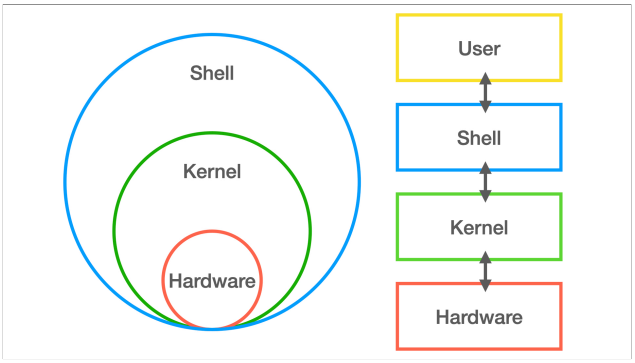
Linux basics. Outline

1. Linux and shells
2. reasons why linux is better
3. linux environments
4. commonly-used commands
5. useful tips in Linux
6. command **`man`** (short for MANual page)
7. understand standard input (STDIN, 0), output (STDOUT, 1), and error (STDERR, 2) using **`ls`**
8. file and folder permissions
9. vi editor
10. **`sed`** (short for Stream EDitor) to find and replace text
11. print and count file content
12. use **`grep`** and **`wc`** to count sequences
13. Shell scripting
14. Q&A

- Question 1. How many sequences in a fasta/fastq file?
- Question 2. How to generate a metadata file (.csv) from a sequence file (.fasta)?

1. Linux and shells

- Linux is a family of **"open-source" and "Unix-like"** operating systems
    - **kernel**: a computer program, or the core of a computer's OS, controling everything in the system.
    - **shell**: a special user program, providing an interface for user to use OS services.



2. reasons why linux is better

- **free to use**
    - better community and support
- **open source culture** (started from Freax v0.01 in 1991)
    - various distributions, ownership, and customization
    - revive older computers
    - reliability (do not worry about re-install or re-boot)
- **security and privacy**
- **perfect for programmers**
    - shell script
    - software updates more quickly

3. linux environments

1. Mac user: Terminal app
2. PC user: Windows Subsystem for Linux (WSL) (**https://learn.microsoft.com/en-us/windows/wsl/install**)

4. commonly-used commands

Mac/Linux	Windows	Note
cd	cd	move to
pwd	cd	print working directory
ls	dir	list files
mkdir	mkdir	create directory
touch	-	create file
cp	copy	copy file
mv	move	move file
rm	del	delete file
clear	cls	clear history

- Don't use these commands in cmd.exe (命令提示字元) of PC

5. useful tips in Linux

1. All commands in Linux are **case-sensitive**.
2. Type commands or scripts in **the system prompt (means ready to command)** and processing actions only when you hit **enter**
3. Use **up arrow** to show the last executed command, and you can search and modify them
4. Use `history` to see all executed commands
5. Use **tab** to auto-complete filenames
6. Use **wildcards** to match filenames
    - * - character(s) in any length (including none)
    - ? - one single character
7. Type Ctrl+d to logout and Ctrl+c to kill running job
8. Type tab in linux: Ctrl+v+tab

6. command **`man`** (short for MANual page)

- `man` command in Linux is used to display the user manual of any command.
- for example, `man ls`

useful tips

- Hit **return/enter** to advance one line at a time
- Hit **space** to advance one page at a time
- Hit **Ctrl-f/Ctrl-b** to forward/backward one page at a time
- Hit **q** to quit

7. Standard input (STDIN, 0), output (STDOUT, 1), and error (STDERR, 2) using `ls`

Difference between ">" and ">>"

In [ ]:
```
## ```ls``` is short for "list".
   ## use it to list files and folders in current (or target) directory

## use > to output into a file
! ls > LIST

## use ```cat``` (short for concatenate) file to print the file
! cat LIST

## use >> to append to a file
! ls >> LIST
! cat LIST
```

use 2> output error message to a file

In [13]:
```
# output the standard ouput into STDOUT
! ls none > STDOUT
```

```
ls: none: No such file or directory
```

In [15]:
```
# STDOUT is empty
! cat STDOUT
```

In [16]:
```
# output the standard error into STDERR
! ls none > STDOUT 2> STDERR
```

In [17]:
```
! cat STDERR
```

```
ls: none: No such file or directory
```

In [2]:
```
# assign 2 into 1, and then output into allinone file
! ls none > STDALL 2>&1
```

