

BI0609 Introduction to UNIX/Linux and Bash Scripting

31st October, 2022

Deepak Kumar Tanwar

URPP Evolution in Action
Embedded bioinformatician

 @d_k_tanwar

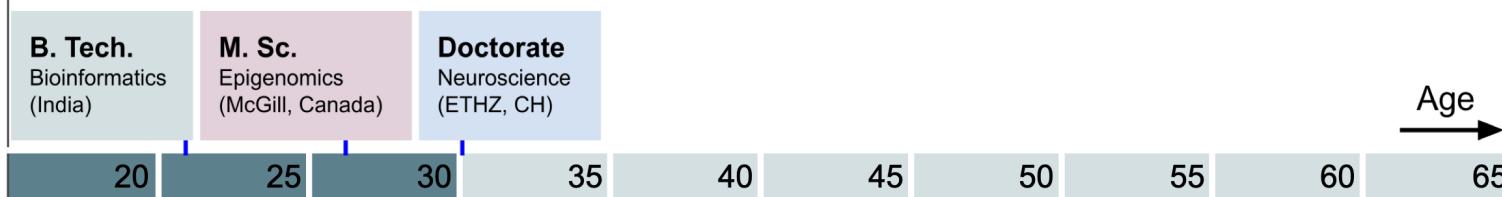
 deepak.tanwar@evolution.uzh.ch

Deepak Tanwar (embedded bioinformatics scientist)

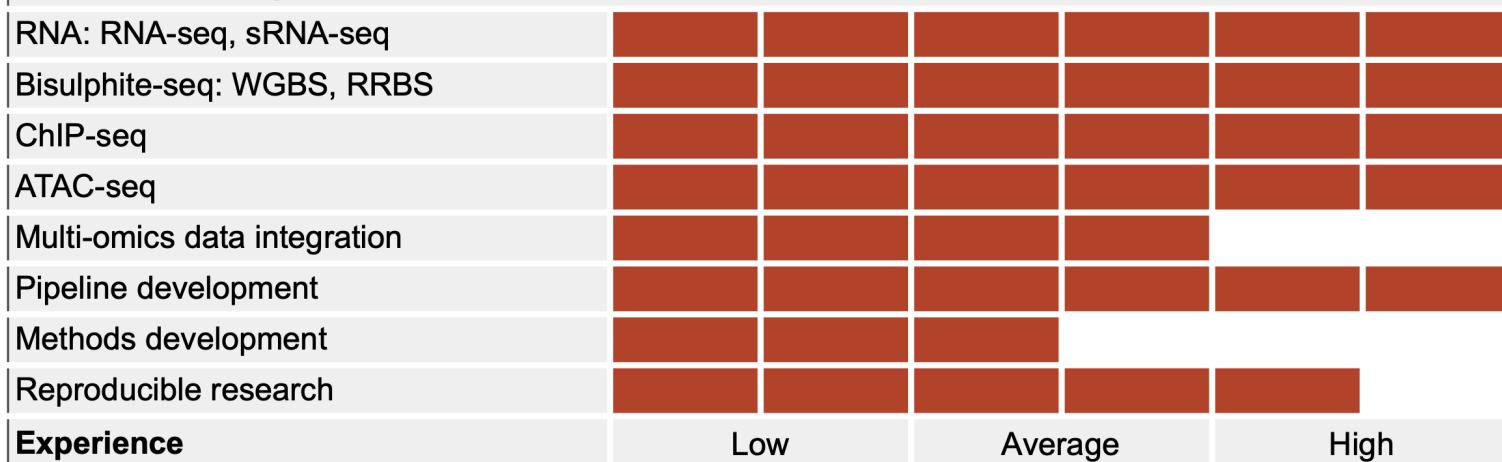


Deepak is a bioinformatician at the URPP Evolution in Action. Deepak has a strong background in multi-omics research and has expertise in reproducible data analysis, benchmarking, and methods development.

