

Introduction to UNIX/Linux or: How I learned to Stop Worrying and Love the Command Line

Presentation by Eftyhios Kirbizakis



What is an operating system (OS)?

What is an operating system?



You

What is the definition of an operating system?

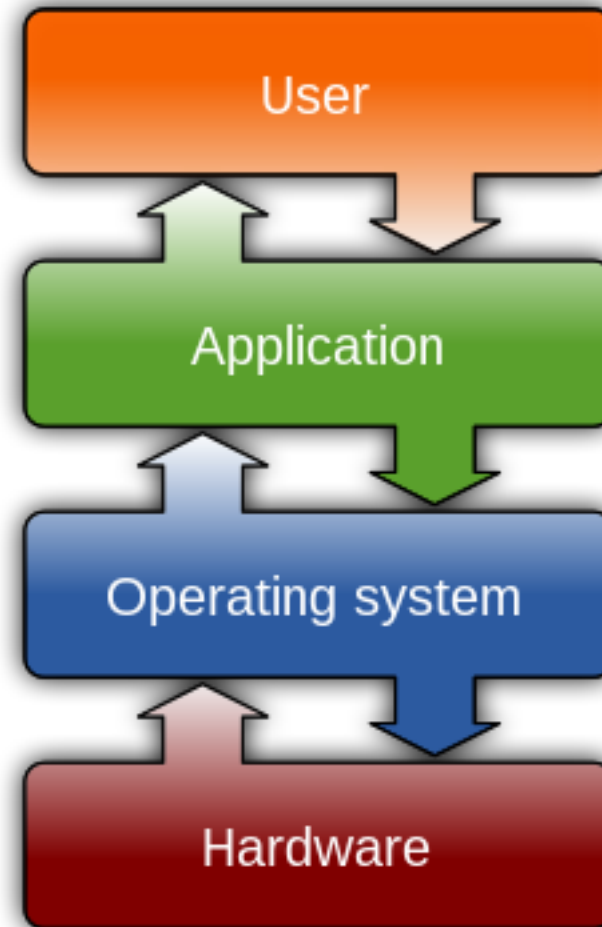


ChatGPT

An operating system (OS) is a software system that manages computer hardware and provides a platform for running applications. It acts as an intermediary between the user and the computer hardware, facilitating communication and coordination between various system components.

What is an operating system?

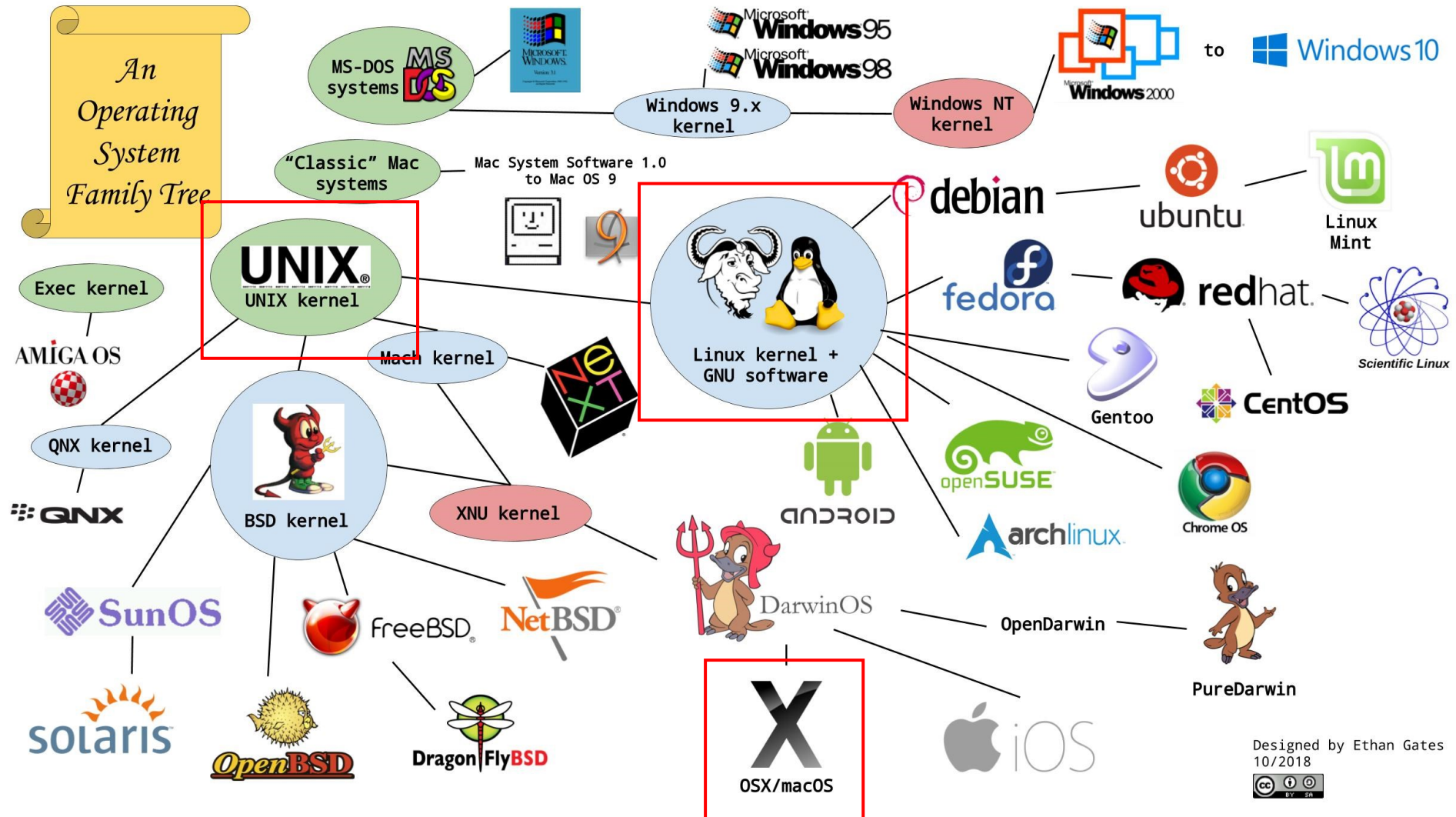
- Abstraction layer between the computer hardware and software
- Allows the developers to write software that runs on an operating system without being concerned about hardware-specific details



What is UNIX/Linux

- Both are OS
- UNIX is a commercial product developed in 1969 at the AT&T Bell Labs
- Linux is freeware developed in 1991 by Linus Torvalds
- No major differences

What is UNIX/Linux



UNIX/Linux philosophy

Write programs that do one thing and do it well.

Write programs to work together.

Write programs to handle text streams, because that is a universal interface.

-Doug McIlroy, summarized by Peter H. Salus

Why UNIX/Linux ?

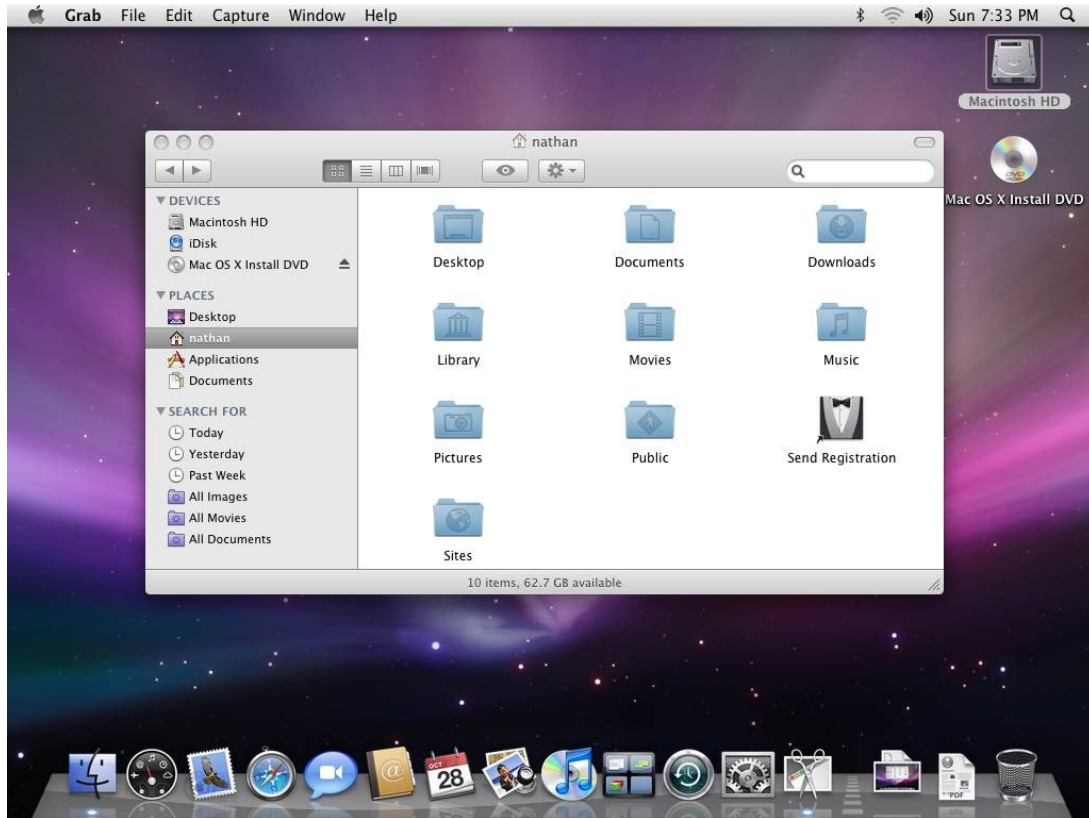
- Stability
- Multitasking
- Flexibility
- Science focus – free
 - Barrier to creating is lower
- Data storage: Handles lots of files well
- Historical reasons: Early software written on UNIX

Why UNIX/Linux ?

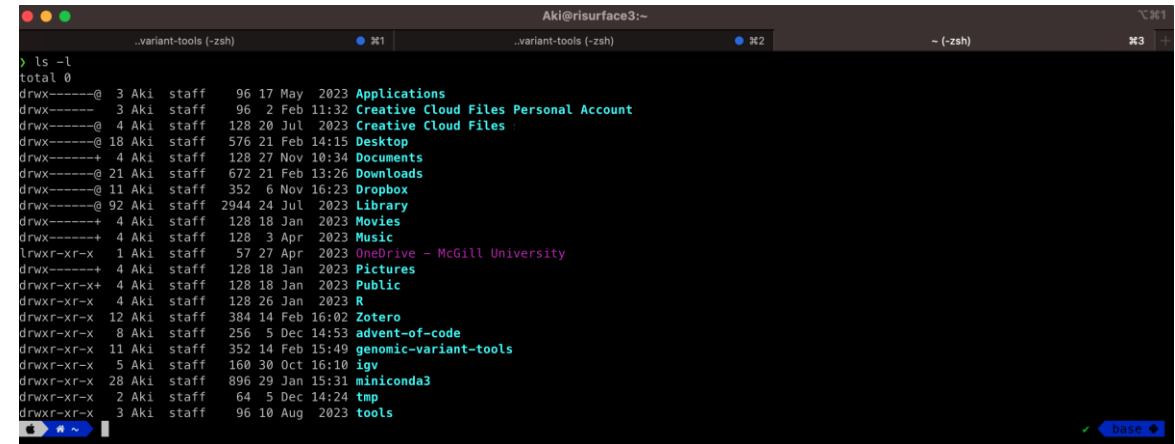
- Stability
- Multitasking
- Flexibility
- Science focus – free
 - Barrier to creating is lower
- Data storage: Handles lots of files well
- Historical reasons: Early software written on UNIX

Unix/Linux systems have become the de facto standard in bioinformatics and genomic data analysis due to their flexibility, scalability, and robust command-line interface.