In [3]:
```
! cat STDALL
```

```
ls: none: No such file or directory
```

more about `ls` options

In [4]:
```
# -l
#   List in long format.
! ls -l
```

```
total 11024
drwxr-xr-x  15 yngong  staff      480 Oct 12 22:20 img
-rw-r--r--   1 yngong  staff     9024 Apr 14 07:27 linux_basics.aux
-rw-r--r--   1 yngong  staff    33141 Oct 12 22:38 linux_basics.ipynb
-rw-r--r--   1 yngong  staff    39459 Apr 14 07:27 linux_basics.log
-rw-r--r--   1 yngong  staff     1901 Apr 14 07:27 linux_basics.out
-rw-r--r--@  1 yngong  staff   803225 Apr 14 07:27 linux_basics.pdf
-rw-r--r--@  1 yngong  staff   625501 Apr 20 00:00 linux_basics.slides.html
-rw-r--r--   1 yngong  staff    51333 Apr 14 07:27 linux_basics.tex
-rw-r--r--   1 yngong  staff    51342 Apr 14 07:27 linux_basics.texe
-rw-r--r--   1 yngong  staff     7567 Apr 14 09:29 linux_shell.aux
-rw-r--r--   1 yngong  staff    25757 Apr 14 10:03 linux_shell.ipynb
-rw-r--r--   1 yngong  staff    40113 Apr 14 09:29 linux_shell.log
-rw-r--r--   1 yngong  staff     1571 Apr 14 09:29 linux_shell.out
-rw-r--r--@  1 yngong  staff  2862419 Apr 14 09:29 linux_shell.pdf
-rw-r--r--@  1 yngong  staff   619540 Apr 20 00:28 linux_shell.slides.html
-rw-r--r--   1 yngong  staff    51269 Apr 14 09:29 linux_shell.tex
-rw-r--r--   1 yngong  staff    51278 Apr 14 09:29 linux_shell.texe
drwxr-xr-x   5 yngong  staff      160 Oct 12 22:40 workspace
```

In [5]:
```
# -a
#    do not ignore hidden entries starting with .
! ls -la
```

```
total 11048
drwxr-xr-x  23 yngong  staff      736 Oct 12 22:39 .
drwxr-xr-x  12 yngong  staff      384 Oct 12 16:40 ..
-rw-r--r--@  1 yngong  staff     6148 Oct 12 22:40 .DS_Store
drwxr-xr-x   4 yngong  staff      128 Mar 16  2021 .ipynb_checkpoints
-rw-r--r--   1 yngong  staff     1562 Apr 13  2021 .log
drwxr-xr-x  15 yngong  staff      480 Oct 12 22:20 img
-rw-r--r--   1 yngong  staff     9024 Apr 14 07:27 linux_basics.aux
-rw-r--r--   1 yngong  staff    33141 Oct 12 22:38 linux_basics.ipynb
-rw-r--r--   1 yngong  staff    39459 Apr 14 07:27 linux_basics.log
-rw-r--r--   1 yngong  staff     1901 Apr 14 07:27 linux_basics.out
-rw-r--r--@  1 yngong  staff   803225 Apr 14 07:27 linux_basics.pdf
-rw-r--r--@  1 yngong  staff   625501 Apr 20 00:00 linux_basics.slides.html
-rw-r--r--   1 yngong  staff    51333 Apr 14 07:27 linux_basics.tex
-rw-r--r--   1 yngong  staff    51342 Apr 14 07:27 linux_basics.texe
-rw-r--r--   1 yngong  staff     7567 Apr 14 09:29 linux_shell.aux
-rw-r--r--   1 yngong  staff    25757 Apr 14 10:03 linux_shell.ipynb
-rw-r--r--   1 yngong  staff    40113 Apr 14 09:29 linux_shell.log
-rw-r--r--   1 yngong  staff     1571 Apr 14 09:29 linux_shell.out
-rw-r--r--@  1 yngong  staff  2862419 Apr 14 09:29 linux_shell.pdf
-rw-r--r--@  1 yngong  staff   619540 Apr 20 00:28 linux_shell.slides.html
-rw-r--r--   1 yngong  staff    51269 Apr 14 09:29 linux_shell.tex
-rw-r--r--   1 yngong  staff    51278 Apr 14 09:29 linux_shell.texe
drwxr-xr-x   5 yngong  staff      160 Oct 12 22:40 workspace
```