Education timeline



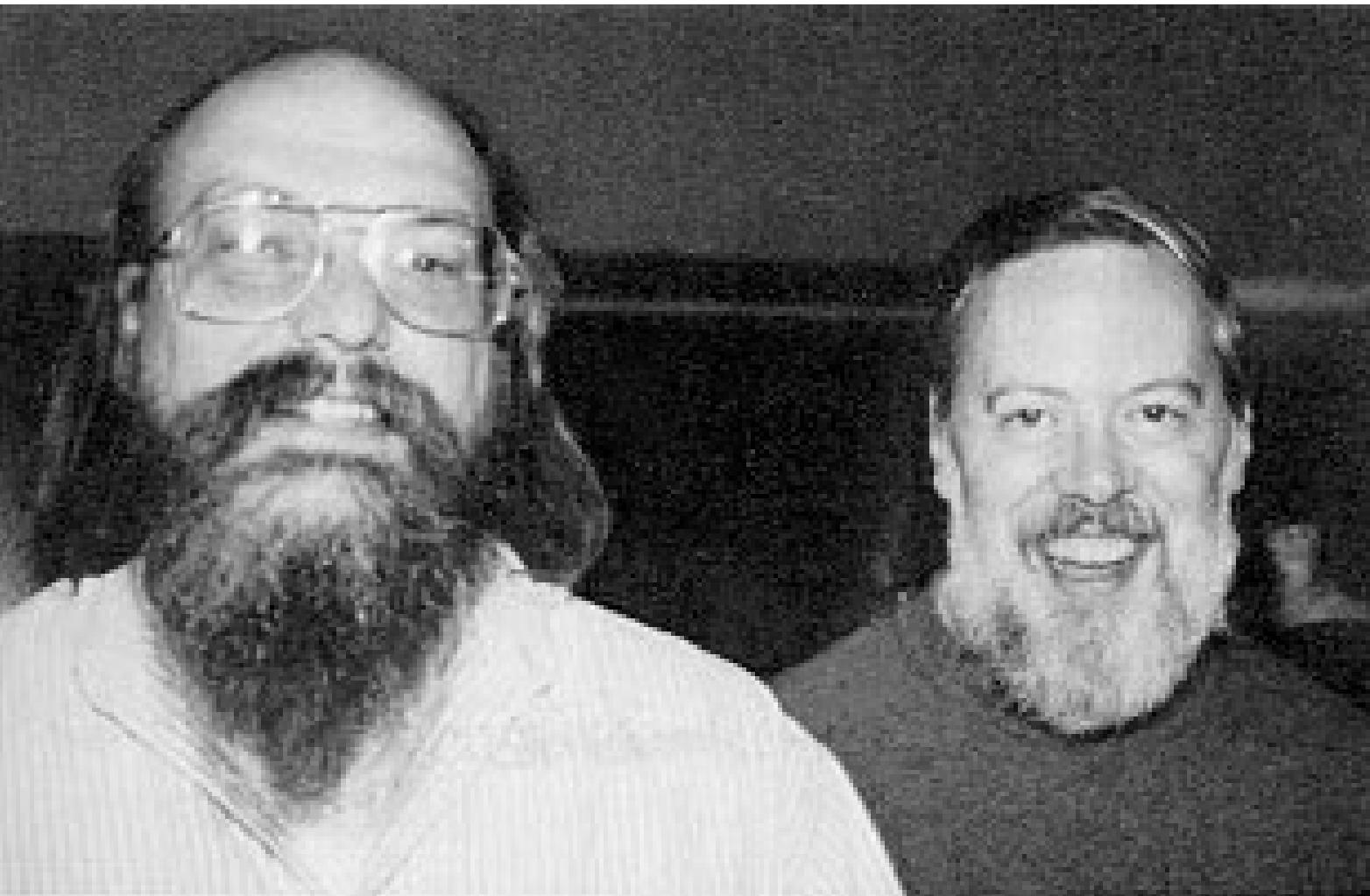
Multi-omics experience



Distribution of time between bench work and computational work



UNIX





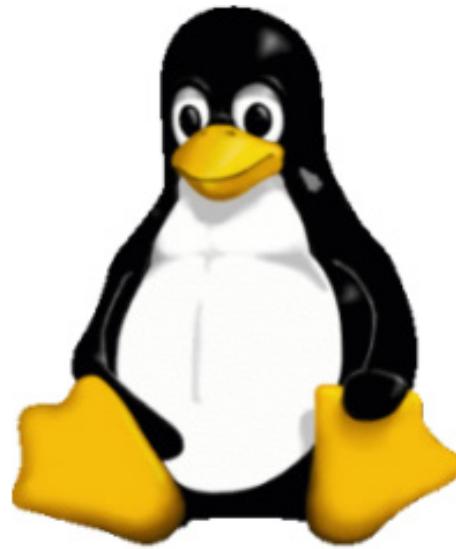
Ken Thompson

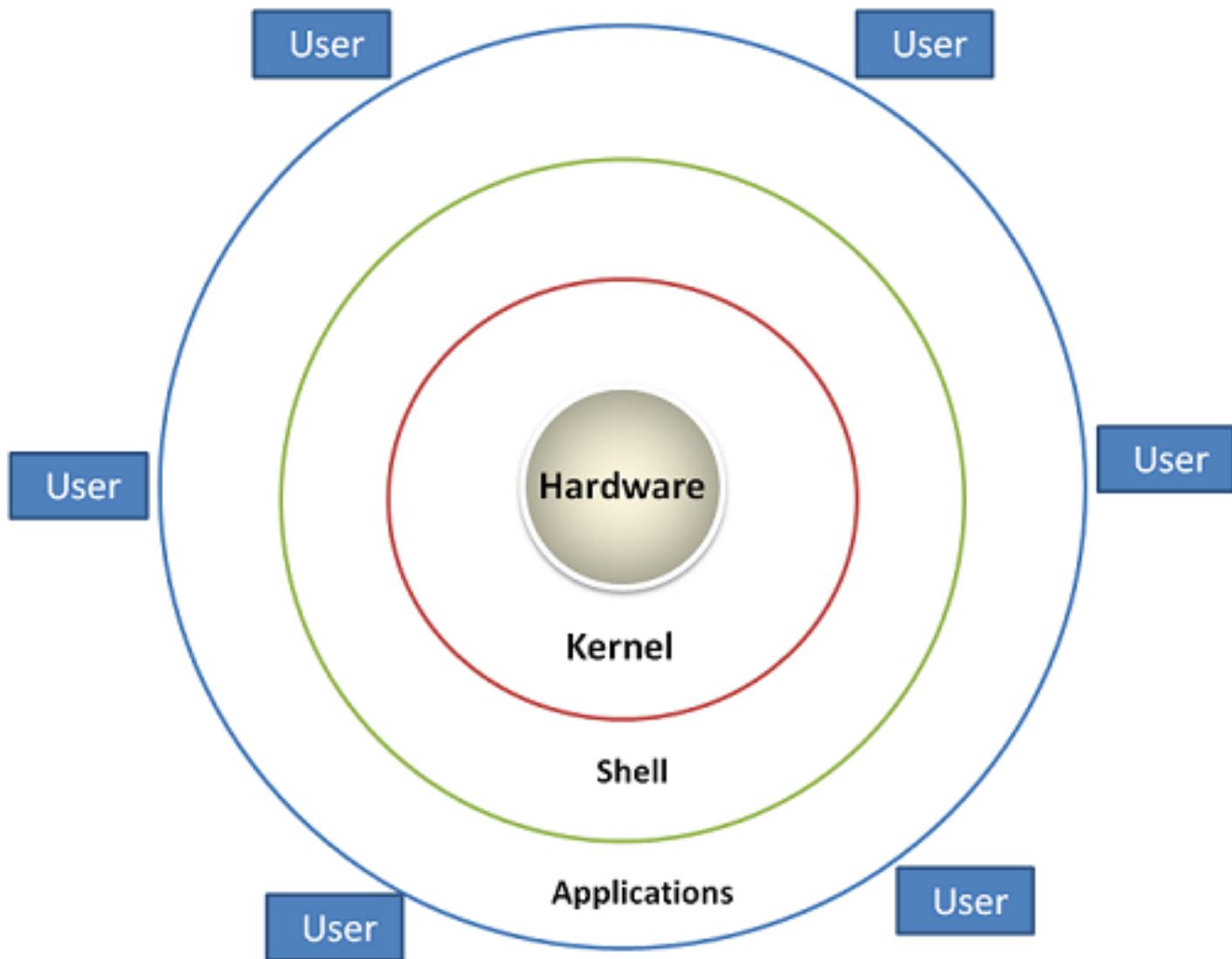
Dennis Ritchie





Linus Torvalds





TecAdmin.net

Linux System Architecture

KERNEL VERSUS SHELL

KERNEL

A computer program which acts as the core of the computer's operating system and has the control over everything in the system

Core of the system that controls all the tasks of the system

Does not have types

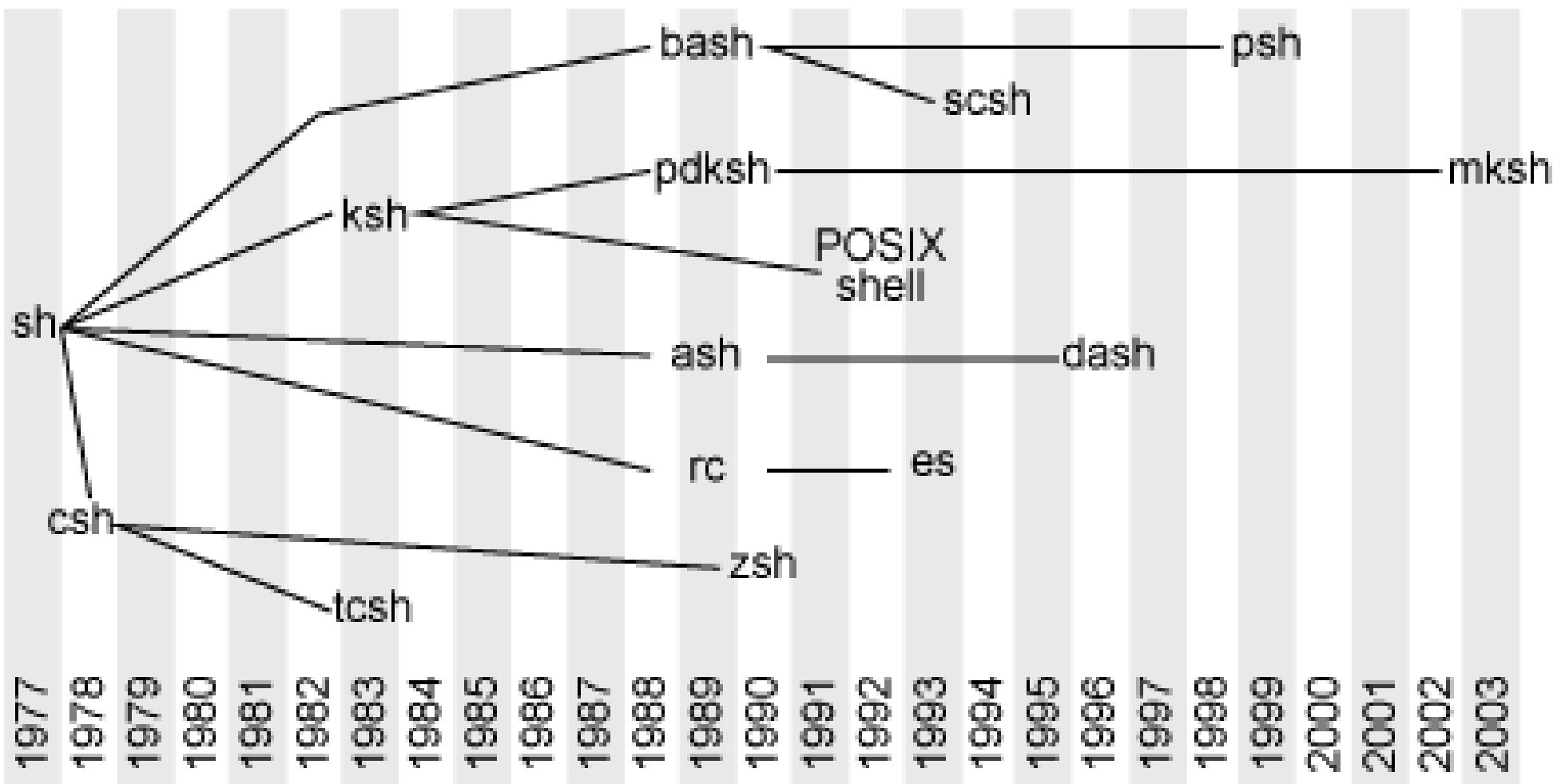
SHELL

A computer program which works as the interface to access the services provided by the operating system

Interface between the kernel and user

Has types such as Bourne shell, C shell, Korn Shell, Bourne Again Shell, etc.