Graphical User Interface

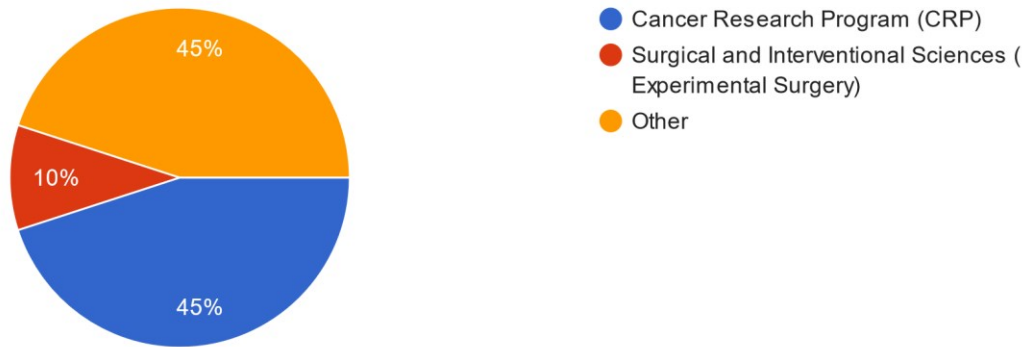


Command line interface (CLI)



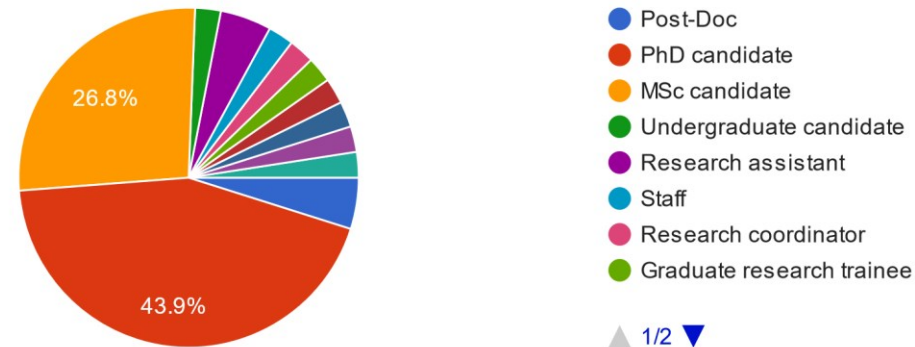
Choose your program

40 responses



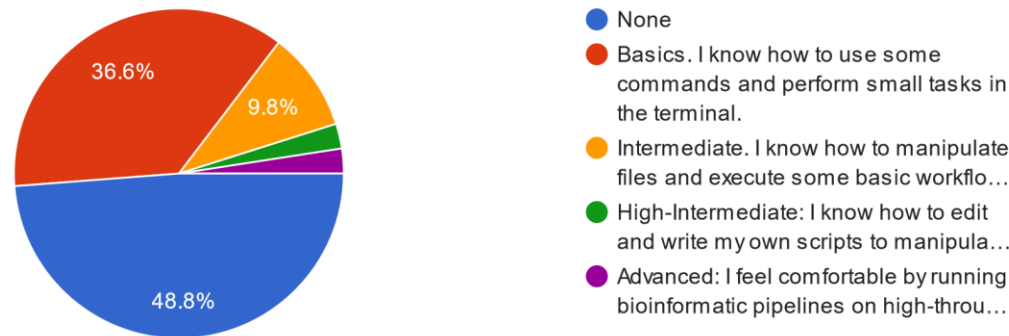
Student status

41 responses



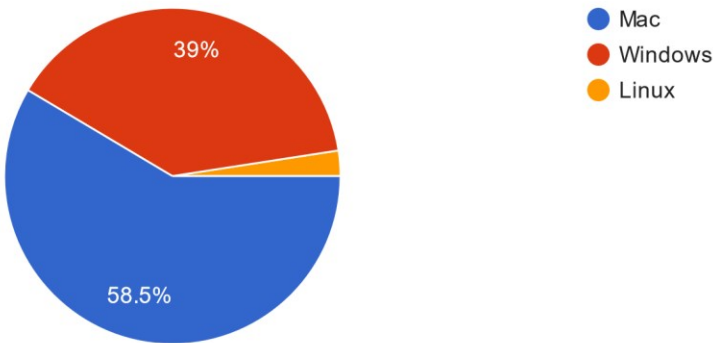
What is your experience level in Unix/Linux programming language?

41 responses



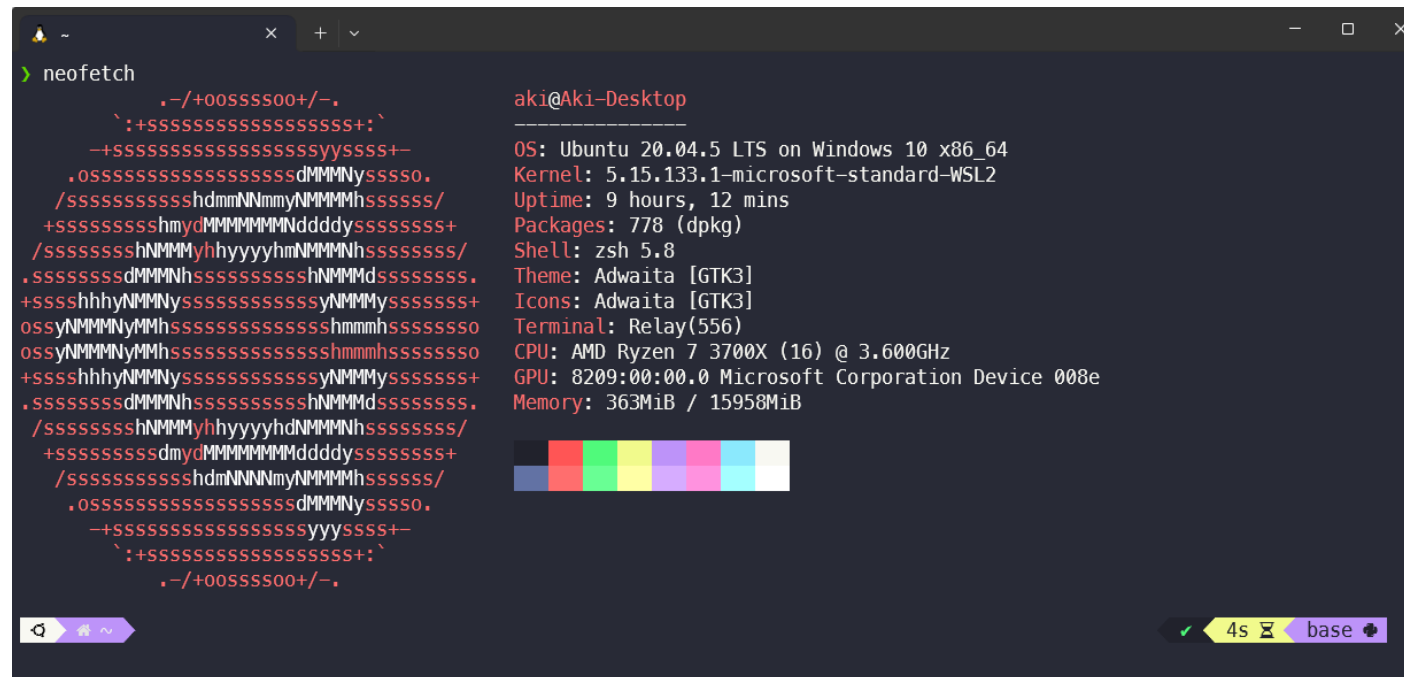
Could you please indicate what is your operative system software?

41 responses



How to proceed

- OSX/Linux users: Good to go!
- Windows: Windows Subsystem for Linux (WSL) + Windows Terminal



The screenshot shows a Windows Terminal window with a dark theme. The title bar indicates the user is 'aki@Aki-Desktop'. The terminal prompt is '>' and the command 'neofetch' has been executed. The output of neofetch is a ASCII art logo of a cat, followed by system information: OS: Ubuntu 20.04.5 LTS on Windows 10 x86_64, Kernel: 5.15.133.1-microsoft-standard-WSL2, Uptime: 9 hours, 12 mins, Packages: 778 (dpkg), Shell: zsh 5.8, Theme: Adwaita [GTK3], Icons: Adwaita [GTK3], Terminal: Relay(556), CPU: AMD Ryzen 7 3700X (16) @ 3.600GHz, GPU: 8209:00:00.0 Microsoft Corporation Device 008e, Memory: 363MiB / 15958MiB. A color calibration bar is visible below the system information. The terminal window has a status bar at the bottom showing a green checkmark, '4s', and 'base'.

```
> neofetch

      .-/+00SSSS00+/- .
      `:+SSSSSSSSSSSSSSSS+:`
      --SSSSSSSSSSSSSSSSyySSSS+-
      .oSSSSSSSSSSSSSSSSdMMMNySSSSo.
      /SSSSSSSSShdmmNNmmyNMMMMhSSSSS/
      +SSSSSSSSshmyoMMMMMMNdddySSSSSSS+
      /SSSSSSShNMMMyhhyyyhmNMMMNhSSSSSSS/
      .SSSSSSSdMMMNhSSSSSSShNMMMdSSSSSSS.
      +SSSShhhyNMMNySSSSSSSSSyNMMMySSSSSS+
      ossyNMMMNyMMhSSSSSSSSSSShmmhSSSSSSo
      ossyNMMMNyMMhSSSSSSSSSSShmmhSSSSSSo
      +SSSShhhyNMMNySSSSSSSSSyNMMMySSSSSS+
      .SSSSSSSdMMMNhSSSSSSShNMMMdSSSSSSS.
      /SSSSSSShNMMMyhhyyyhdNMMMNhSSSSSSS/
      +SSSSSSSdmyoMMMMMMNdddySSSSSSS+
      /SSSSSSSSShdmmNNmmyNMMMMhSSSSS/
      .oSSSSSSSSSSSSSSSSdMMMNySSSSo.
      --SSSSSSSSSSSSSSSSyySSSS+-
      `:+SSSSSSSSSSSSSSSS+:`
      .-/+00SSSS00+/- .

aki@Aki-Desktop
-----
OS: Ubuntu 20.04.5 LTS on Windows 10 x86_64
Kernel: 5.15.133.1-microsoft-standard-WSL2
Uptime: 9 hours, 12 mins
Packages: 778 (dpkg)
Shell: zsh 5.8
Theme: Adwaita [GTK3]
Icons: Adwaita [GTK3]
Terminal: Relay(556)
CPU: AMD Ryzen 7 3700X (16) @ 3.600GHz
GPU: 8209:00:00.0 Microsoft Corporation Device 008e
Memory: 363MiB / 15958MiB
```

Realistic bioinformatics

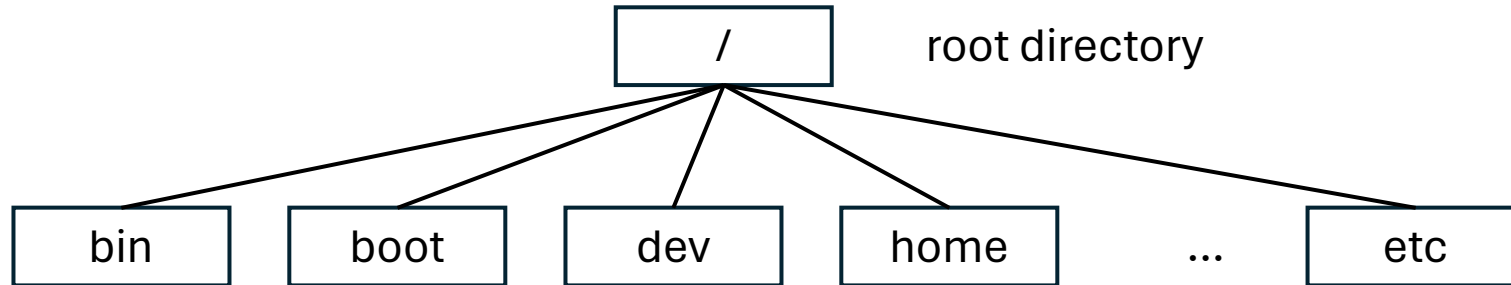
- Your computer will not be enough for serious bioinformatics
- Remote server/cluster
 - Compute Canada
 - Private servers
 - Amazon web services (AWS)