In [6]:
```
# -t
#    Sort by time modified (most recently modified first)
! ls -lat
```

```
total 11048
drwxr-xr-x  23 yngong  staff      736 Oct 12 22:40 .
-rw-r--r--   1 yngong  staff    34228 Oct 12 22:40 linux_basics.ipynb
drwxr-xr-x   5 yngong  staff      160 Oct 12 22:40 workspace
-rw-r--r--@  1 yngong  staff     6148 Oct 12 22:40 .DS_Store
drwxr-xr-x  15 yngong  staff      480 Oct 12 22:20 img
drwxr-xr-x  12 yngong  staff      384 Oct 12 16:40 ..
-rw-r--r--@  1 yngong  staff   619540 Apr 20 00:28 linux_shell.slides.html
-rw-r--r--@  1 yngong  staff   625501 Apr 20 00:00 linux_basics.slides.html
-rw-r--r--   1 yngong  staff    25757 Apr 14 10:03 linux_shell.ipynb
-rw-r--r--   1 yngong  staff    40113 Apr 14 09:29 linux_shell.log
-rw-r--r--@  1 yngong  staff  2862419 Apr 14 09:29 linux_shell.pdf
-rw-r--r--   1 yngong  staff     1571 Apr 14 09:29 linux_shell.out
-rw-r--r--   1 yngong  staff     7567 Apr 14 09:29 linux_shell.aux
-rw-r--r--   1 yngong  staff    51269 Apr 14 09:29 linux_shell.tex
-rw-r--r--   1 yngong  staff    51278 Apr 14 09:29 linux_shell.texe
-rw-r--r--   1 yngong  staff    39459 Apr 14 07:27 linux_basics.log
-rw-r--r--@  1 yngong  staff   803225 Apr 14 07:27 linux_basics.pdf
-rw-r--r--   1 yngong  staff     1901 Apr 14 07:27 linux_basics.out
-rw-r--r--   1 yngong  staff     9024 Apr 14 07:27 linux_basics.aux
-rw-r--r--   1 yngong  staff    51333 Apr 14 07:27 linux_basics.tex
-rw-r--r--   1 yngong  staff    51342 Apr 14 07:27 linux_basics.texe
-rw-r--r--   1 yngong  staff     1562 Apr 13  2021 .log
drwxr-xr-x   4 yngong  staff      128 Mar 16  2021 .ipynb_checkpoints
```