Evolution of shells in Linux



Google Books Ngram Viewer

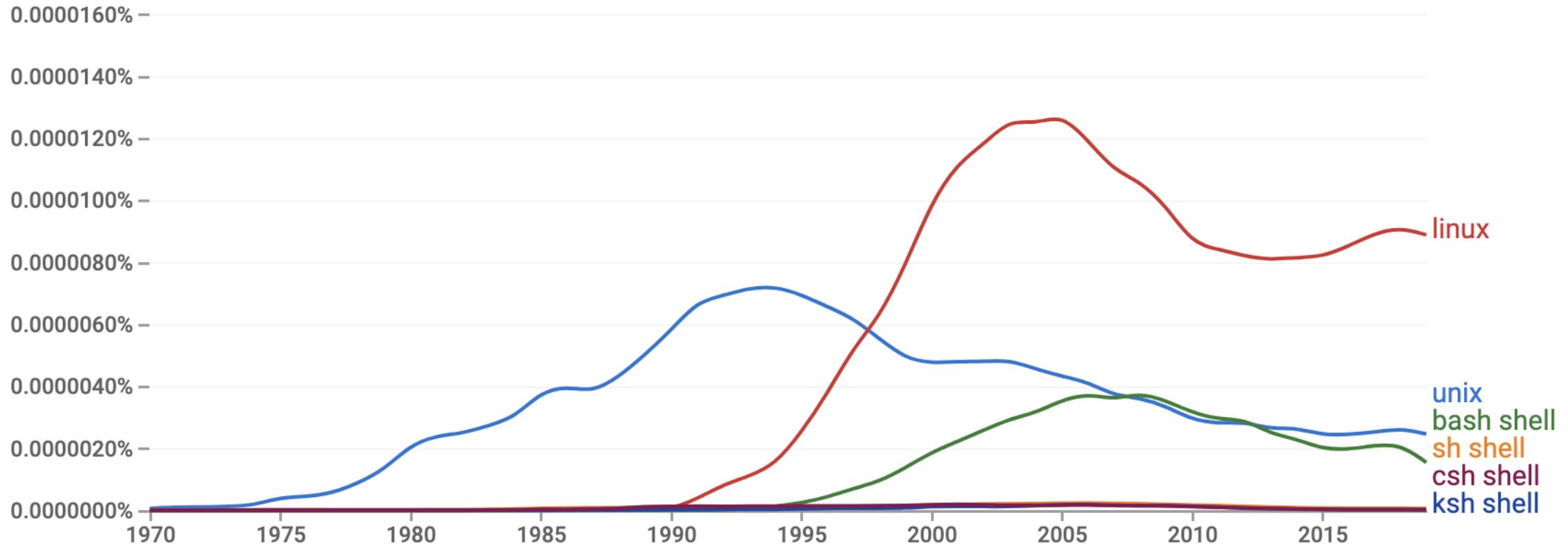
unix,linux,bash shell,sh shell,ksh shell,csh shell

1965 - 2019 ▾

English (2019) ▾

Case-Insensitive

Smoothing of 3 ▾

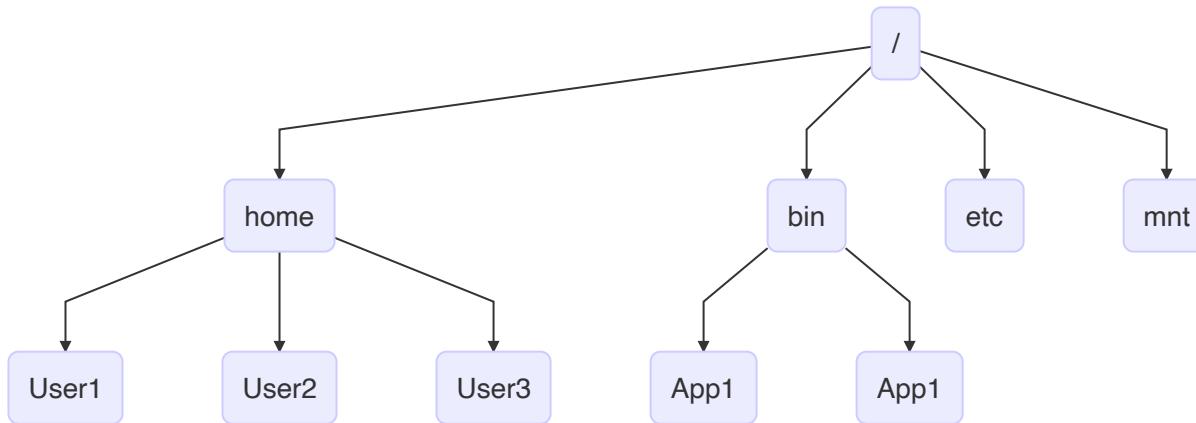


Bourne again shell

Bourne again shell

Directory structure and Bash

Directory structure



Linux command-line has a structure

```
student@biobox:~$
```

PROMPT

Linux command-line has a structure

```
student@biobox:~$ command option argument
```

PROMPT

ls

-l

/

Linux command-line has a structure

student@biobox:~\$ command option argument

PROMPT

ls

-l

/

Name two folders that I should see after running the above command.

Exercise

Enter into the Docker container. [`./login_student.sh`](#)

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Display the content in "long listing format" + "sort by modification time". `ls -l -t`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Display the content in "long listing format" + "sort by modification time". `ls -l -t`

Display the content in "long listing format" + "sort by modification time" + "reverse the order of sorting". `ls -l t -r`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Display the content in "long listing format" + "sort by modification time". `ls -l -t`

Display the content in "long listing format" + "sort by modification time" + "reverse the order of sorting". `ls -l t -r`

Print the working directory using pwd command. `pwd`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Display the content in "long listing format" + "sort by modification time". `ls -l -t`

Display the content in "long listing format" + "sort by modification time" + "reverse the order of sorting". `ls -l t -r`

Print the working directory using pwd command. `pwd`

Check your user name by whoami command. `whoami`

Exercise

Enter into the Docker container. `./login_student.sh`

List the files. `ls`

Change directory to root using cd command. `cd /`

Display the content of root. `ls`

Look for help for ls command using option --help. `ls --help`

Display the content in "long listing format". `ls -l`

Display the content in "long listing format" + "sort by modification time". `ls -l -t`