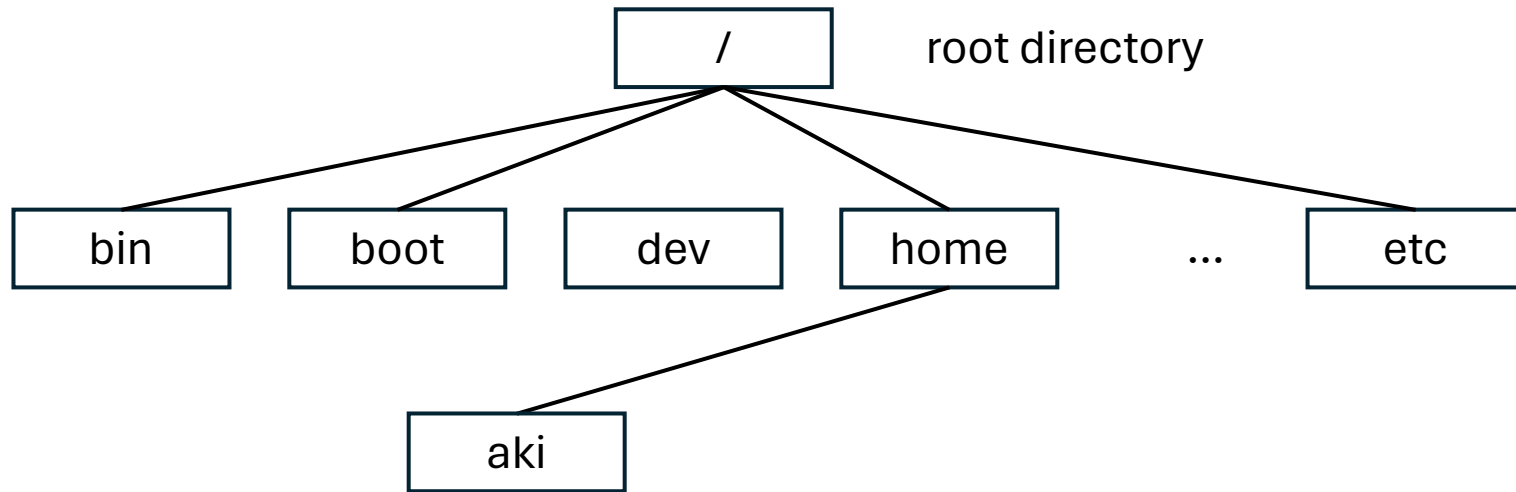
Unix file system

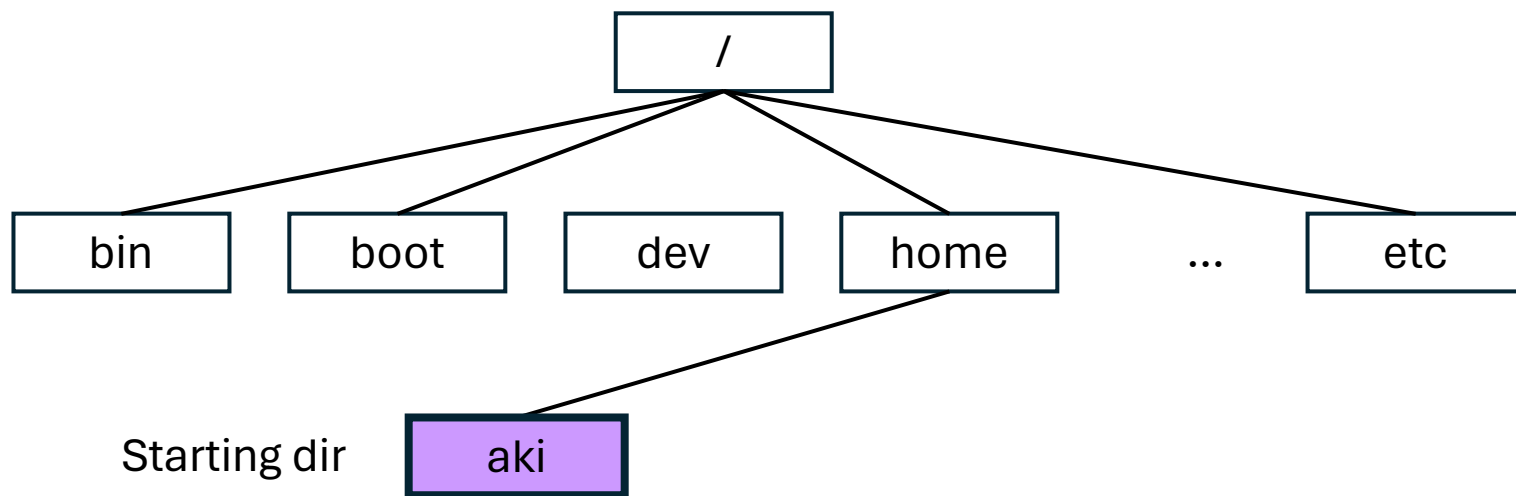
/

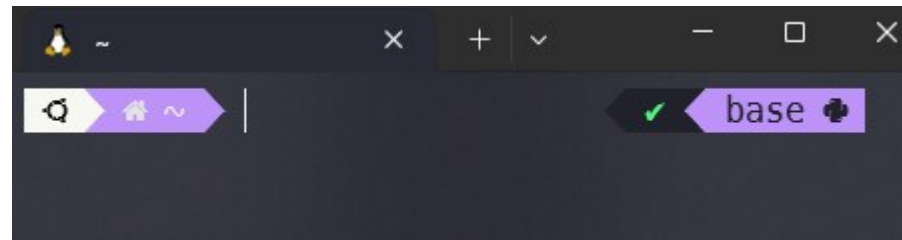
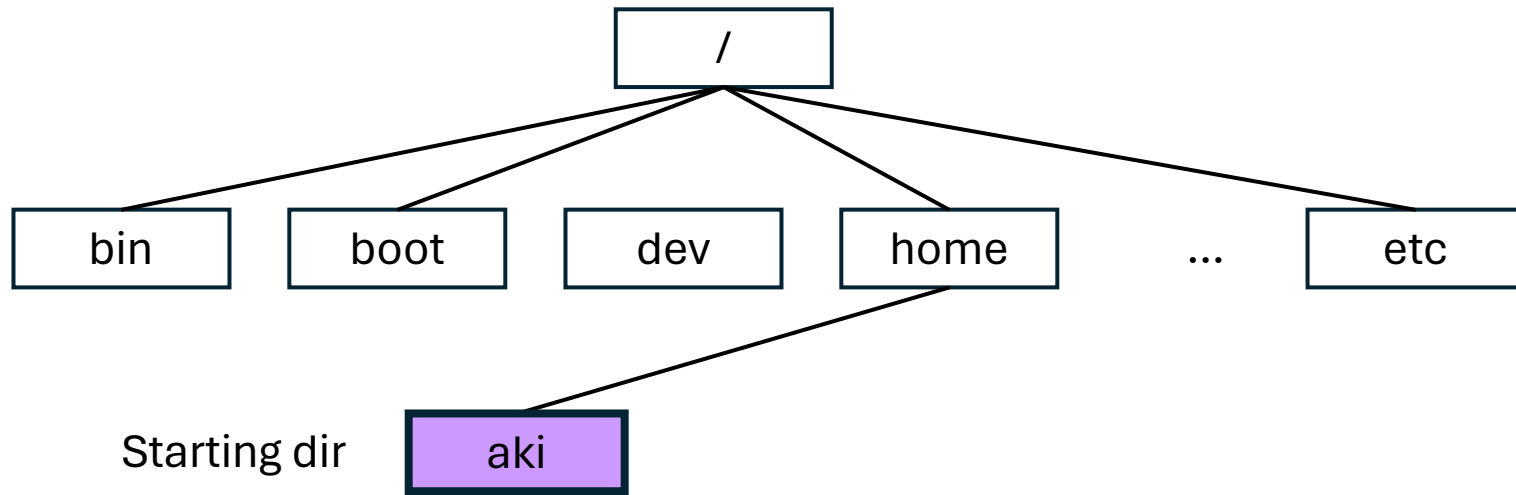
root directory

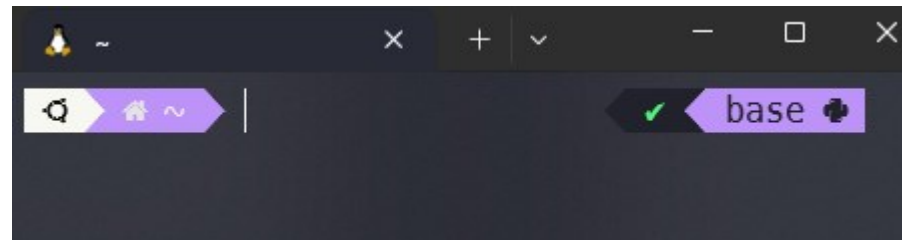
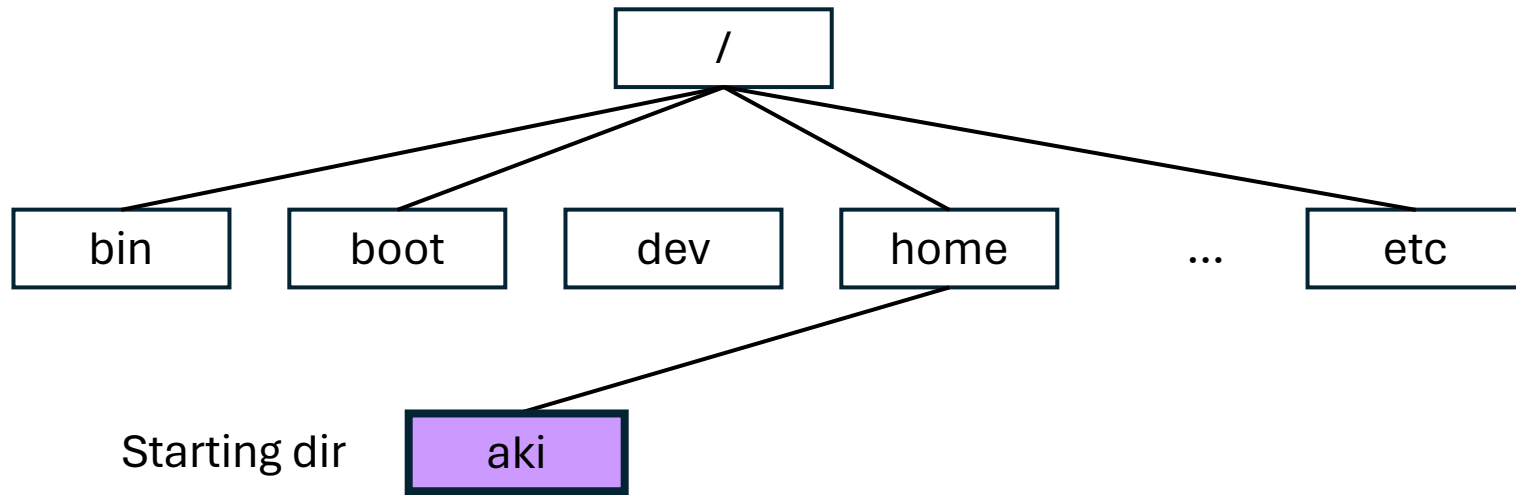
Unix file system: Upside down tree



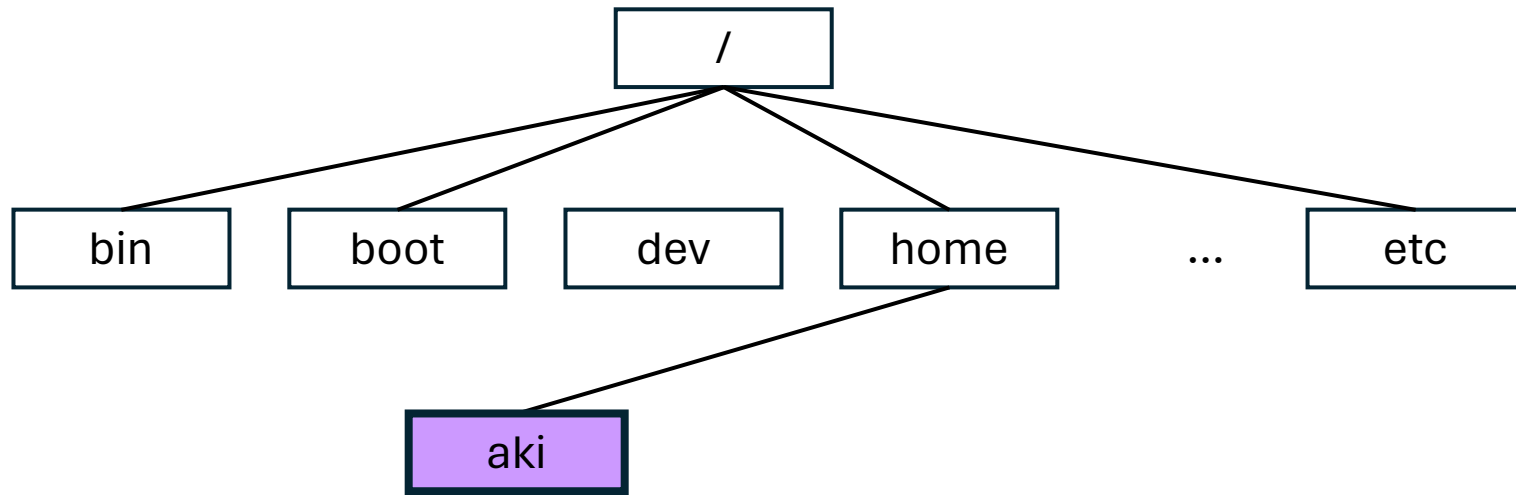




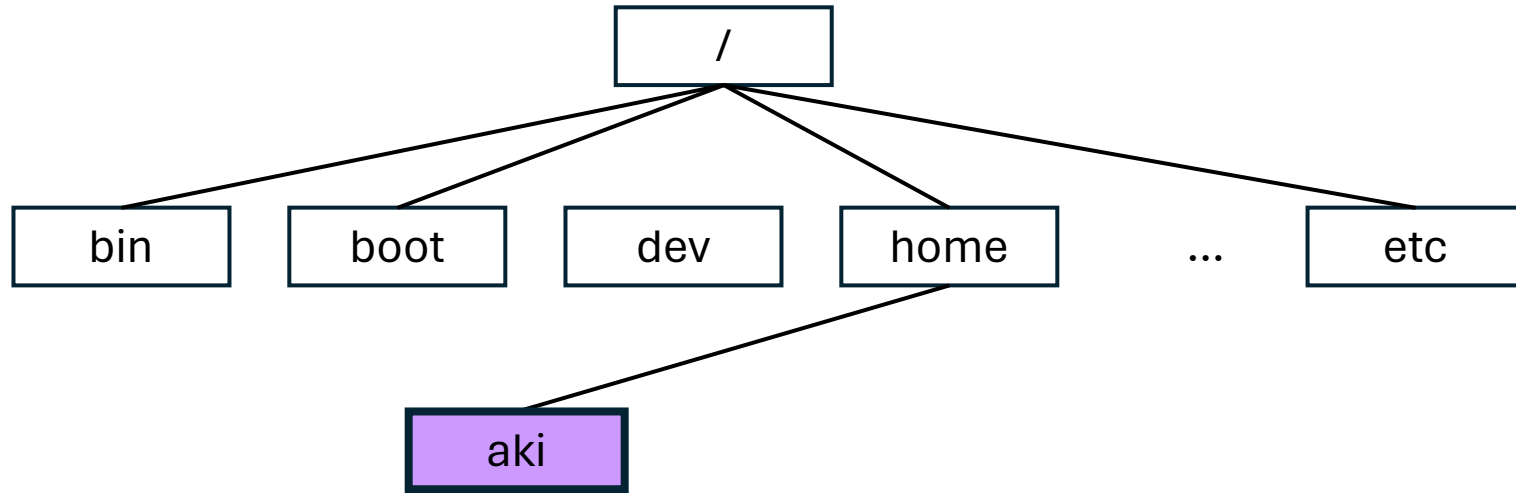




Tilde (~): Shortcut denoting user's home directory

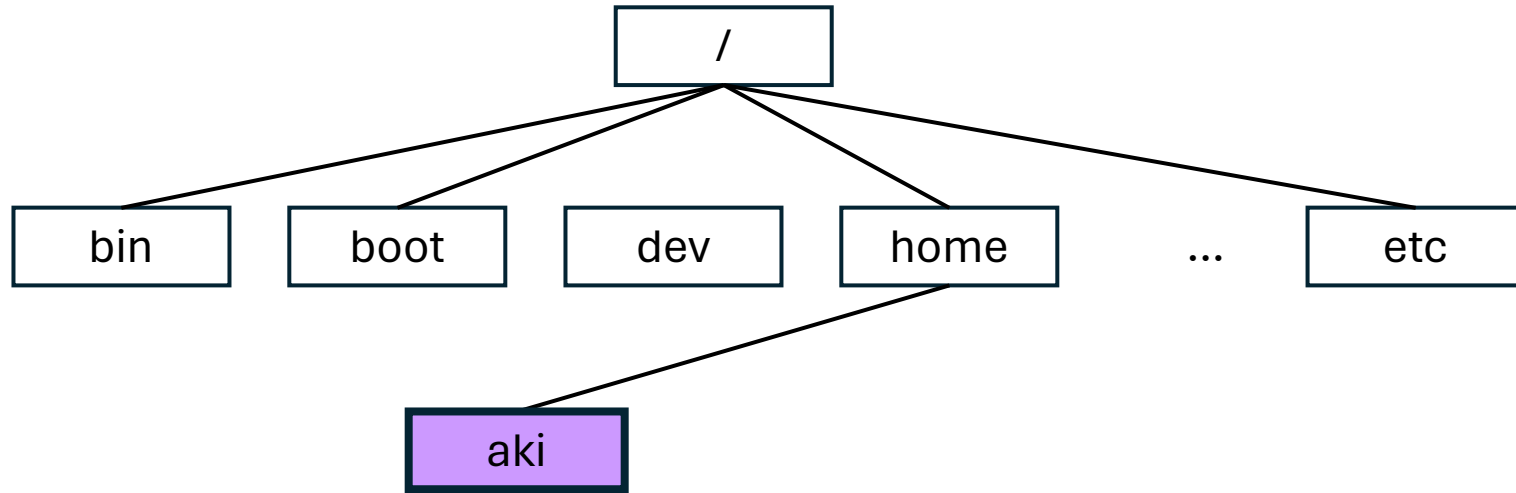


/ is a path separator



/ is a path separator

What is the path to my user folder?