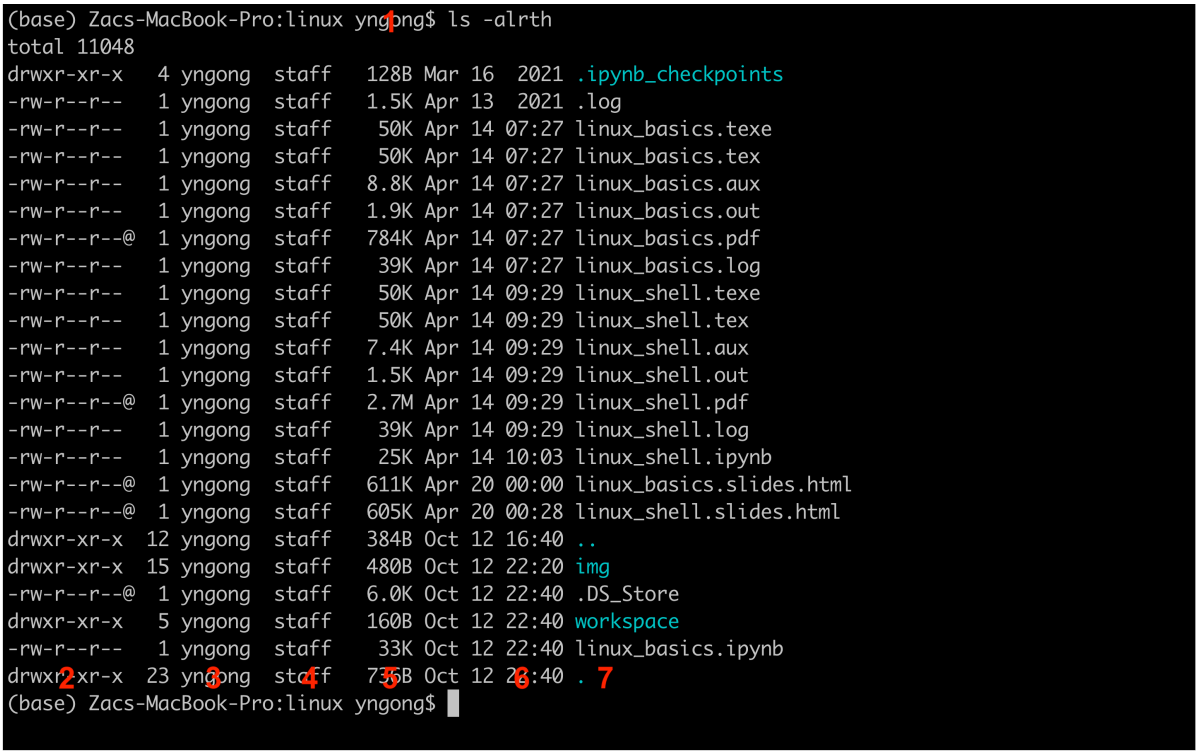
In [7]:
```
# -r
#    Reverse the order of the sort to get reverse lexicographical order
! ls -latr
```

```
total 11048
drwxr-xr-x   4 yngong  staff      128 Mar 16  2021 .ipynb_checkpoints
-rw-r--r--   1 yngong  staff     1562 Apr 13  2021 .log
-rw-r--r--   1 yngong  staff    51342 Apr 14 07:27 linux_basics.texe
-rw-r--r--   1 yngong  staff    51333 Apr 14 07:27 linux_basics.tex
-rw-r--r--   1 yngong  staff     9024 Apr 14 07:27 linux_basics.aux
-rw-r--r--   1 yngong  staff     1901 Apr 14 07:27 linux_basics.out
-rw-r--r--@  1 yngong  staff   803225 Apr 14 07:27 linux_basics.pdf
-rw-r--r--   1 yngong  staff    39459 Apr 14 07:27 linux_basics.log
-rw-r--r--   1 yngong  staff    51278 Apr 14 09:29 linux_shell.texe
-rw-r--r--   1 yngong  staff    51269 Apr 14 09:29 linux_shell.tex
-rw-r--r--   1 yngong  staff     7567 Apr 14 09:29 linux_shell.aux
-rw-r--r--   1 yngong  staff     1571 Apr 14 09:29 linux_shell.out
-rw-r--r--@  1 yngong  staff  2862419 Apr 14 09:29 linux_shell.pdf
-rw-r--r--   1 yngong  staff    40113 Apr 14 09:29 linux_shell.log
-rw-r--r--   1 yngong  staff    25757 Apr 14 10:03 linux_shell.ipynb
-rw-r--r--@  1 yngong  staff   625501 Apr 20 00:00 linux_basics.slides.html
-rw-r--r--@  1 yngong  staff   619540 Apr 20 00:28 linux_shell.slides.html
drwxr-xr-x  12 yngong  staff      384 Oct 12 16:40 ..
drwxr-xr-x  15 yngong  staff      480 Oct 12 22:20 img
-rw-r--r--@  1 yngong  staff     6148 Oct 12 22:40 .DS_Store
drwxr-xr-x   5 yngong  staff      160 Oct 12 22:40 workspace
-rw-r--r--   1 yngong  staff    34228 Oct 12 22:40 linux_basics.ipynb
drwxr-xr-x  23 yngong  staff      736 Oct 12 22:40 .
```

In [8]:
```
# -h
#    When used with the -l option, use unit suffixes: Byte, Kilobyte,
#    Megabyte, Gigabyte, Terabyte and Petabyte
# for example, this option shows linux_shell.ipynb in 25K, not 25757
! ls -latrh
```

```
total 11048
drwxr-xr-x   4 yngong  staff   128B Mar 16  2021 .ipynb_checkpoints
-rw-r--r--   1 yngong  staff   1.5K Apr 13  2021 .log
-rw-r--r--   1 yngong  staff    50K Apr 14 07:27 linux_basics.texe
-rw-r--r--   1 yngong  staff    50K Apr 14 07:27 linux_basics.tex
-rw-r--r--   1 yngong  staff   8.8K Apr 14 07:27 linux_basics.aux
-rw-r--r--   1 yngong  staff   1.9K Apr 14 07:27 linux_basics.out
-rw-r--r--@  1 yngong  staff   784K Apr 14 07:27 linux_basics.pdf
-rw-r--r--   1 yngong  staff    39K Apr 14 07:27 linux_basics.log
-rw-r--r--   1 yngong  staff    50K Apr 14 09:29 linux_shell.texe
-rw-r--r--   1 yngong  staff    50K Apr 14 09:29 linux_shell.tex
-rw-r--r--   1 yngong  staff   7.4K Apr 14 09:29 linux_shell.aux
-rw-r--r--   1 yngong  staff   1.5K Apr 14 09:29 linux_shell.out
-rw-r--r--@  1 yngong  staff   2.7M Apr 14 09:29 linux_shell.pdf
-rw-r--r--   1 yngong  staff    39K Apr 14 09:29 linux_shell.log
-rw-r--r--   1 yngong  staff    25K Apr 14 10:03 linux_shell.ipynb
-rw-r--r--@  1 yngong  staff   611K Apr 20 00:00 linux_basics.slides.html
-rw-r--r--@  1 yngong  staff   605K Apr 20 00:28 linux_shell.slides.html
drwxr-xr-x  12 yngong  staff   384B Oct 12 16:40 ..
drwxr-xr-x  15 yngong  staff   480B Oct 12 22:20 img
-rw-r--r--@  1 yngong  staff   6.0K Oct 12 22:40 .DS_Store
drwxr-xr-x   5 yngong  staff   160B Oct 12 22:40 workspace
-rw-r--r--   1 yngong  staff    33K Oct 12 22:40 linux_basics.ipynb
drwxr-xr-x  23 yngong  staff   736B Oct 12 22:40 .
```