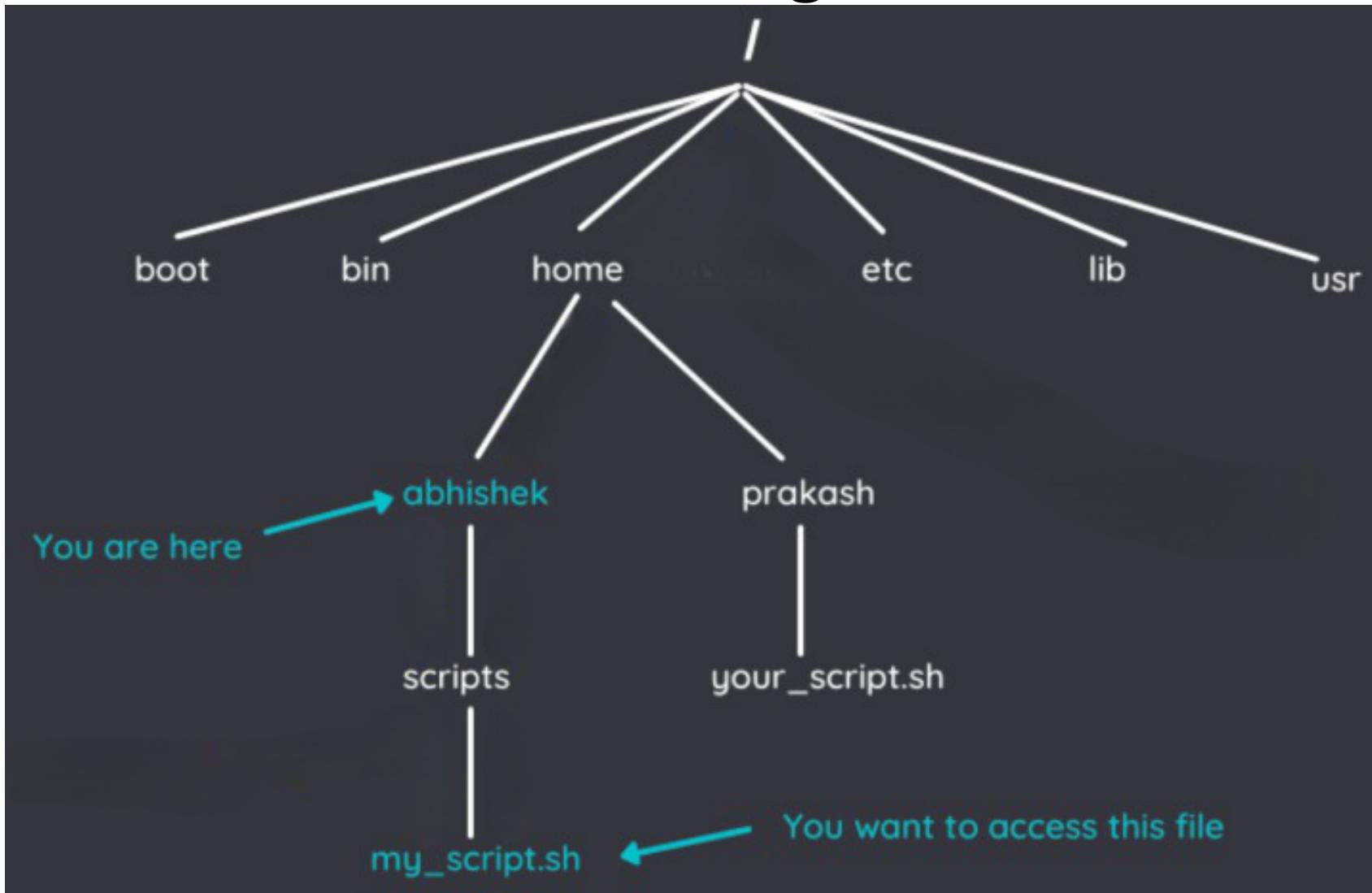
Display the content in "long listing format" + "sort by modification time" + "reverse the order of sorting". `ls -l t -r`

Print the working directory using pwd command. `pwd`

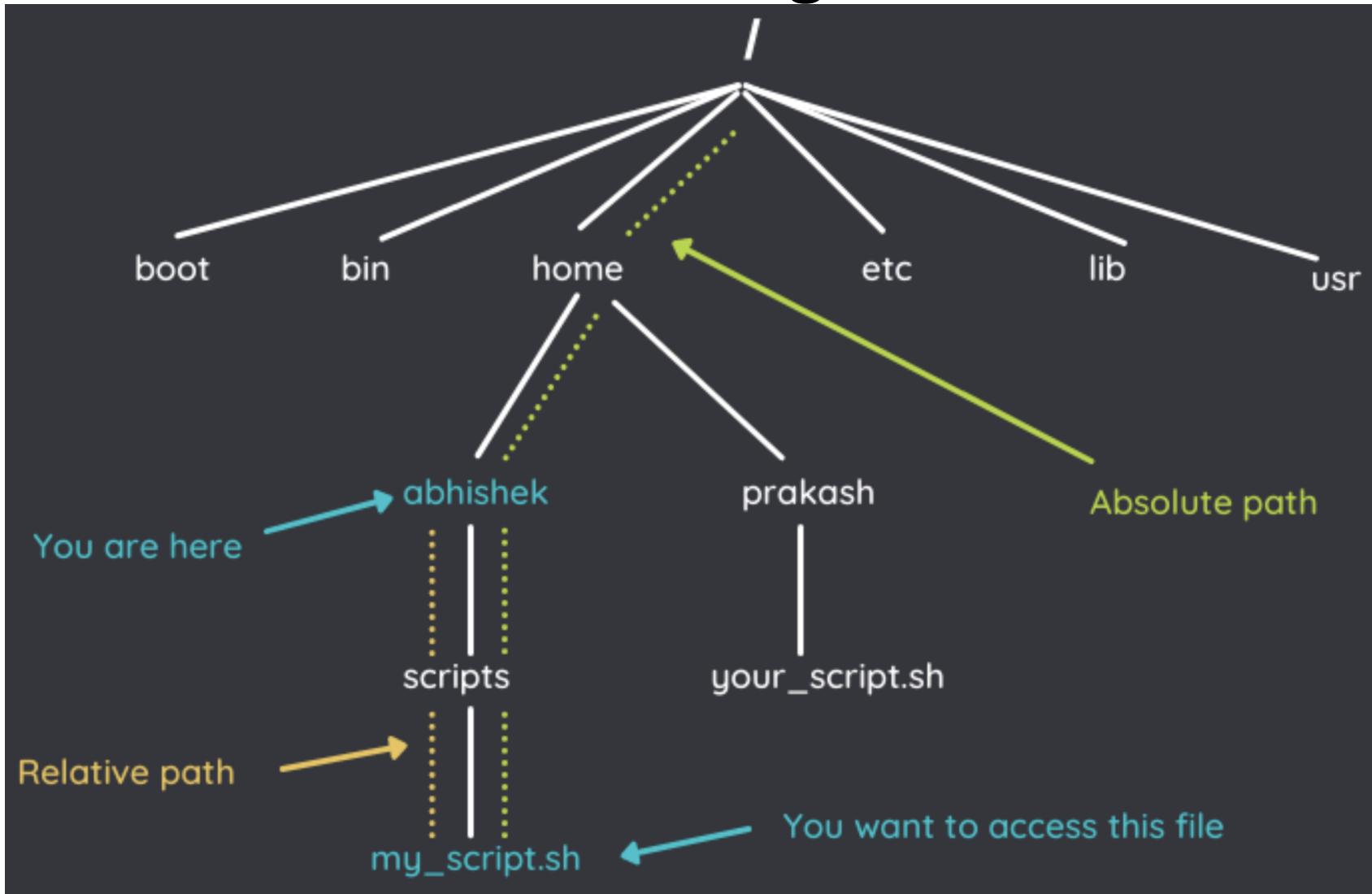
Check your user name by whoami command. `whoami`

Most command names are abbreviations of their function.

Path to navigate



Path to navigate



Some important symbols/ commands

Symbols/ Commands	Descriptions
.	current directory
..	directory up in heirarchy
cd or cd ~ or cd ~/	change directory to home directory

Some important symbols/ commands

Symbols/ Commands	Descriptions
touch	create a file
*	list out all the files
List?	any file that begins with List followed by 1 character

Exercise

Make a folder in your home directory: test. Enter into the test folder. Create some files using touch command: test1.txt, test2.txt, abc.txt, DEF.txt, 1file.txt, and bcd.txt. Display all files, display all test files using ?, and display files starting with a.