/ is a path separator

What is the path to my user folder?

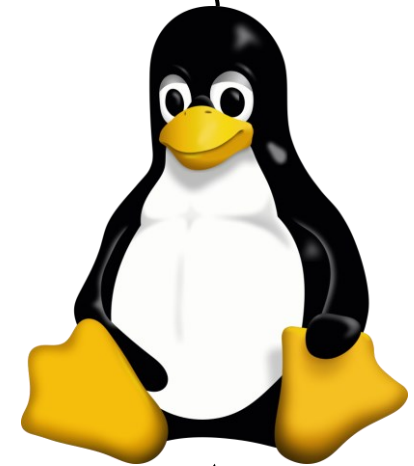
/home/aki

UNIX shell

4. Executes command

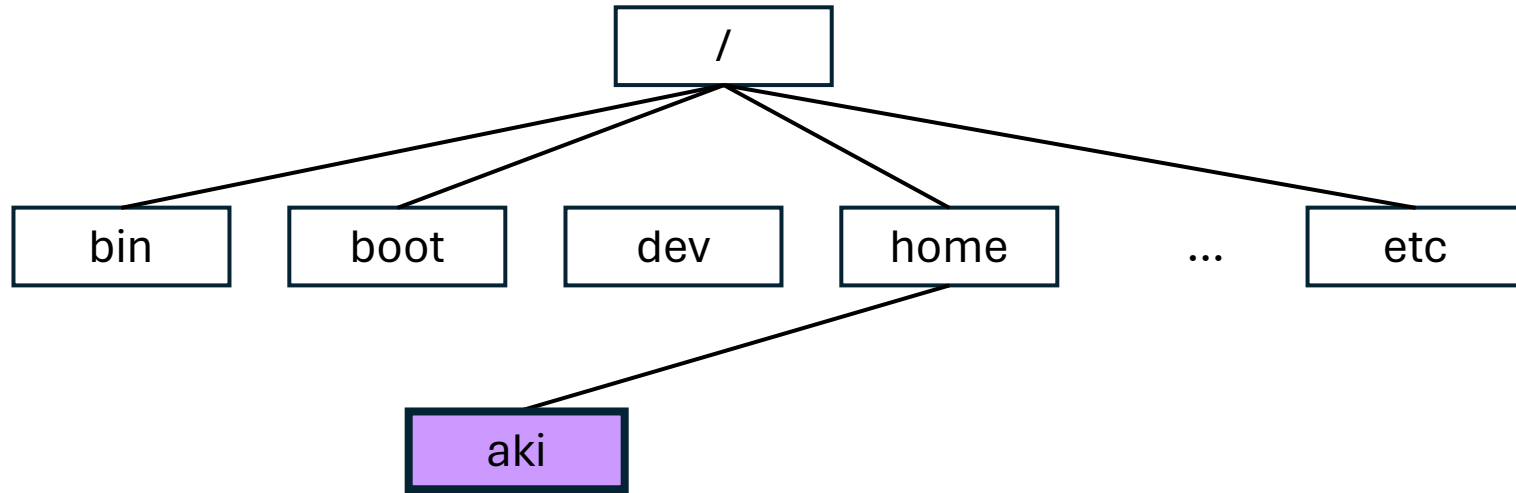
1. Display prompt
\$

User → 2. Enters command



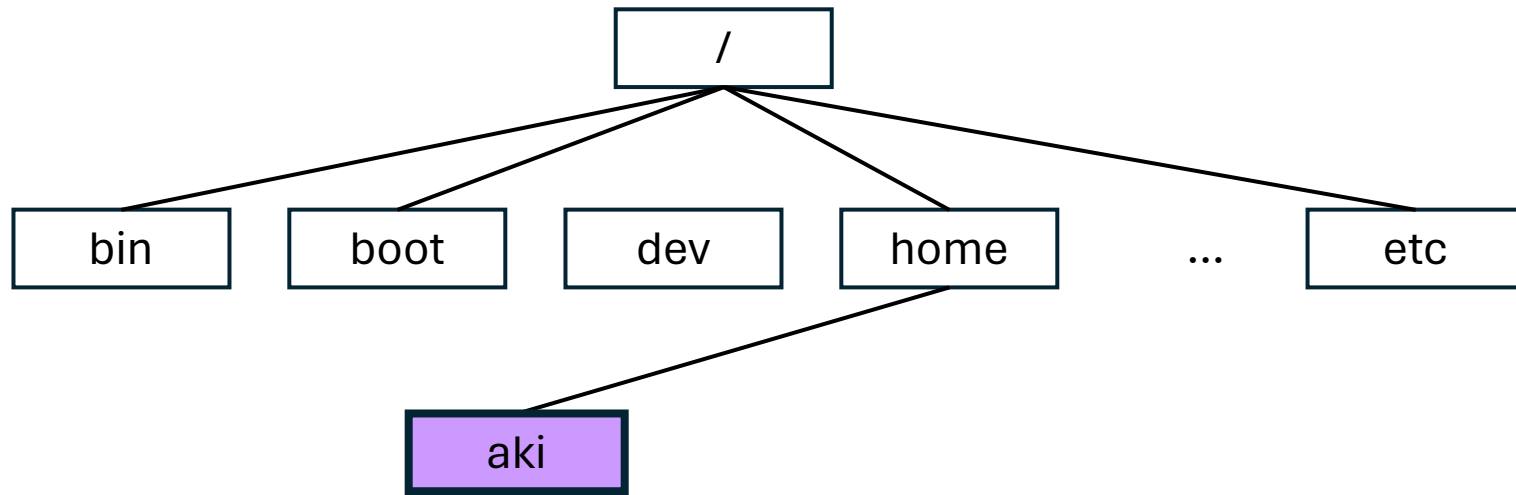
3. Interprets command

\$ command
output



```
$ pwd
```

Print work directory



```
$ pwd  
/home/aki
```

Basic command structure

```
$ ls -l folder
```

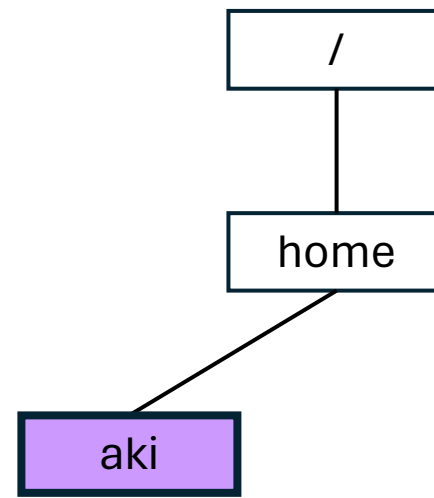
- Prompt
 - Shell is ready to receive command
- Command
 - Program being executed
- Options (or flags)
 - Configurations for the program
- Argument(s)
 - Target(s) of command

Basic command structure

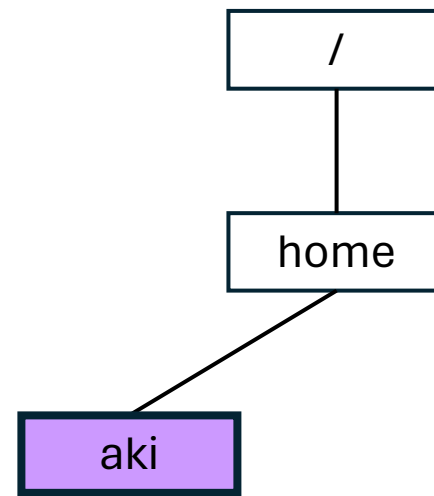
```
$ ls -l folder
```

- Prompt
 - Shell is ready to receive command
- Command
 - Program being executed
- Options (or flags)
 - Configurations for the program
- Argument(s)
 - Target(s) of command

Watch out for spelling and whitespace errors!



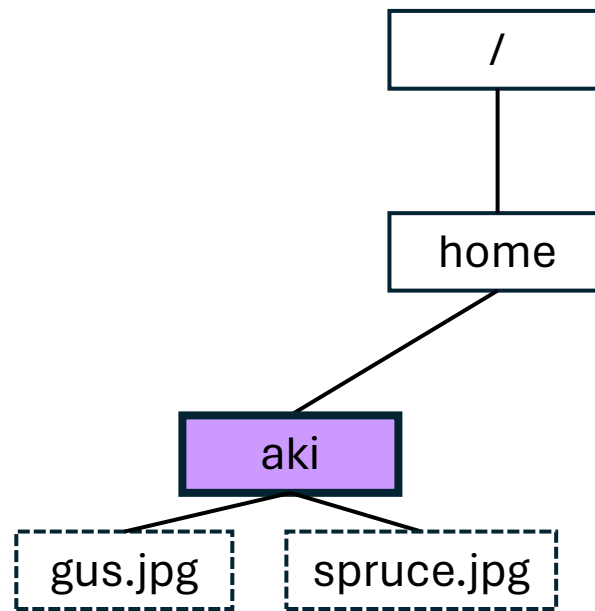
Current directory: `/home/aki`



`$ ls`

List directory contents

Current directory: `/home/aki`



```
$ ls  
gus.jpg spruce.jpg
```

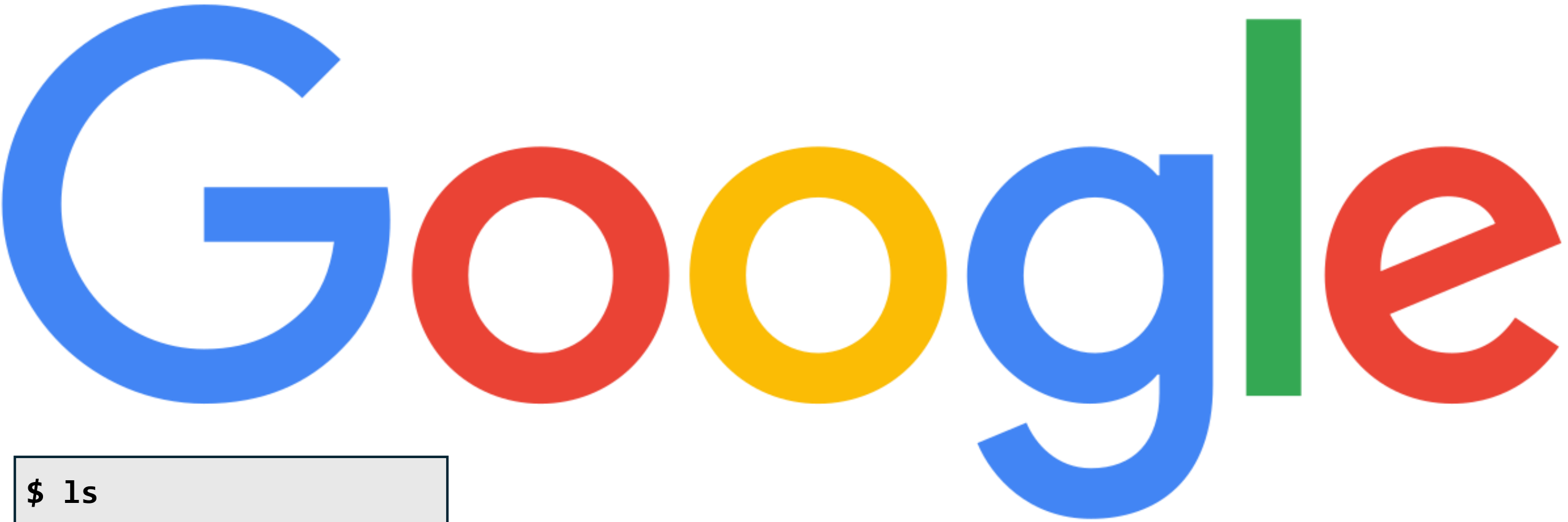
Current directory: /home/aki

How do we find out what a command does?

```
$ ls
```

```
Current directory: /home/aki
```

How do we find out what a command does?



```
$ ls
```

```
Current directory: /home/aki
```


How do we find out what a command does?