8. file and folder permissions



- (1) System prompt model (2) file type and permission (3) owner (4) group who owns this file (5) file size (6) time stamp (7) file name (. means current directory and .. as parent)

ten letters and dashes indicate the permissions

for example, d rwx r-x r-x

- The first letter **d** means a directory rather than a file.
- The remaining nine letters are grouped into 3 triplets, including owner, group, and other levels
- r = read; w = write; x = execute
- the owner has the read/write/execute permission
- other users in the same group have the read/execute permission, but no write permission

convert permission code

- The 3 triplets (e.g., **d rwx r-x r-x**) can be represented by three 3-bit codes (111 101 101).
- converter: $r*2^2 + w*2^1 + x*2^0$
    - rwx as 111: $1*2^2 + 1*2^1 + 1*2^0 = 4+2+1 = 7$
    - r-x as 101: $1*2^2 + 0*2^1 + 1*2^0 = 4+0+1 = 5$
- File permission of "d rwx r-x r-x" will be 755.

change permission

- change permission using `chmod` (short for change mode)

```
In [17]: #####
         # change the permission of list file from 644 to 755
#####
## create a LIST file first
! ls > LIST
! ls -l LIST
```

```
-rw-r--r--@ 1 yngong  staff  459 Oct 12 22:53 LIST
```

```
In [18]: ! chmod 755 LIST
```

```
In [19]: ! ls -l LIST
```

```
-rwxr-xr-x@ 1 yngong  staff  459 Oct 12 22:53 LIST
```

9. vi editor

- `vi` is the most popular and basic text editor on Linux-like system.
- three editing modes in vi
    - command mode (full screen, by default)
    - insert mode (full screen)
    - command-line mode (one working line)

try the followings to enter insert mode

- i: insert before cursor
- I: insert at the beginning of line
- a: append after cursor
- A: append at end of line
- o: open new line below current line
- O: open new line above current line
- press ESC to leave the insert mode

command-line mode

- press *__shift and :__* to get a command-line prompt **:** at the bottom
- `:wq` : exit and save changes (w = write, q = quit)
- `:q!` : exit without saving changes (! = force quitting by neglecting changes)
- `:w filename` : write to a new filename
- `:w! filename` : force write to a new filename
- delete `:` or press ESC twice to get back to full-screen command mode

delete in command mode

- x: delete character under cursor
- dd: delete one line under cursor
- dw: delete till the end of this word
- db: delete characters in this word before
- **press u immediately to undo the deletion**

search and replace in command-line mode

- /string: Search for string
- enter `/` alone followed by return will repeatly search
- s/string1/string2: Replace string1 with string2
- s/string1/string2/g: Replace all string1 with string2 in the current line.
- %s/string1/string2/g: Replace string1 with string2 in the entire document.
- set nu : display line numbers
- set nonu : hide the line numbers

Go to specific line and delete lines

- go to line 3: type `:3` in command-line mode or type `3G` in command mode
- delete from line number 3 to 5 by typing `:3,5d`
- delete from line number 3 to end of file by typing `:3,$d`

