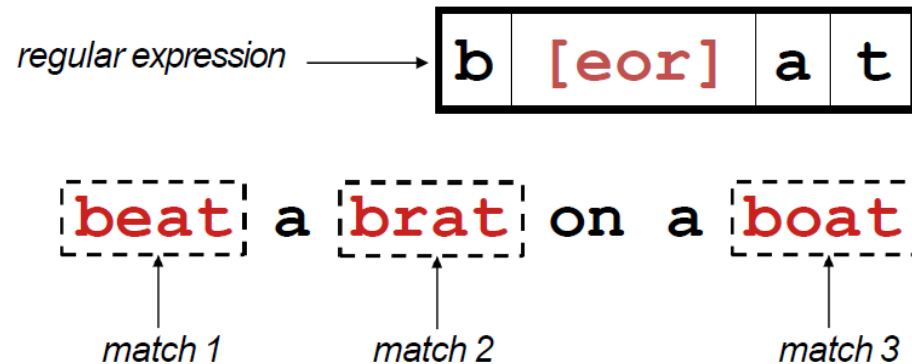
Hello Leo.

```
[wbguo@login1 day2]$ echo "Hello Leo." | grep --color 'o\.'
```

Hello Leo.

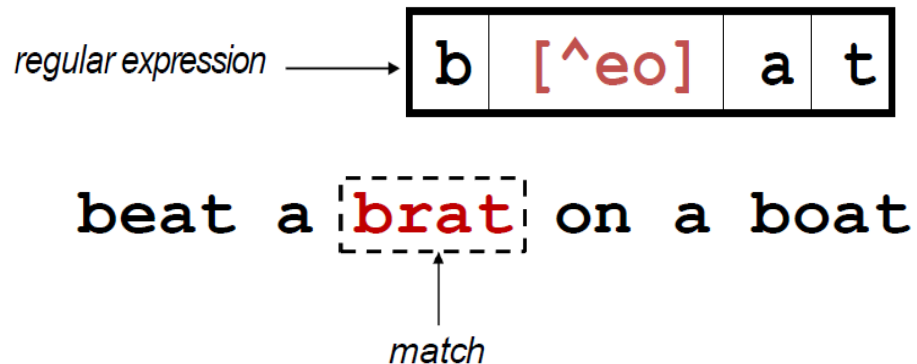
Rules for writing regular expression

- “.” can be used to match any character
 - If you want to match a “.” itself, use “\” (escape) which can remove the specific meaning of the following character
- **[]** can be used to match **any**/none of the specified characters



Rules for writing regular expression

- “.” can be used to match any character
 - If you want to match a “.” itself, use “\” (escape) which can remove the specific meaning of the following character
- **[]** can be used to match any/**none** of the specified characters



Rules for writing regular expression

- “.” can be used to match any character
 - If you want to match a “.” itself, use “\” (escape) which can remove the specific meaning of the following character
- **[]** can be used to match any/none of the specified characters
 - Digits: [0-9] or [[:digit:]]
 - Alphabetic characters: [a-zA-Z] or [[:alpha:]]
 - Alphanumeric characters: [a-zA-Z0-9] or [[:alnum:]]

Practice:



- If I want to match both Fastq and fastq, what's the corresponding regular expression?
- Try the following commands on the terminal
 - `echo "beat a brat boat" | grep --color 'b[eor]at'`
 - `echo "beat a brat boat" | grep --color 'b[^eo]at'`

Practice:



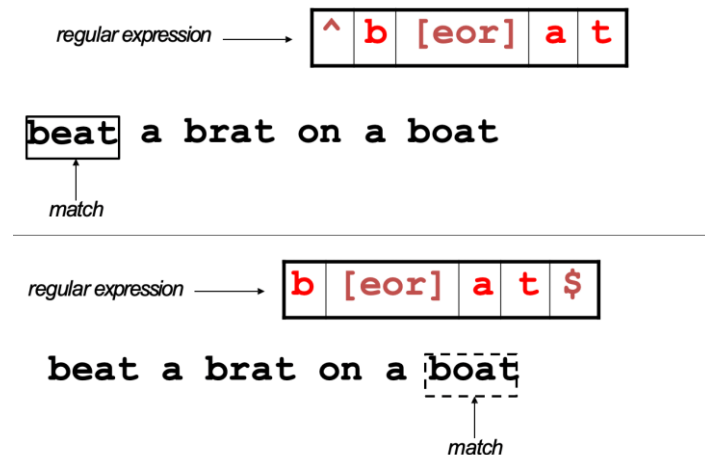
- If I want to match both Fastq and fastq, what's the corresponding regular expression?

`[Ff]astq`

- Try the following commands on the terminal
 - `echo "beat a brat boat" | grep --color 'b[eor]at'`
 - `echo "beat a brat boat" | grep --color 'b[^eo]at'`

Rules for writing regular expression

- “.” can be used to match any character
 - If you want to match a “.” itself, use “\” (escape) which can remove the specific meaning of the following character
- “[]” can be used to match any/none of the specified characters
 - Digits: [0-9] or [[:digit:]]
 - Alphabetic characters: [a-zA-Z] or [[:alpha:]]
 - Alphanumeric characters: [a-zA-Z0-9] or [[:alnum:]]
- “^” means the beginning of the string, “\$” means the end of the string



Rules for writing regular expression

Repetition operators and alternative operators

- *** : match preceeding pattern 0 or more times (e.g. **el*o** will match **Hello** and **video**)
- +** : match preceeding pattern 1 or more time (e.g. **el+o** will match **Hello** but not **video**)
- ?** : match preceeding pattern 0 or 1 time (e.g. **el?o** will match **video** and **develop** but not **Hello**)
- {n}** : match preceeding pattern exactly n times
- {n,}** : match preceeding pattern exactly at least n times
- {,m}** : match preceeding pattern exactly at most m times
- {n,m}** : match preceeding pattern at least n times and at most m times
- r1|s2** : alternative, either r1 or s2
- E {

Rules for writing regular expression

Groups

- Use **()** to group the characters

- Examples

a* matches 0 or more occurrences of a

abc* matches ab followed by 0 or more occurrences of c: **ab**, **abc**, **abcc**, ...

(abc)* matches 0 or more occurrences of abc: **abc**, **abcabc**, **abcabcabc**, ...

Practice:



- Try to write the regular expression:
 - it starts with 2 capitalized character, and ends with .2, the middle 6 are digits
- Try to apply the regular expression to hg38_chr.fai file

Practice:



- Try to write the regular expression:
 - it starts with 2 capitalized characters, and ends with .2, the middle 6 are digits

`[A-Z]{2}[0-9]{6}\.2`

- Try to apply the regular expression to hg38_chr.fai file
 - `cat hg38_chr.fai | grep -E --color '[A-Z]{2}[0-9]{6}\.2'`

```
[wbguo@login1 day2]$ cat hg38_chr.fai | grep -E --color '[A-Z]{2}[0-9]{6}\.2'
```

GL000008.2	209709	3139758105	60	61
GL000009.2	201709	3139971333	60	61
GL000205.2	185591	3140557079	60	61
GL000216.2	176608	3141147081	60	61

Summary

Command	Function
cat	Show file contents/merge files
more/less	Show file contents
head/tail	Show file contents at beginning/end
zcat/zmore	Show contents of compressed file
sort	Sort a file based on column
>/>>	Redirect output to files (write/append)
	Combine commands
wc	count
cut	Extract a part of string
uniq	Show the unique line
diff	Show the difference of 2 files
sed	Non-interactive editing
awk	Text processing
grep	search string/pattern in the file

Q&A

[Google doc](#)

We did it!