We read the manual

A light gray rectangular box with a dark gray border, representing a terminal window. It contains a dollar sign (\$) in the top-left corner.

\$

A light gray horizontal bar with a dark gray border, representing a terminal status bar. It contains the text "Current directory: /home/aki".

Current directory: /home/aki

How do we find out what a command does?

We read the manual

```
$ man ls
```

```
Current directory: /home/aki
```

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with .

-A, --almost-all

do not list implied . and ..

--author

LS(1)

User Commands

LS(1)

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with .

-A, --almost-all

do not list implied . and ..

--author

Manual page ls(1) line 1 (press h for help or q to quit)

Flags allow us to pass options to commands

```
LS(1)                                User Commands                                LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default). Sort
    entries alphabetically if none of -cftuvSUX nor --sort is specified.

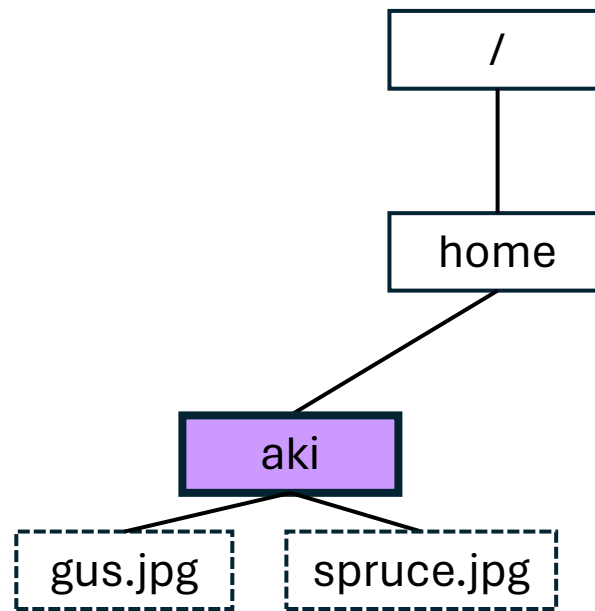
    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

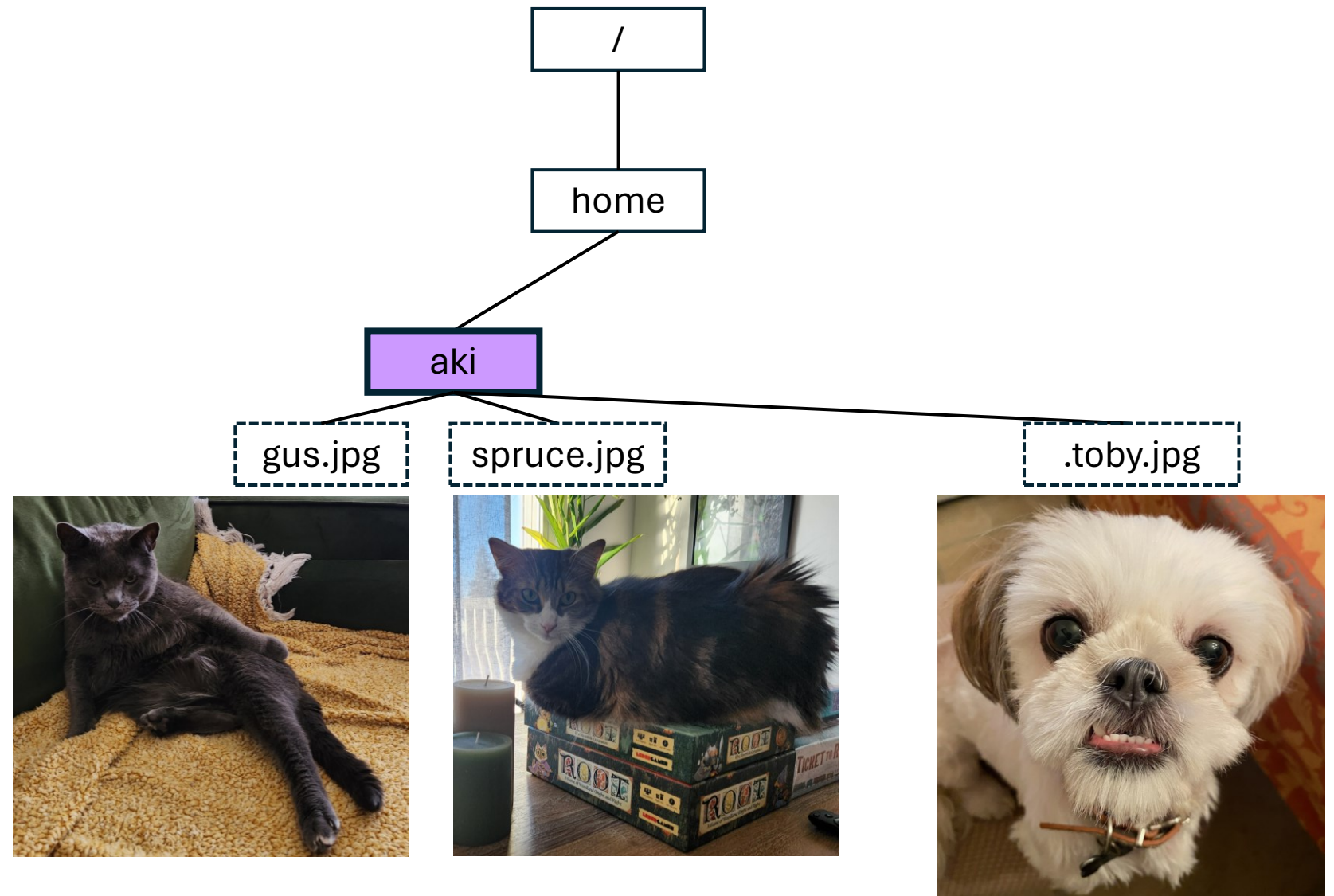
    --author

Manual page ls(1) line 1 (press h for help or q to quit)
```



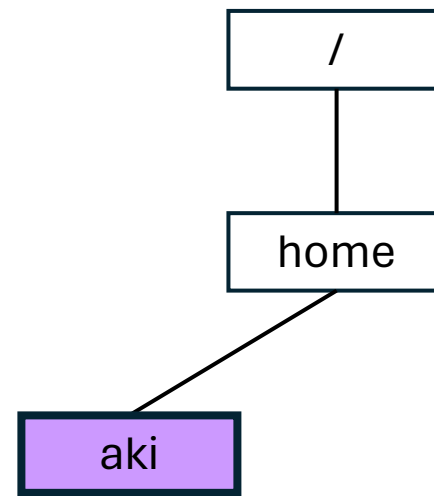
```
$ ls  
gus.jpg spruce.jpg
```

Current directory: /home/aki



```
$ ls -a
gus.jpg spruce.jpg
.toby.jpg
```

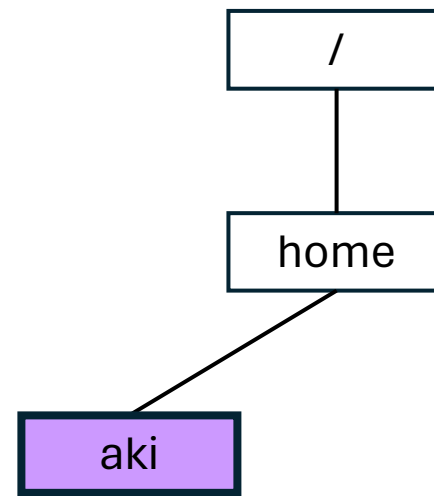
Current directory: `/home/aki`



```
$ mkdir
```

Make directory

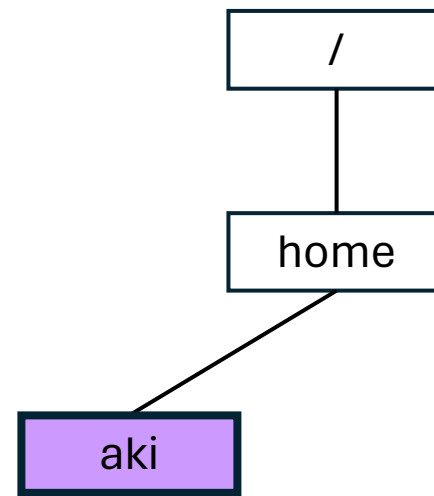
```
Current directory: /home/aki
```

```
$ mkdir  
mkdir: missing operand
```

Make directory

Current directory: /home/aki



```
$ man mkdir
```

Check the manual

Current directory: /home/aki

MKDIR(1)

User Commands

MKDIR(1)

NAME

mkdir - make directories

SYNOPSIS

mkdir [OPTION]... DIRECTORY...

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

-m, --mode=MODE

set file mode (as in chmod), not a=rwx - umask

-p, --parents

no error if existing, make parent directories as needed

-v, --verbose

print a message for each created directory

Manual page mkdir(1) line 1 (press h for help or q to quit)

No square brackets. Missing directory name

```
MKDIR(1)                                User Commands                                MKDIR(1)

NAME
    mkdir - make directories

SYNOPSIS
    mkdir [OPTION]... DIRECTORY...

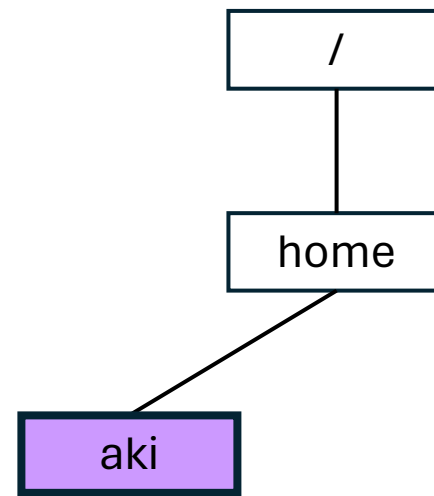
DESCRIPTION
    Create the DIRECTORY(ies), if they do not already exist.

    Mandatory arguments to long options are mandatory for short options too.

    -m, --mode=MODE
        set file mode (as in chmod), not a=rwx - umask

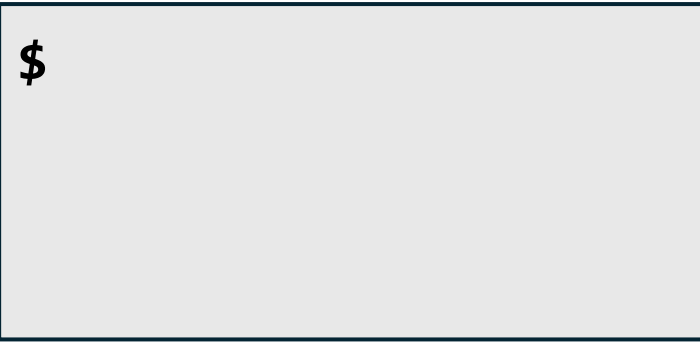
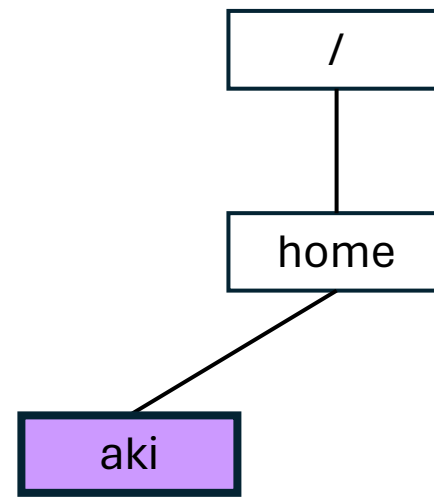
    -p, --parents
        no error if existing, make parent directories as needed

    -v, --verbose
        print a message for each created directory
Manual page mkdir(1) line 1 (press h for help or q to quit)
```