10. **sed** (short for Stream EDitor) to find and replace text

In [2]:
```
## similar option in vi, e.g., s/string1/string2/g
## to replace string1 with string2 in the entire document

! ls
! ls > LIST
```

```
2022                    generateCSV.sh           linux_basics.tex
LIST                    generated8shell.csv      linux_basics.texe
LIST.hs1                img                      linux_shell.ipynb
LIST.hs2                linux_basics.aux         linux_shell.pdf
LIST.hs3                linux_basics.ipynb       linux_shell.slides.html
STD                     linux_basics.log         missfont.log
STDERR                  linux_basics.out         test1
demo.csv                linux_basics.pdf         tmp
demo.fas                linux_basics.slides.html workspace
```

```
In [3]: ! sed 's/linux/os/g' LIST
```

```
2022
LIST
LIST.hs1
LIST.hs2
LIST.hs3
STD
STDERR
demo.csv
demo.fas
generateCSV.sh
generated8shell.csv
img
os_basics.aux
os_basics.ipynb
os_basics.log
os_basics.out
os_basics.pdf
os_basics.slides.html
os_basics.tex
os_basics.texe
os_shell.ipynb
os_shell.pdf
os_shell.slides.html
missfont.log
test1
tmp
workspace
```

11. print and count file context

- `cat` will not be a good option, if the file contains too many lines.
- `head filename` or `tail filename` to simply display the first or last lines (default as ten lines).
- More (or less) lines can be displayed, for example, `head -5 filename` or `head -5 filename`.
- `wc -l filename` (short for word count) shows number of lines in file.

```
In [4]: ! head -3 LIST
```

```
2022
LIST
LIST.hs1
```

```
In [5]: ! tail -3 LIST
```

```
test1
tmp
workspace
```

```
In [6]: ! wc -l LIST
```

```
      27 LIST
```

12. use `grep` and `wc` to count sequences

- copy a fasta file from google drive (**https://drive.google.com/file/d/1xQRHVKzlVtHV4vGOxAulfJ2YN5OVDx6r/view?usp=sharing**)
- grep (short for Globally search a Regular Expression and Print)
    - a command-line utility for searching plain-text data sets for lines that match a **regular expression**
    - from **https://en.wikipedia.org/wiki/Grep**
- wc (short for word count)
    - -l print the line count
    - -w print the word count
    - from **https://en.wikipedia.org/wiki/Wc_(Unix)**
- pipe | : The pipe is used to combine two or more commands. The output of the first command acts as input to the next one.

```
In [1]: #####
        # Question 1. How many sequences in a fasta/fastq file?
# Try to grep header lines and wc lines, and get a count of 1000
#####
! grep ">" demo.fas | wc -l
```

```
1000
```

```
In [1]: #####
        # only print the first or last 3 sequences
#####
! grep ">" demo.fas | head -3
```

```
>hCoV-19/USA/CA-CZB-2021/2020 | EPI_ISL_486273 | 2020-05-04 | NorthAmerica
>hCoV-19/Wuhan/WIV04/2019 | EPI_ISL_402124 | 2019-12-30 | China
>hCoV-19/India/GJ-GBRC99/2020 | EPI_ISL_450788 | 2020-05-06 | Asia
```

```
In [2]: ! grep ">" demo.fas | tail -3
```

```
>hCoV-19/Peru/LIM-INS-155/2020 | EPI_ISL_536548 | 2020-03-19 | SouthAmerica
>hCoV-19/Scotland/EDB6157/2020 | EPI_ISL_473893 | 2020-05-14 | Europe
>hCoV-19/Switzerland/BS-ETHZ-110001/2020 | EPI_ISL_486443 | 2020-04-01 | Europe
```

Reminder: grep is a case insensitive search

- Differences among the following commands?

```
In [4]: ! grep -i Usa demo.fas | wc -l #--ignore-case
        ! grep USA demo.fas | wc -l
! grep usa demo.fas | wc -l
```

```
211
211
  0
```

use `sed` to replace the separator

- Usage:
  - sed 's/regexp/replacement/g' inputFileName > outputFileName
  - s: substitution; g: global
  - Tips: space is a char; you need a backslash "\" for slash "/" in sed

In [5]:
```
# toy example: replace xyz with abc
! echo xyz | sed 's/x/a/g' | sed 's/y/b/g' | sed 's/z/c/g'
```

```
abc
```

- Current format (start with ">" and split by a pipe "|")
  - e.g., >hCoV-19/USA/CA-CZB-2021/2020 | EPI_ISL_486273 | 2020-05-04 | NorthAmerica
  - virus/location/strain/year | accession_number | collection_date | area
- Expected format (split by a semicolon without the space)
  - virus,location,strain,year,accession_number,collection_date,area