Exercise

Make a folder in your home directory: test. Enter into the test folder. Create some files using touch command: test1.txt, test2.txt, abc.txt, DEF.txt, 1file.txt, and bcd.txt. Display all files, display all test files using ?, and display files starting with a.

```
cd ~/ # go to home directory
mkdir test # make a test directory
cd test # change directory
touch test1.txt test2.txt abc.txt DEF.txt 1file.txt bcd.txt # create files
ls # display all files
ls test?.txt # display all test files
ls a* # display all files starting with a
```

Some important symbols/ commands

Symbols/ Commands	Descriptions
[abc]*	any file that begins with either ‘a’ or ‘b’ or ‘c’ and ending with any number of characters
[[:upper:]]*	any file that begins with an uppercase letter
[![:upper:]]*	any file that does not begin with an uppercase letter
[[:digit:]]*	any file that begins with digit
[![:digit:]]*	any file that does not begin with digit
[a-d]???	any file that begins from a range of a-d and followed by exactly 3 character

Some important symbols/ commands

Symbols/ Commands	Descriptions
echo	display argument
\$*	stored variables
history	display history of your activity

Exercise

Print "Hello World!". Store "Hello World!" in a variable named `hello`. Display the history of your activity.

Exercise

Print "Hello World!". Store "Hello World!" in a variable named `hello`. Display the history of your activity.

```
echo "Hello World!" # print  
hello="Hello World!" # store into a variable  
echo $hello # print content of variable  
history # display history of your activity
```

Some important symbols/ commands

Symbols/ Commands	Descriptions
mkdir	make a directory
rmdir	remove a directory
rm	remove files/ directories
cp	copy files/ directories
mv	move files/ directories
wget	download file from internet

Some important symbols/ commands

Symbols/ Commands	Descriptions
less, more, head, tail, cat	show content
grep, cut, uniq, sort	search or extract or modify or sort
tr	string manipulation
wc	count

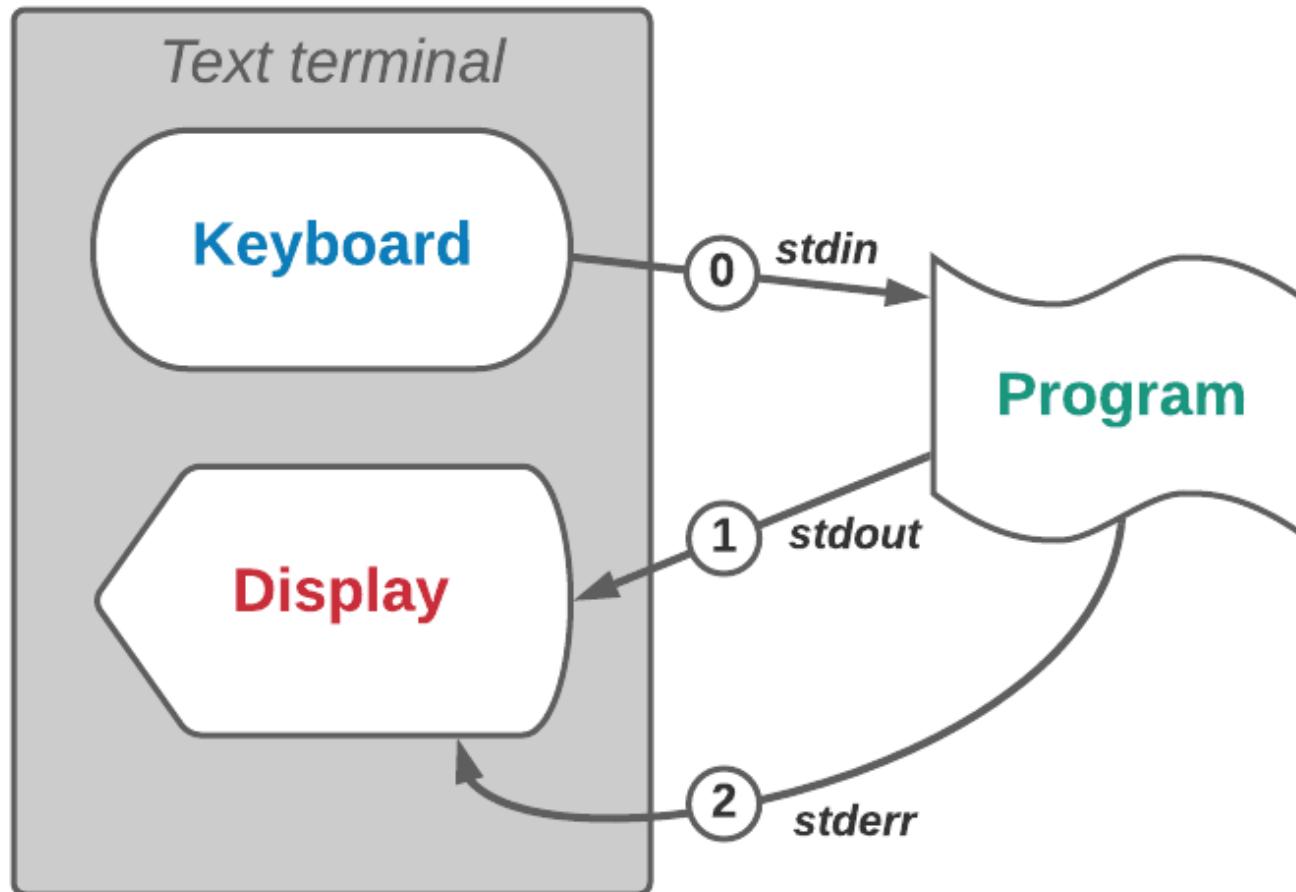
Some important symbols/ commands

Symbols/ Commands	Descriptions
less, more, head, tail, cat	show content
grep, cut, uniq, sort	search or extract or modify or sort
tr	string manipulation
wc	count

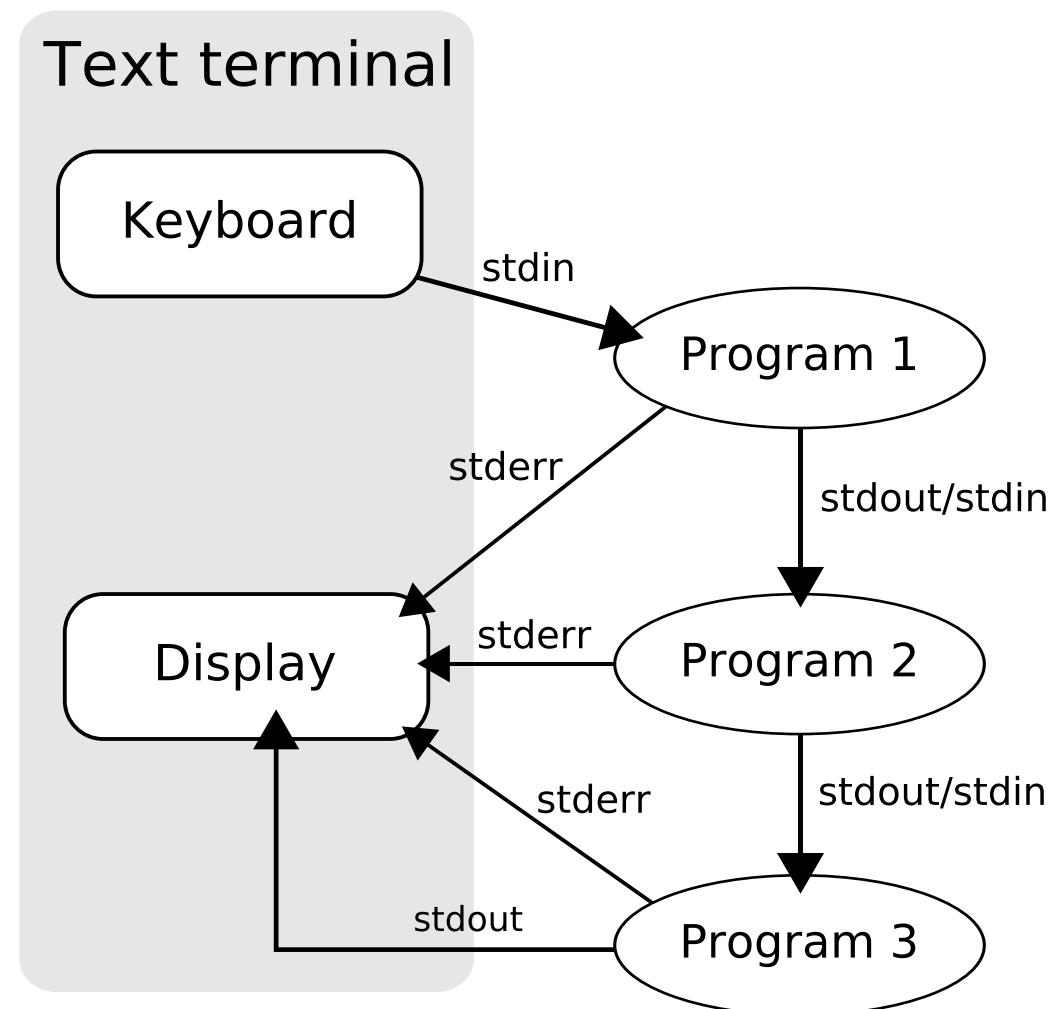
arabidopsis.org/download_files/Genes/TAIR10_genome_release/TAIR10_gff3/TAIR10_GFF3_genes.gff