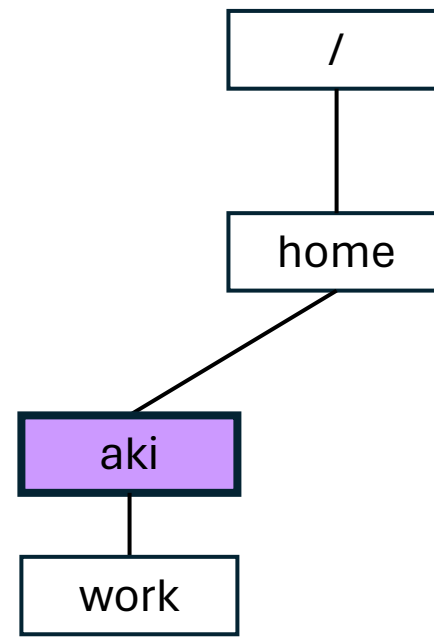


```
$ mkdir work
```

Current directory: /home/aki

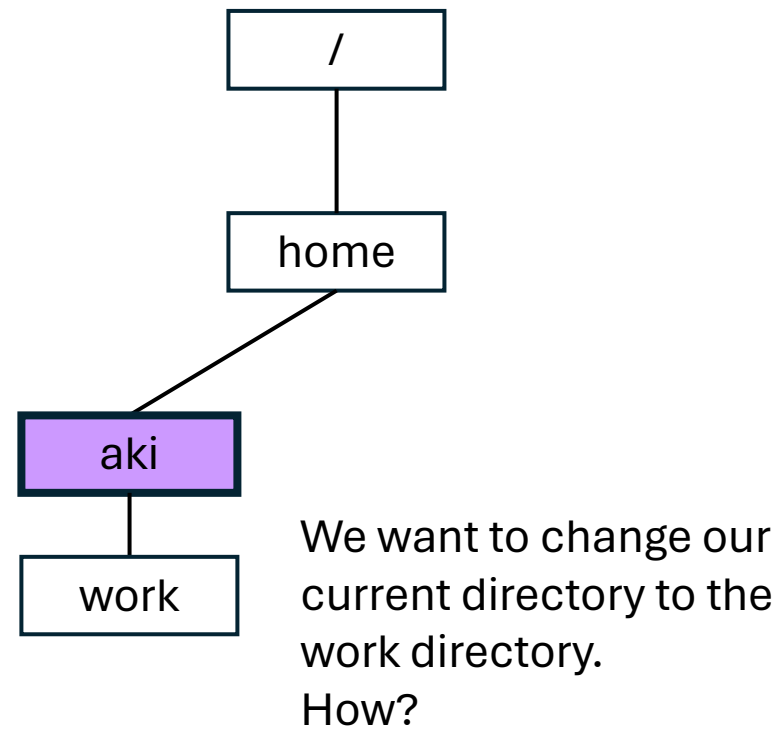


Current directory: `/home/aki`



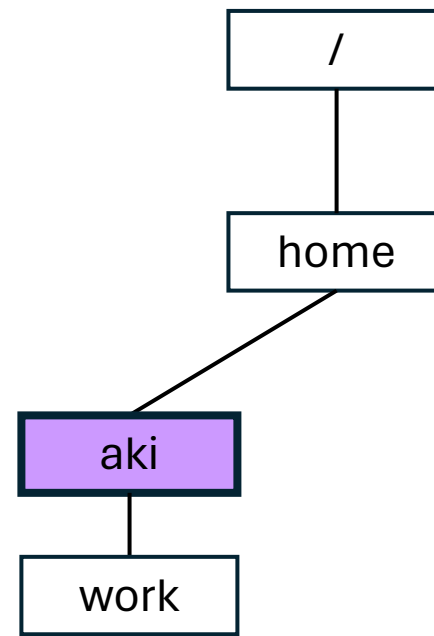
```
$ ls
work gus.jpg
spruce.jpg
```

Current directory: /home/aki



\$

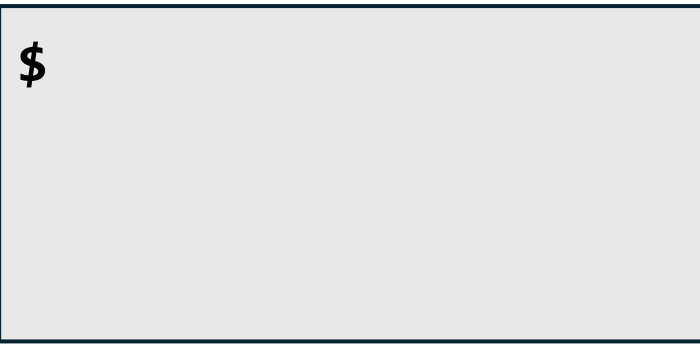
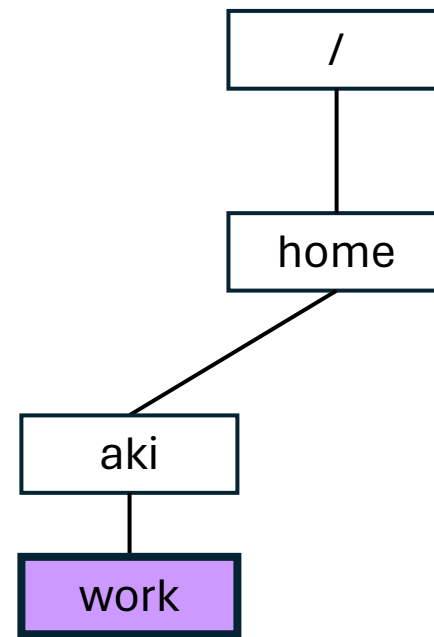
Current directory: `/home/aki`