In [7]:
```
# double check metadata
! grep ">" demo.fas | head -1
```

```
>hCoV-19/USA/CA-CZB-2021/2020 | EPI_ISL_486273 | 2020-05-04 | NorthAmerica
```

In [12]:
```
#####
        # Step 1. create CSV file
#####
! echo "virus,location,strain,year,accession_number,collection_date,area" > demo.csv
```

In [13]:
```
#####
        # Step 2. retrieve meta data and stream editor
# current format: >hCoV-19/USA/CA-CZB-2021/2020 | EPI_ISL_486273 | 2020-05-04 | NorthAmerica
# expected format: hCoV-19;USA;CA-CZB-2021;2020;EPI_ISL_486273;2020-05-04;NorthAmerica
# 1st sed: remove ">"
# 2nd sed: replace "/" to " | "
# 3rd sed: replace " | " to ","
#####
! grep ">" demo.fas | sed 's/>//g;s/\// | /g;s/ | /,/g' >> demo.csv
```

In [14]:
```
#####
        # final check
#####
! head -3 demo.csv
```

```
virus,location,strain,year,accession_number,collection_date,area
hCoV-19,USA,CA-CZB-2021,2020,EPI_ISL_486273,2020-05-04,NorthAmerica
hCoV-19,Wuhan,WIV04,2019,EPI_ISL_402124,2019-12-30,China
```

13. Shell scripting

- **Shell scripts** allow several commands that would be **entered manually at a command-line interface** to be **executed automatically**, and without having to wait for a user to trigger each stage of the sequence.
- components
    - all of Linux commands
    - variables
    - looping
    - any 3rd-party or user developed programs that can be executed on Linux, like python, R, java
- from **https://en.wikipedia.org/wiki/Shell_script**

The first shell script

1. Use `touch` and `vi` to create and edit a new file
2. Type Shebeng `#!/bin/bash`
3. Type something about you
4. Type the `read` command to take the input from the keyborard and assign to the INPUT variable

In [ ]:
```
#!/bin/bash

# Author : Zac Gong
# Affiliation: CGU
# The first shell script follows here:

echo "What is your name?"
read INPUT
echo "Hello, $INPUT. Good to see you."
```

In [ ]:
```
# how to run your shell (Kahoot)
       ! cp /home/public/ngs/script/input_name.sh ./
! bash ./input_name.sh
```

In [ ]:
```
# or just type the file name
       ! ./input_name.sh
```

Shell Basic Operators

- All the following tables from **https://www.tutorialspoint.com/unix/unix-basic-operators.htm**
- **Arithmetic Operators**

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator	`expr $a + $b` will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand	`expr $a - $b` will give -10
* (Multiplication)	Multiplies values on either side of the operator	`expr $a \* $b` will give 200
/ (Division)	Divides left hand operand by right hand operand	`expr $b / $a` will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr $b % $a` will give 0
= (Assignment)	Assigns right operand in left operand	a = $b would assign value of b into a
== (Equality)	Compares two numbers, if both are same then returns true.	[$a == $b] would return false.
!= (Not Equality)	Compares two numbers, if both are different then returns true.	[$a != $b] would return true.

In [ ]:
```
#!/bin/bash

val=`expr 1 + 1`
echo "Sum : $val"
```

```
## The above script will generate the following result
## Sum : 2
```

In [ ]:
```
! cp /home/public/ngs/script/arith_opt.sh ./
! bash ./arith_opt.sh
```

Relational Operators

- Assume variable a holds 10 and variable b holds 20

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b] is not true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b] is true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b] is not true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b] is true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b] is not true.
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b] is true.

String Operators

- Assume variable a holds 10 and variable b holds 20

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a = $b] is not true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[$a != $b] is true.
-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[-z $a] is not true.
-n	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[-n $a] is not false.
str	Checks if **str** is not the empty string; if it is empty, then it returns false.	[$a] is not false.

Boolean Operators

- Assume variable a holds 10 and variable b holds 20

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical **OR**. If one of the operands is true, then the condition becomes true.	[$a -lt 20 -o $b -gt 100] is true.
-a	This is logical **AND**. If both the operands are true, then the condition becomes true otherwise false.	[$a -lt 20 -a $b -gt 100] is false.