Redirection

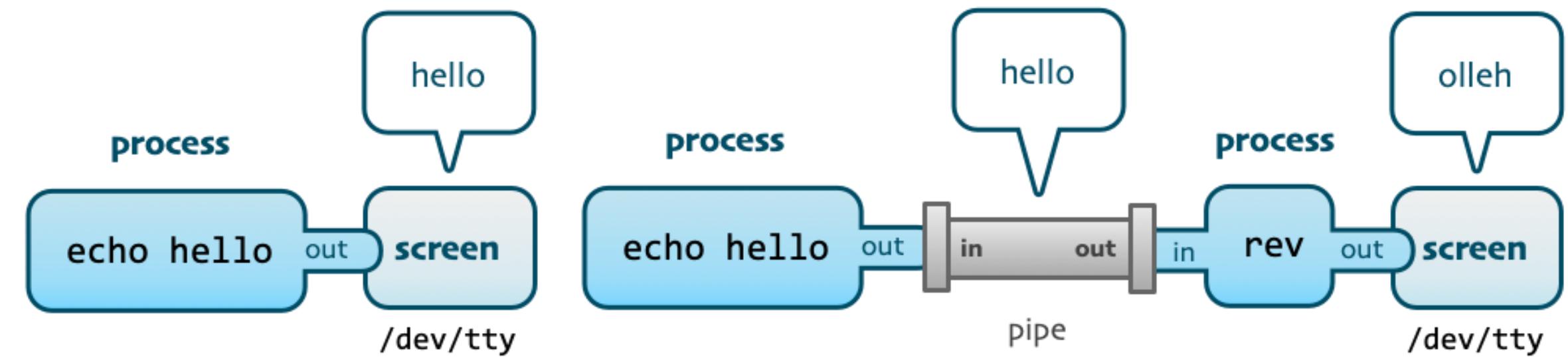
STDIN, STDOUT, and STDERR



STDIN, STDOUT, and STDERR



Pipes



Redirection

Symbols/ Commands	Descriptions
>	Standard output redirection
<	Standard Input redirection
2>	Standard error redirection
&2>	Standard output and error redirection
>>	Standard output redirection
<<	Standard input redirection
	Redirection using pipes

Bash scripting

Text editors

- nano
- vim
- emacs

Bash scripts

```
#!/bin/bash  
  
# This script will print ...  
  
echo "Hello World!"
```

```
# shebang line  
  
# a comment  
  
# code
```

Bash scripts

```
#!/bin/bash  
  
# This script will print ...  
  
echo "Hello World!"
```

shebang line
a comment
code

```
#!/bin/bash  
  
# This script will print my name  
  
name="Deepak Tanwar"  
  
echo "Hello $name!"
```

shebang line
a comment
store name in a variable
print name

Bash script

Using nano, write a script to print your name.

```
nano name.sh # create a file using nano  
#!/bin/bash  
echo "Deepak Tanwar"  
# close the file with ctrl + x
```

Bash script

Using nano, write a script to print your name.

```
nano name.sh # create a file using nano  
#!/bin/bash  
echo "Deepak Tanwar"  
# close the file with ctrl + x
```

Redirect output as STDOUT: name.stdout.

```
bash name.sh > name.stdout
```

Bash script

Redirect output as STDERR: name.stderr.

```
bash name.sh 2> name.stderr
```

Bash script

Redirect output as STDERR: **name.stderr**.

```
bash name.sh 2> name.stderr
```

Append **name.stdout** with your favourite animal.

```
echo "Tiger" >> name.stdout
```

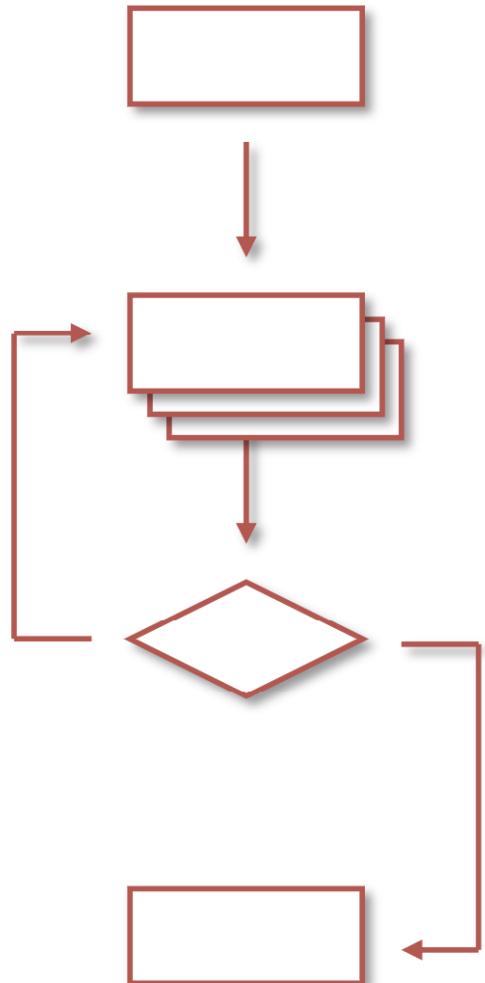
Bash script

Append `name.stderr` with your favourite food.