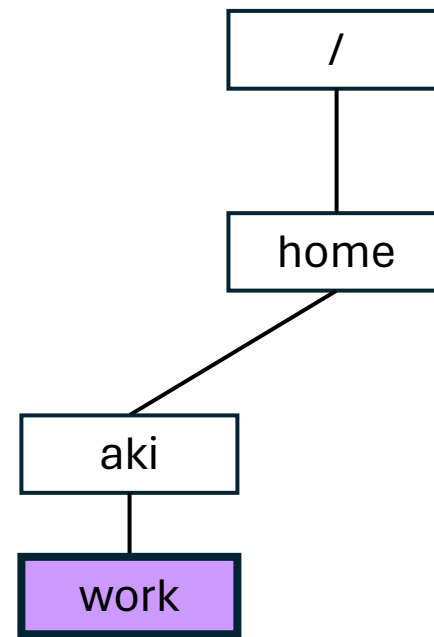
```
$ cd work
```

Change directory

```
Current directory: /home/aki
```



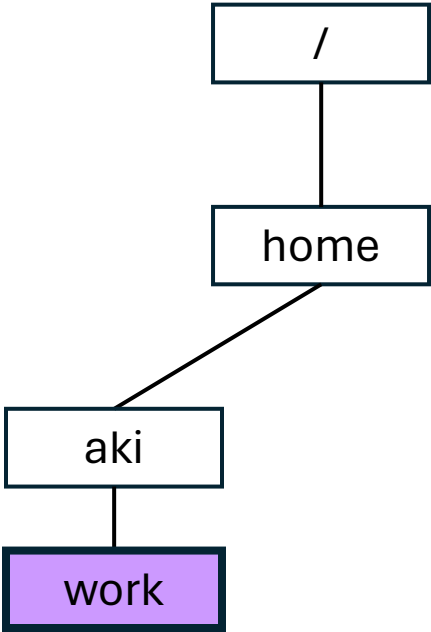
Current directory: `/home/aki/work`



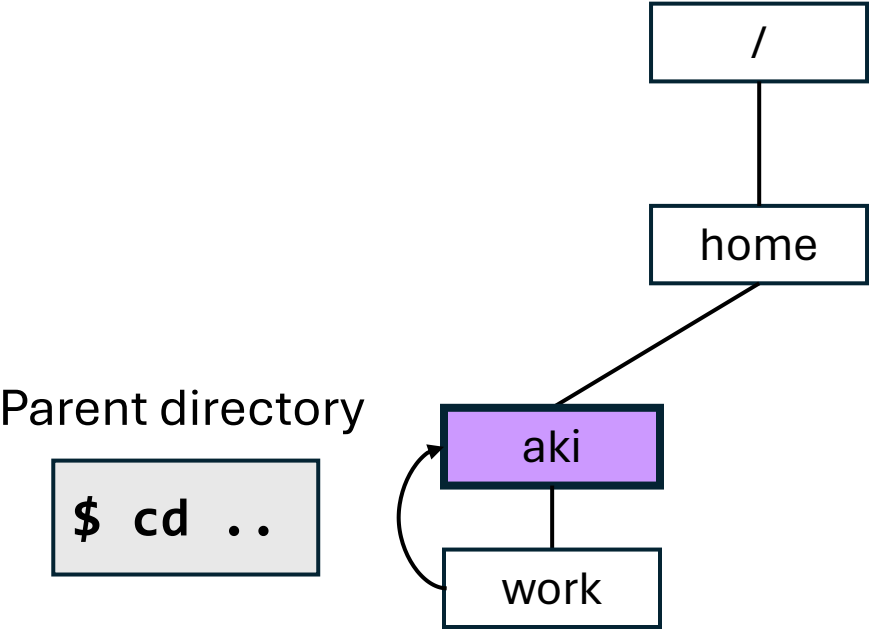
```
$ pwd  
/home/aki/work
```

Current directory: /home/aki/work

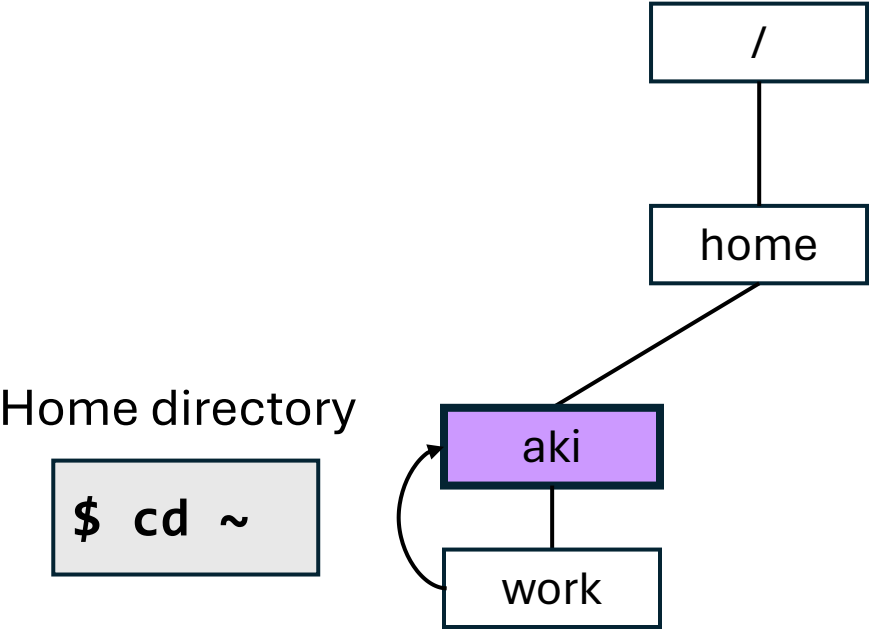
Navigating through directories



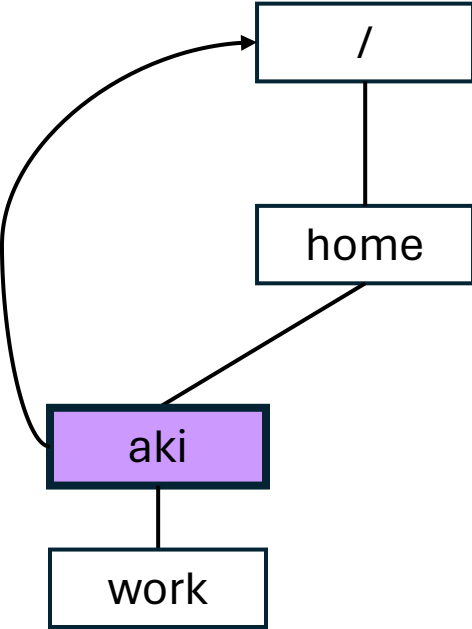
Navigating through directories



Navigating through directories



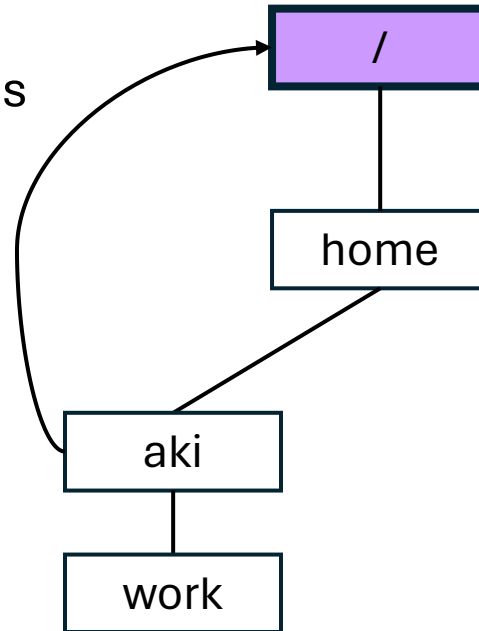
Navigating through directories



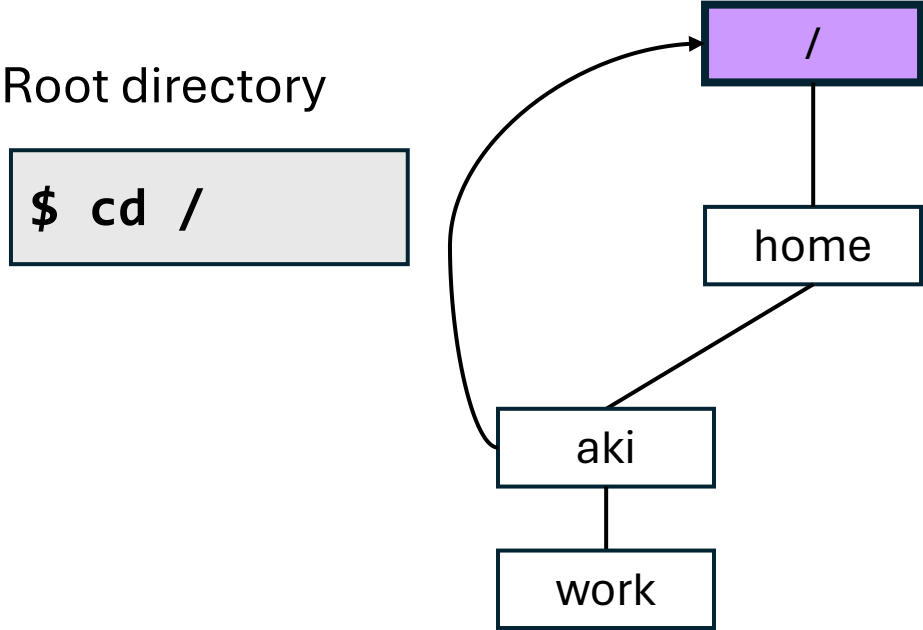
Navigating through directories

Up two parent directories

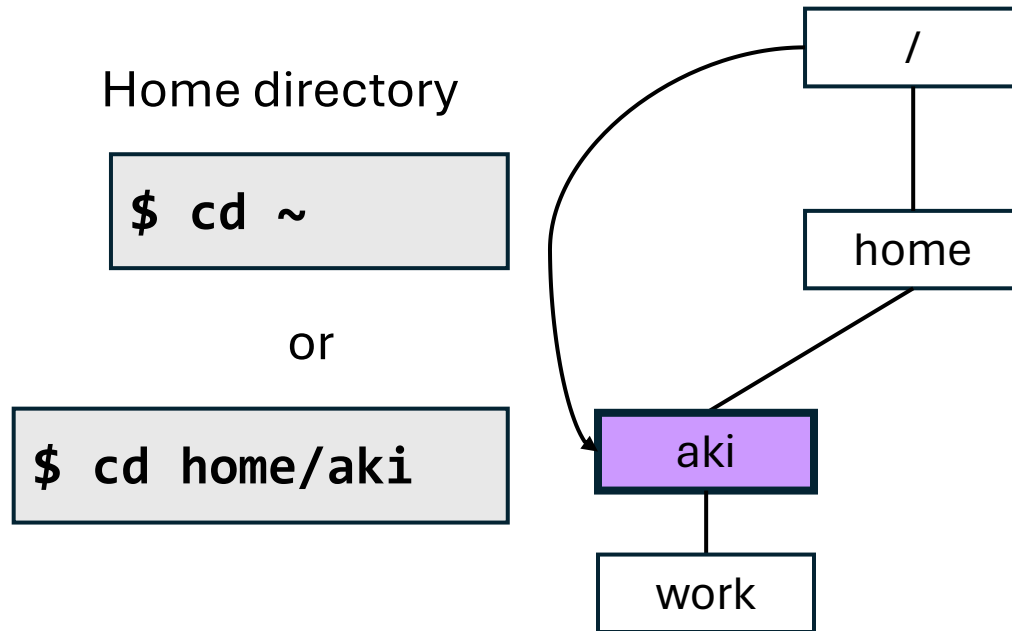
```
$ cd ../../
```



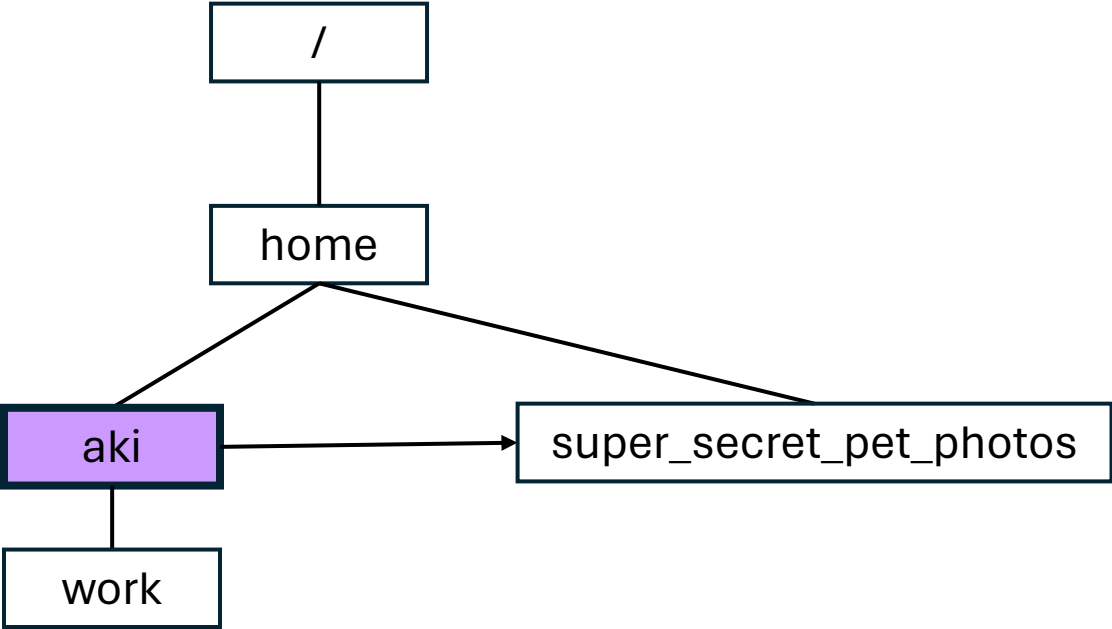
Navigating through directories



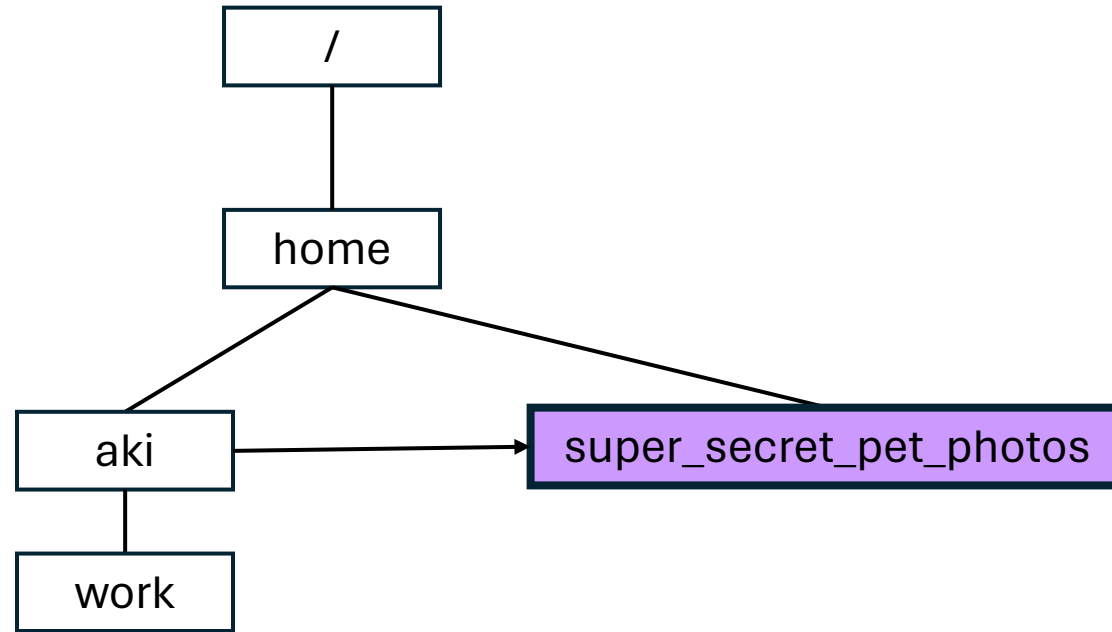
Navigating through directories



Navigating through directories



Navigating through directories



Relative path

```
$ cd ../super_secret_pet_photos
```

or

Absolute path

```
$ cd /home/super_secret_pet_photos
```



1



2.txt



3



1

Folder



2.txt

Text file



3

Binary file

File types

- All objects are files
- File names can be anything, however
 - Avoid whitespace
 - new file.txt looks like new\ file.txt -> Not pleasant to work with
 - Avoid special characters. Use – or _
 - Remember that files are case-sensitive
- Extensions do not matter (.txt, .tsv, etc)
 - Only for humans. Some software looks to perform function based on extension
 - Not enforced by the shell

File types

Generally, you will work with two file types:

1. Plain text files

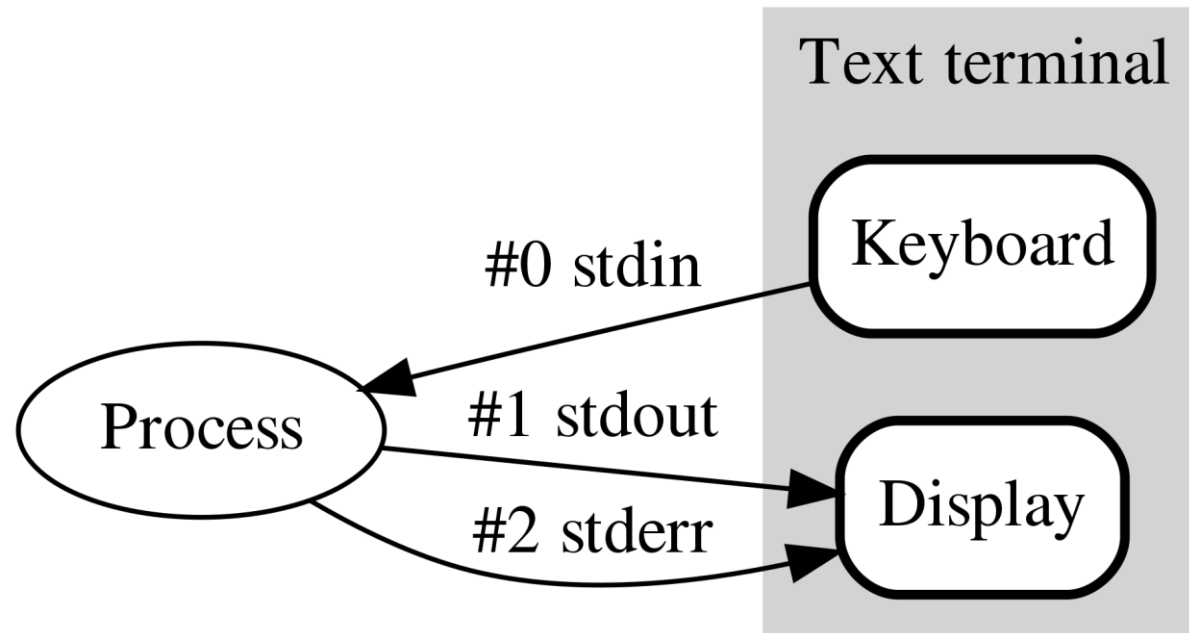
- Human readable
- Portable and can be read by many programs
- Take more space