File Test Operators

- Assume variable a holds 10 and variable b holds 20

Operator	Description	Example
-b file	Checks if file is a block special file; if yes, then the condition becomes true.	[-b $file] is false.
-c file	Checks if file is a character special file; if yes, then the condition becomes true.	[-c $file] is false.
-d file	Checks if file is a directory; if yes, then the condition becomes true.	[-d $file] is not true.
-f file	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[-f $file] is true.
-g file	Checks if file has its set group ID (SGID) bit set; if yes, then the condition becomes true.	[-g $file] is false.
-k file	Checks if file has its sticky bit set; if yes, then the condition becomes true.	[-k $file] is false.
-p file	Checks if file is a named pipe; if yes, then the condition becomes true.	[-p $file] is false.
-t file	Checks if file descriptor is open and associated with a terminal; if yes, then the condition becomes true.	[-t $file] is false.
-u file	Checks if file has its Set User ID (SUID) bit set; if yes, then the condition becomes true.	[-u $file] is false.
-r file	Checks if file is readable; if yes, then the condition becomes true.	[-r $file] is true.
-w file	Checks if file is writable; if yes, then the condition becomes true.	[-w $file] is true.
-x file	Checks if file is executable; if yes, then the condition becomes true.	[-x $file] is true.
-s file	Checks if file has size greater than 0; if yes, then condition becomes true.	[-s $file] is true.
-e file	Checks if file exists; is true even if file is a directory but exists.	[-e $file] is true.

Shell Decision Making

- if...else is a commonly-used, decision-making statement.
    - if...fi statement
    - if...else...fi statement
    - if...elif...else...fi statement

In [ ]:
```
#!/bin/bash

# Author : Zac Gong
# Affiliation: CGU
# The first shell script follows here:

echo "What is your name?"
read INPUT

if [ "$INPUT" == ""]; then
    echo "Null input. Program stopped."
    exit
else
    echo "Hello, $INPUT. Good to see you."
fi
```

In [ ]:
```
! cp /home/public/ngs/script/input_name_decision_making.sh ./
! bash input_name_decision_making.sh
```

use getopts to parse command-line arguments and display your input and output file names

- getopts command: retrieve options and their arguments
- for example, two required arguments as input and output file names

In [ ]:
```
! cp /home/public/ngs/script/print_io.sh ./
! bash print_io.sh -i input.fas -o output.fas
```

In [ ]:
```
#!/bin/bash
# Retrieve file names and print out

### get args
while getopts i:o: option
do
case "${option}"
in
i) i=${OPTARG};;
o) o=${OPTARG};;
esac
done

### check args
if [ "$i" == "" -o "$o" == "" ]
then
        echo "Input Error, please make sure"
        echo "-i as input file name"
        echo "-o as output file name"
        echo "Usage: sh printInputOutput.sh -i input.fas -o output.fas"
        exit
fi

echo "input file name: "$i
echo "output file name: "$o
```

Read a fasta file and generate a CSV file

In [ ]:
```
! cp /home/public/ngs/script/generate_csv.sh ./
! bash generate_csv.sh -i input.fas -o output.fas
```

In [ ]:
```
#!/bin/bash
# This script takes fasta data to generate CSV file.

### get args
while getopts i:o: option
do
case "${option}"
in
i) i=${OPTARG};;
o) o=${OPTARG};;
esac
done

### check args
if [ "$i" == "" -o "$o" == "" ]
then
        echo "Input Error, please make sure"
        echo "-i as input file name"
        echo "-o as output file name"
        echo "Usage: bash generate_csv.sh -i input.fas -o output.fas"
        exit
fi

echo "input file name: "$i
echo "output file name: "$o

# step 1. create CSV file
echo "virus,location,strain,year,accession_number,collection_date,area" > $o
# step 2. retrieve meta data and stream editor
grep ">" $i | sed 's/>//g' | sed 's/\// | /g' | sed 's/ | /,/g' >> $o
```

In [3]:
```
# execute this shell script
! bash generate_csv.sh -i demo.fas -o metadata.csv
```

```
input file name: demo.fas
output file name: metadata.csv
```

In [4]:
```
! head -3 metadata.csv
```

```
virus,location,strain,year,accession_number,collection_date,area
hCoV-19,USA,CA-CZB-2021,2020,EPI_ISL_486273,2020-05-04,NorthAmerica
hCoV-19,Wuhan,WIV04,2019,EPI_ISL_402124,2019-12-30,China
```

Q&A