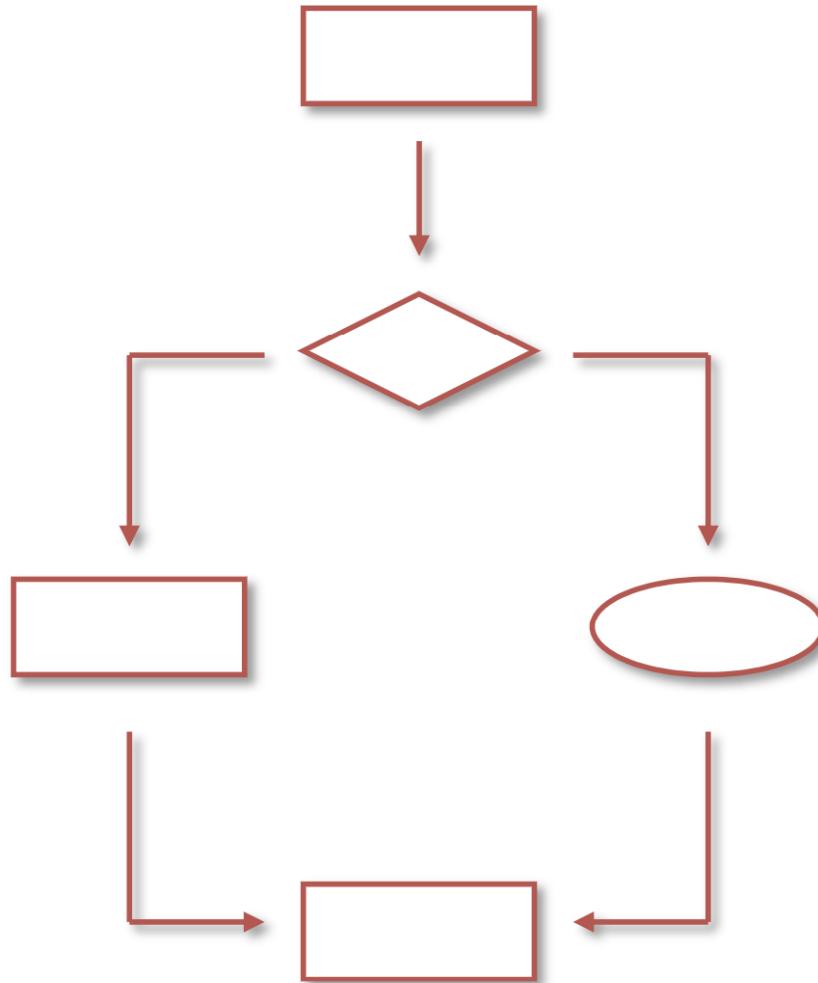
```
echo "Fondue" 2>> name.stderr
```

Flow control

repetition



selection



for loop

```
for VARIABLE in 1 2 3  
do  
    command1  
done
```

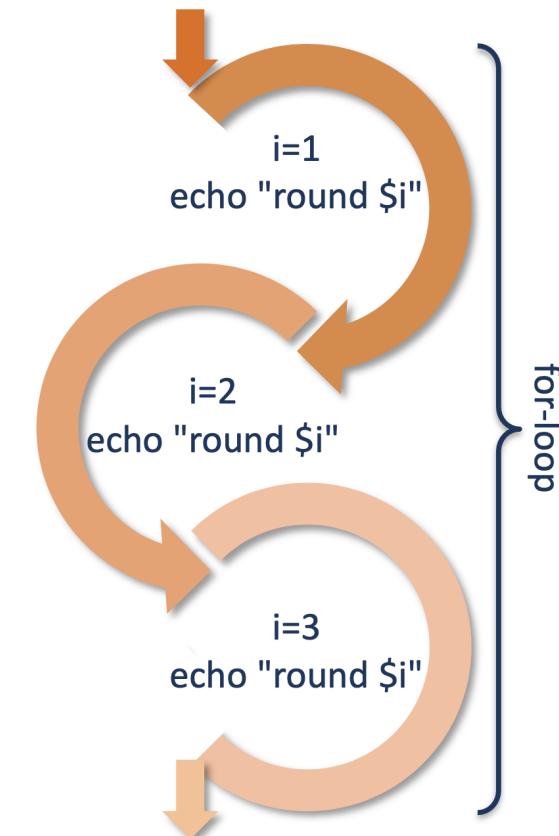
- Can be used to repeat certain tasks

- Examples:

```
#!/bin/bash  
for i in 1 2 3  
do  
    echo "round $i"  
done
```

```
#!/bin/bash  
for i in {1..3}  
do  
    echo "round $i"  
done
```

```
round 1  
round 2  
round 3
```



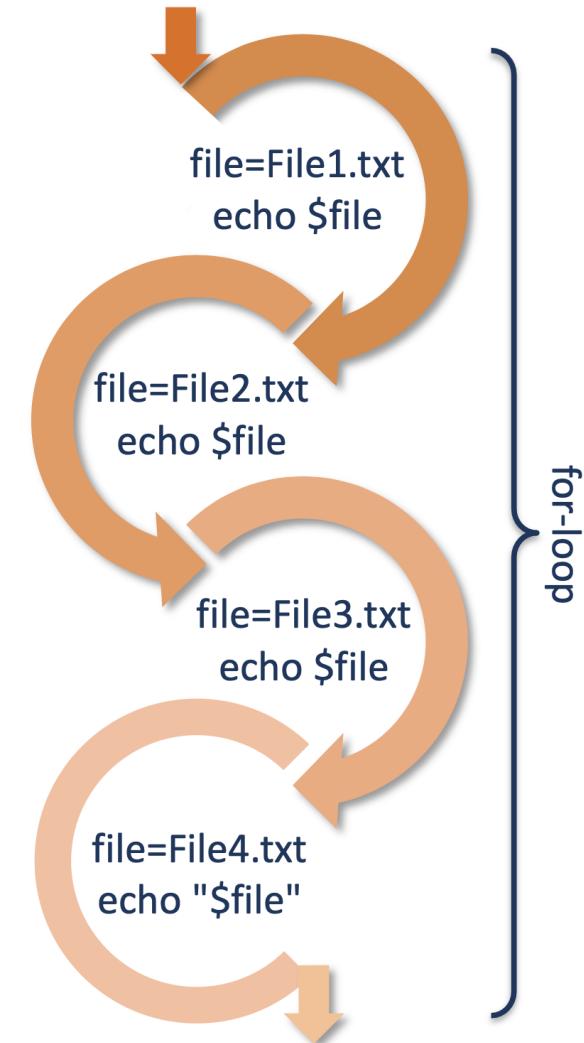
for loop

- Examples:

Go through all files in a folder:

```
#!/bin/bash
for file in /home/user/*
do
    echo $file
done
```

```
File1.txt
File2.txt
File3.txt
File4.txt
```



while loop

- Can also be used to repeat certain tasks:

```
while [[ something ]]  
do  
    command  
done
```

- Example:

```
#!/bin/bash  
count=0  
while [[ $count -lt 4 ]]  
do  
    echo $count  
    let count+=1  
done  
echo "done"
```

```
0  
1  
2  
3  
done
```

if-else

```
if [[ someting ]]
then
    command1
elif [[ something ]]
then
    command2
else
    command3
fi
```

if-else

Boolean operators:

- **-e FILE:** True if file exists

```
#!/bin/bash
if [[ -e log.txt ]]
then
    echo "log file exist"
else
    echo "log file doesn't exist"
fi
```

Setting path and permissions

Setting path

```
export PATH="/path/to/dir:$PATH"
```

Setting path

```
export PATH="/path/to/dir:$PATH"
```

```
export PATH="/path/to/dir:$PATH" >> ~/.bash_profile
```

Setting path

```
export PATH="/path/to/dir:$PATH"
```

```
export PATH="/path/to/dir:$PATH" >> ~/.bash_profile
```

```
export PATH="/path/to/dir:$PATH" >> ~/.bashrc
```

Permissions and changing permissions

```
$ ls -l
```

-rw-r--r--	1	swyder	staff	6677	30	Okt	22:02	At.gff
-rw-r--r--@	1	swyder	staff	3723486	28	Okt	17:30	CheatSheetguideUNIX.pdf
drwxr-xr-x	11	swyder	staff	374	11	Nov	16:13	FASTAS
-rw-r--r--	1	swyder	staff	21	30	Okt	22:04	indA.txt

Permissions

Owner

Group

Last modification

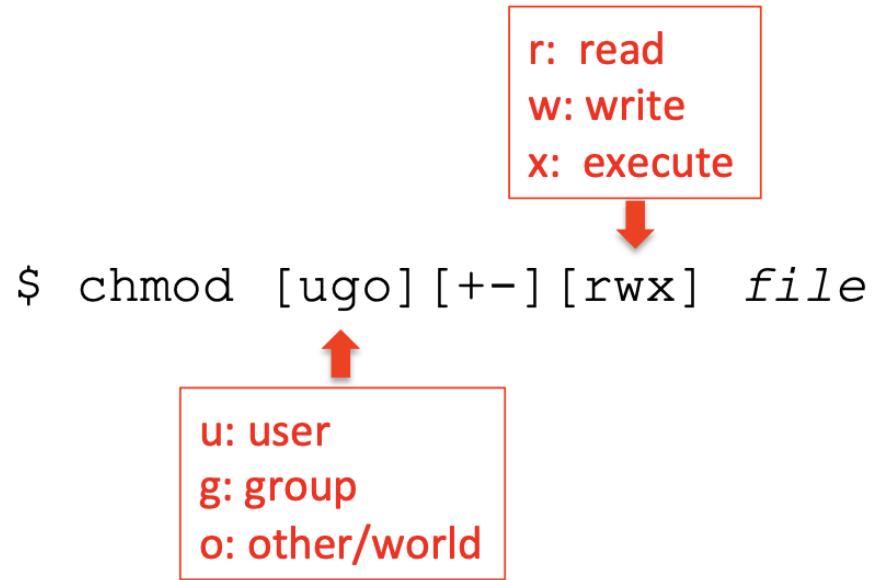
File size
(bytes)

File type

d : directory

- : regular file

Permissions and changing permissions



Make a file executable (for you)

```
$ chmod +x file  
chmod ug+rx my_file
```

Setting exact permission (only reading, for you)

```
$ chmod =r file
```

Removing permissions (for you)

```
$ chmod -wx file
```

Permissions and changing permissions

Owner	Group	Other
r w x	r w x	r - x
4+2+1	4+2+1	4+0+1

r: read
w: write
x: execute



7 7 5

`chmod 775 file`

Power of &&

Do a **task2** only after **task1** is completed

Power of &&

Do a **task2** only after **task1** is completed

Not working example

```
echo "Hello"; echo "World"
```

Power of &&

Do a **task2** only after **task1** is completed

Not working example

```
echo "Hello"; echo "World"
```

Working example

```
echo "Hello" && echo "World"
```

Testing your knowledge

Problem 1

PFAM is a domain database. It also includes domains that have already been computed for all proteomes. The most recent version is available here:

<https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/>. Each file represents a proteome, which is recognized by its taxonomy ID. The ID 9606 belongs to a human. Each of these files is tab-delimited, with the domain ID in the sixth column. Using wget, download the human proteome file. After downloading, run a single line of bash commands to determine the number of domain types (unique domains) in the human genome. You may use as many commands as you want, connected together in pipes.

Problem 1

PFAM is a domain database. It also includes domains that have already been computed for all proteomes. The most recent version is available here:

<https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/>. Each file represents a proteome, which is recognized by its taxonomy ID. The ID 9606 belongs to a human. Each of these files is tab-delimited, with the domain ID in the sixth column. Using wget, download the human proteome file. After downloading, run a single line of bash commands to determine the number of domain types (unique domains) in the human genome. You may use as many commands as you want, connected together in pipes.

```
 wget https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/9606.tsv.gz
```

Problem 1

PFAM is a domain database. It also includes domains that have already been computed for all proteomes. The most recent version is available here:

<https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/>. Each file represents a proteome, which is recognized by its taxonomy ID. The ID 9606 belongs to a human. Each of these files is tab-delimited, with the domain ID in the sixth column. Using wget, download the human proteome file. After downloading, run a single line of bash commands to determine the number of domain types (unique domains) in the human genome. You may use as many commands as you want, connected together in pipes.

```
wget https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/9606.tsv.gz
```

```
zcat 9606.tsv.gz | cut -f 6 | grep -v "#" | less | sort | uniq | wc -l
```

Problem 1

PFAM is a domain database. It also includes domains that have already been computed for all proteomes. The most recent version is available here:

<https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/>. Each file represents a proteome, which is recognized by its taxonomy ID. The ID 9606 belongs to a human. Each of these files is tab-delimited, with the domain ID in the sixth column. Using wget, download the human proteome file. After downloading, run a single line of bash commands to determine the number of domain types (unique domains) in the human genome. You may use as many commands as you want, connected together in pipes.

```
wget https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/9606.tsv.gz
```

```
zcat 9606.tsv.gz | cut -f 6 | grep -v "#" | less | sort | uniq | wc -l
```

6680

Problem 2

Download the covid19 fasta file from

https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam_SARS-CoV-2_2.0/ using wget command.

1. How many proteins are there?
2. How many amino acids are there?
3. Take only the protein sequences, and convert them to lower case.

Problem 2

Download the covid19 fasta file from

https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam_SARS-CoV-2_2.0/ using wget command.

1. How many proteins are there?
2. How many amino acids are there?
3. Take only the protein sequences, and convert them to lower case.

```
cat covid-19.fasta | grep ">" | wc -l
```

Problem 2

Download the covid19 fasta file from

https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam_SARS-CoV-2_2.0/ using wget command.

1. How many proteins are there?
2. How many amino acids are there?
3. Take only the protein sequences, and convert them to lower case.

```
cat covid-19.fasta | grep ">" | wc -l
```

```
cat covid-19.fasta | grep -v ">" | wc -c
```

Problem 2

Download the covid19 fasta file from

https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam_SARS-CoV-2_2.0/ using wget command.

1. How many proteins are there?
2. How many amino acids are there?
3. Take only the protein sequences, and convert them to lower case.

```
cat covid-19.fasta | grep ">" | wc -l
```

```
cat covid-19.fasta | grep -v ">" | wc -c
```

```
cat covid-19.fasta | grep -v ">" | tr [:upper:] [:lower:]
```

Problem 3

PFAM is a domain database. It also includes domains that have already been computed for all proteomes. The most recent version is available here:

<https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/>. Each file represents a proteome, which is recognized by its taxonomy ID. The ID 9606 belongs to a human. Each of these files is tab-delimited, with the domain ID in the sixth column.

1. Make a directory in your home folder software/bin.
2. Change directory to software/bin.
3. Write a bash script that take an input: taxonomy ID.
4. If file does not exist, Script should download the proteome file for the input taxonomy ID.
5. Script should run a single line of bash commands to determine the number of domain types (unique domains) in the genome.
6. Run step 5 **only if** step 4 is finished.
7. Save your file as count_uniq_domains
8. Add your script to PATH
9. Go to your home folder.
10. Create a bash script that will run count_uniq_domains (without specifying path). This script should store STDOUT and STDERR.

Problem 3

```
# 1 and 2
```

```
cd ~/  
mkdir software  
cd software  
mkdir bin
```

```
# 3, 4, 5, 6, and 7
```

```
nano count_uniq_domains
```

```
#!/bin/bash
```

```
txid=$1  
txid2=${txid}_final.tsv.gz  
  
if [ -e $txid2 ]  
  then  
    zcat $txid2 | grep -v "#" | cut -f 6 | sort| uniq| wc -l  
  else  
    wget https://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam35.0/proteomes/$txid.tsv.gz &&\br/>    mv $txid.tsv.gz $txid2 &&\br/>    zcat $txid2 | grep -v "#" | cut -f 6 | sort| uniq| wc -l  
fi
```

Problem 3

```
# 8
chmod +x count_uniq_domains
export PATH=~./software/bin/:$PATH

# 9
cd ~/

# 10
nano run.sh
#!/bin/bash
count_uniq_domains 9696 > STDOUT 2>STDERR
## close the file with ctrl + x
bash run.sh
```

Resources

https://github.com/urppeia/BIO609_2016

https://github.com/urppeia/BIO609_2017

https://github.com/urppeia/BIO609_2018

<https://github.com/urppeia/bio609>

https://github.com/urppeia/BIO609_2021

Instructor and Course evaluation



bit.ly/bio609CE

**Thank you for your
attention!**

bit.ly/bio6092022