2. Binary files

- Not human readable
- Read by specific programs
- More efficient with size

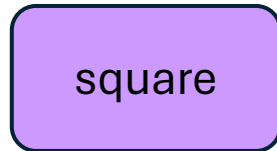
Input/output streams

- Allow the program to interact with its environment



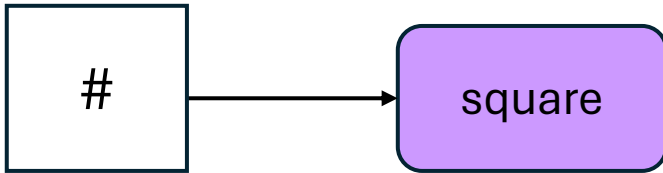
Input/output streams

- Allow the program to interact with its environment



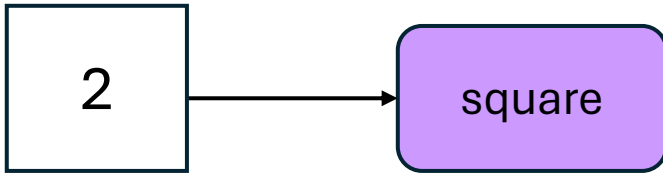
Input/output streams

stdin

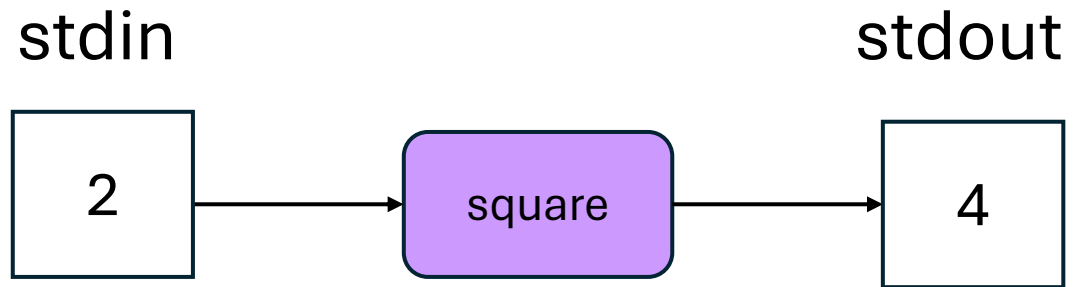


Input/output streams

stdin

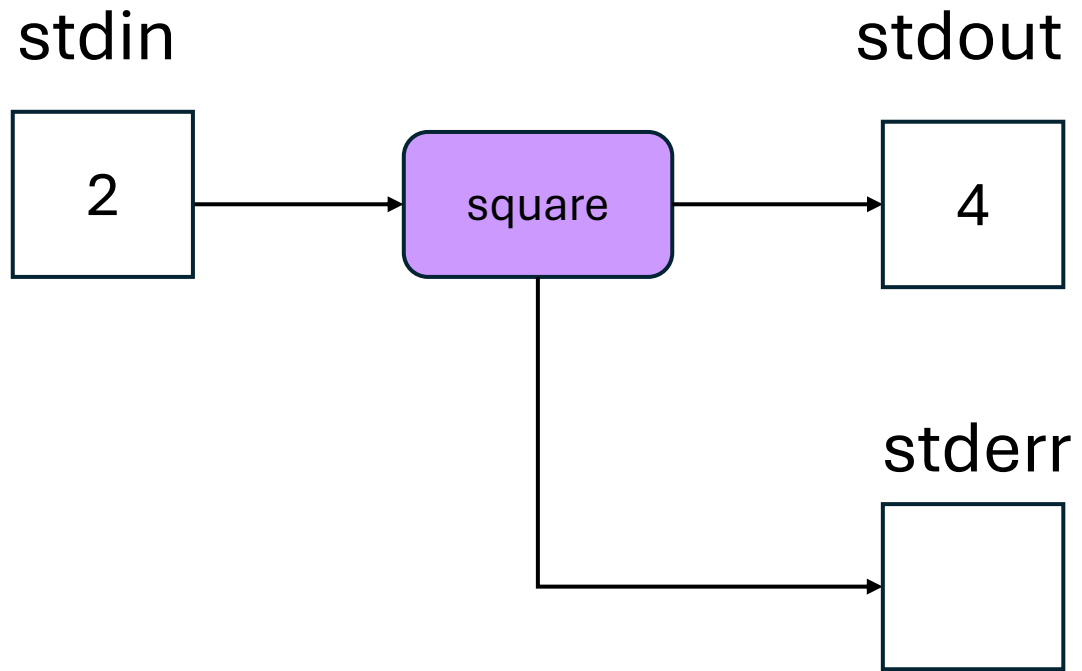


Input/output streams



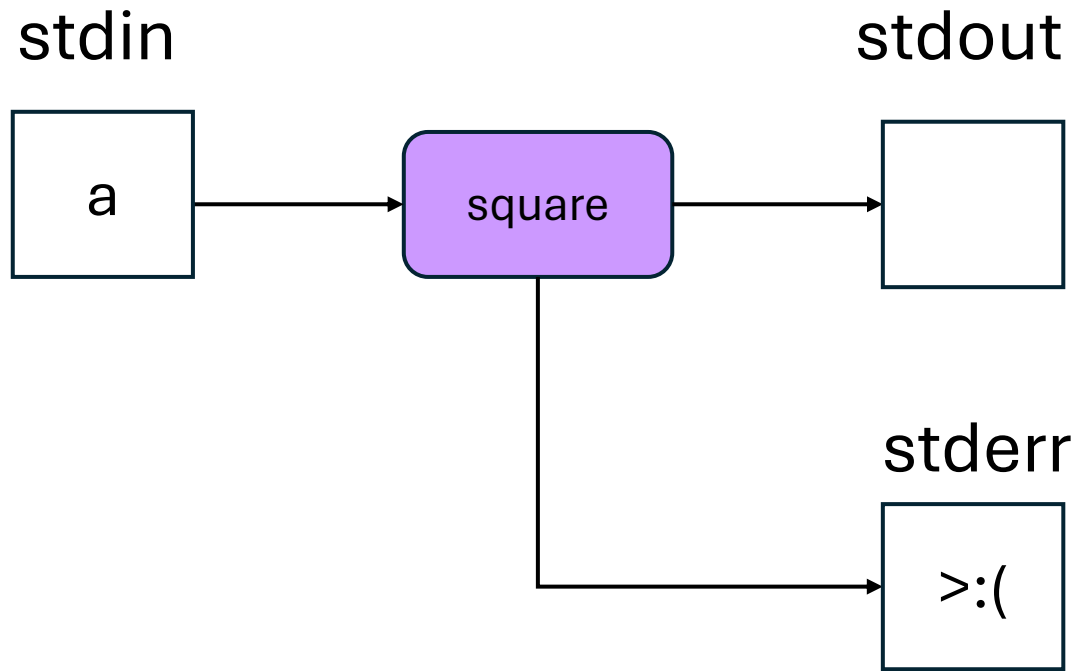
```
$ square 2
```

Input/output streams



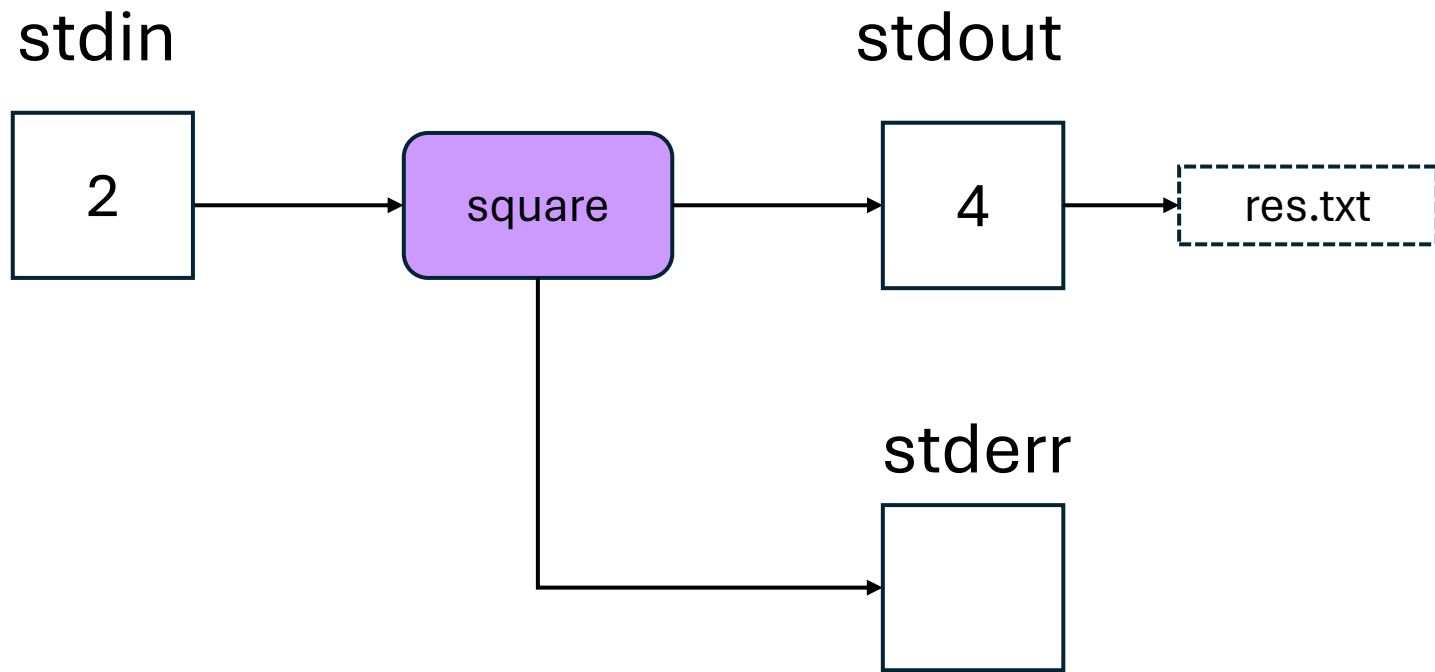
```
$ square 2
```

Input/output streams



```
$ square a
```

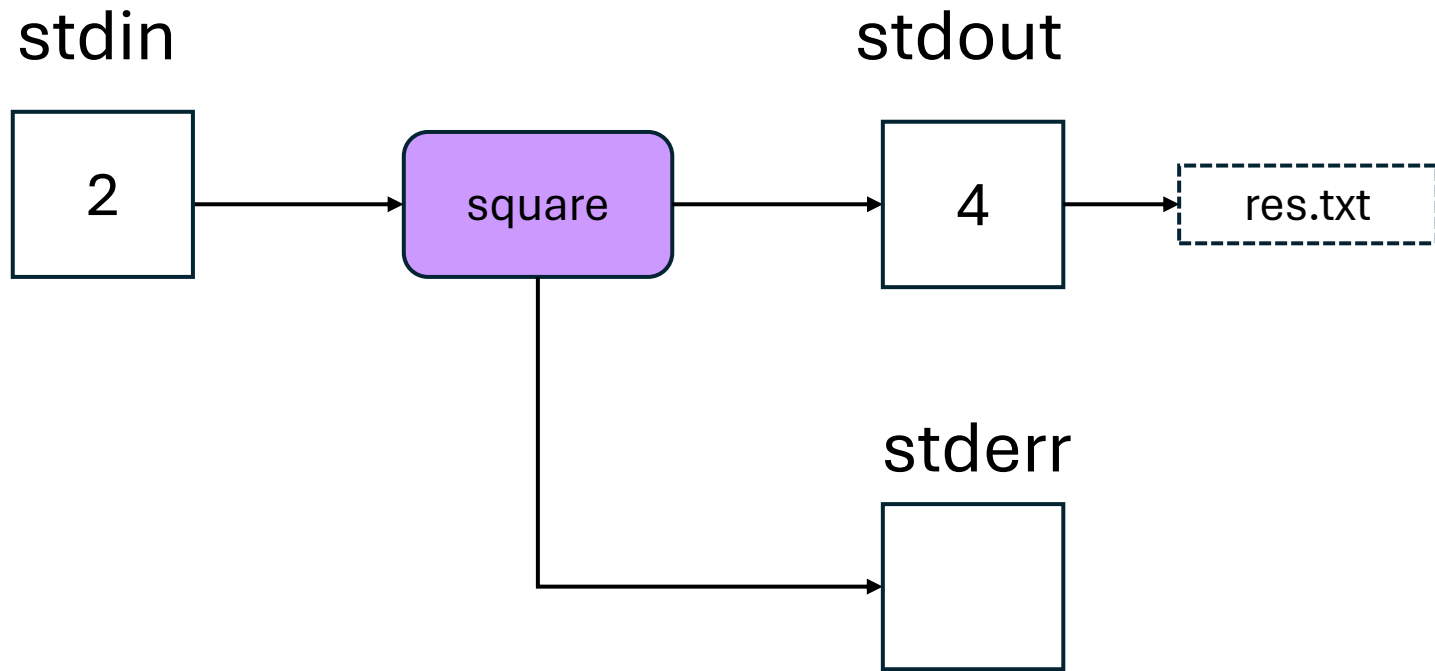
Input/output streams



```
$ square 2 > res.txt
```

overwrite

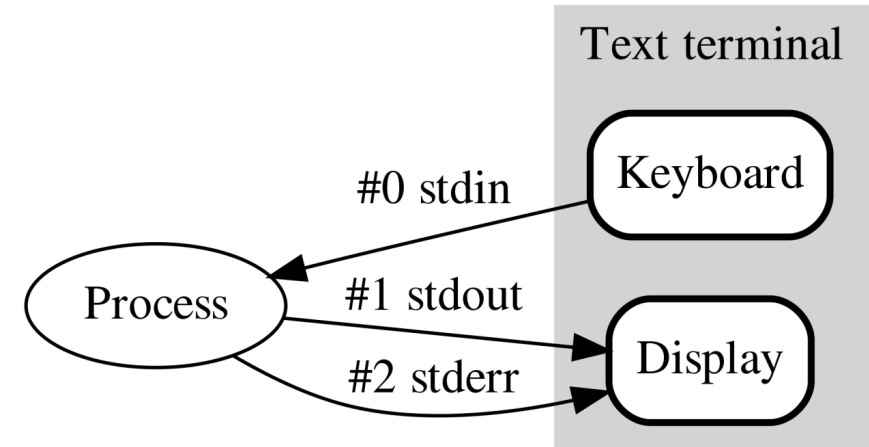
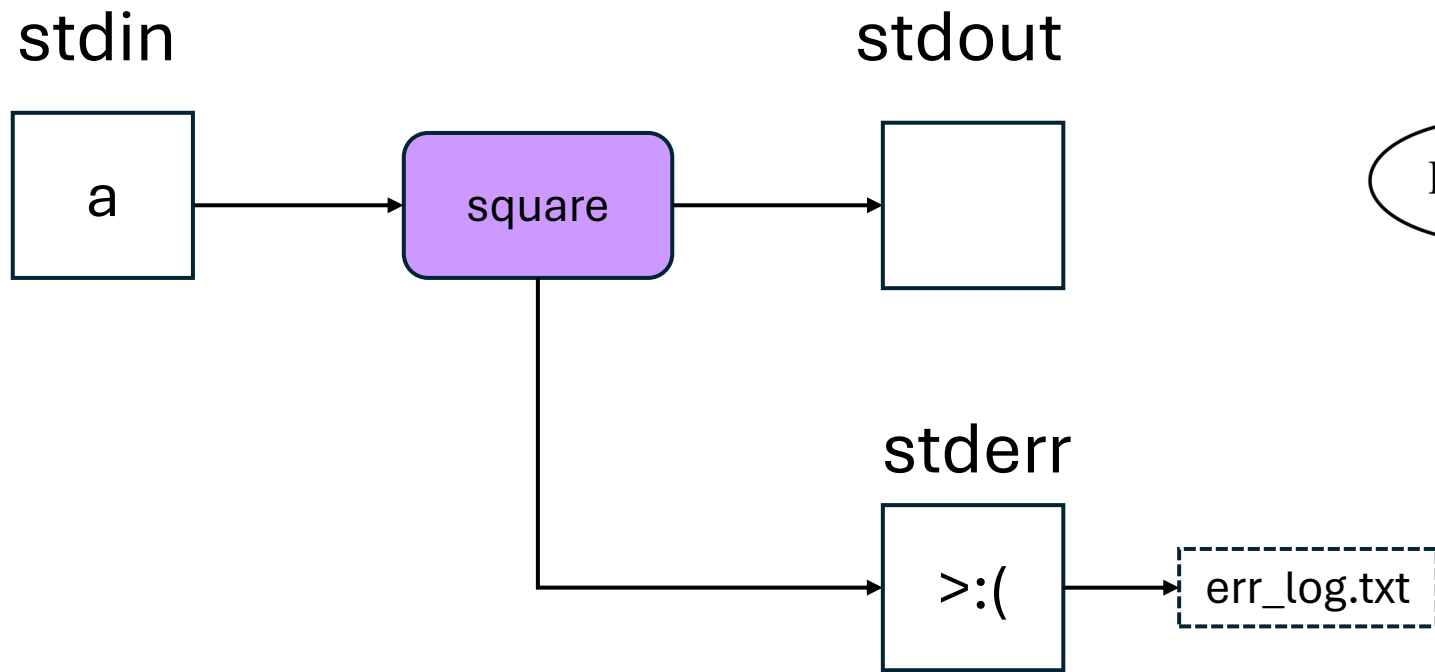
Input/output streams



```
$ square 2 >> res.txt
```

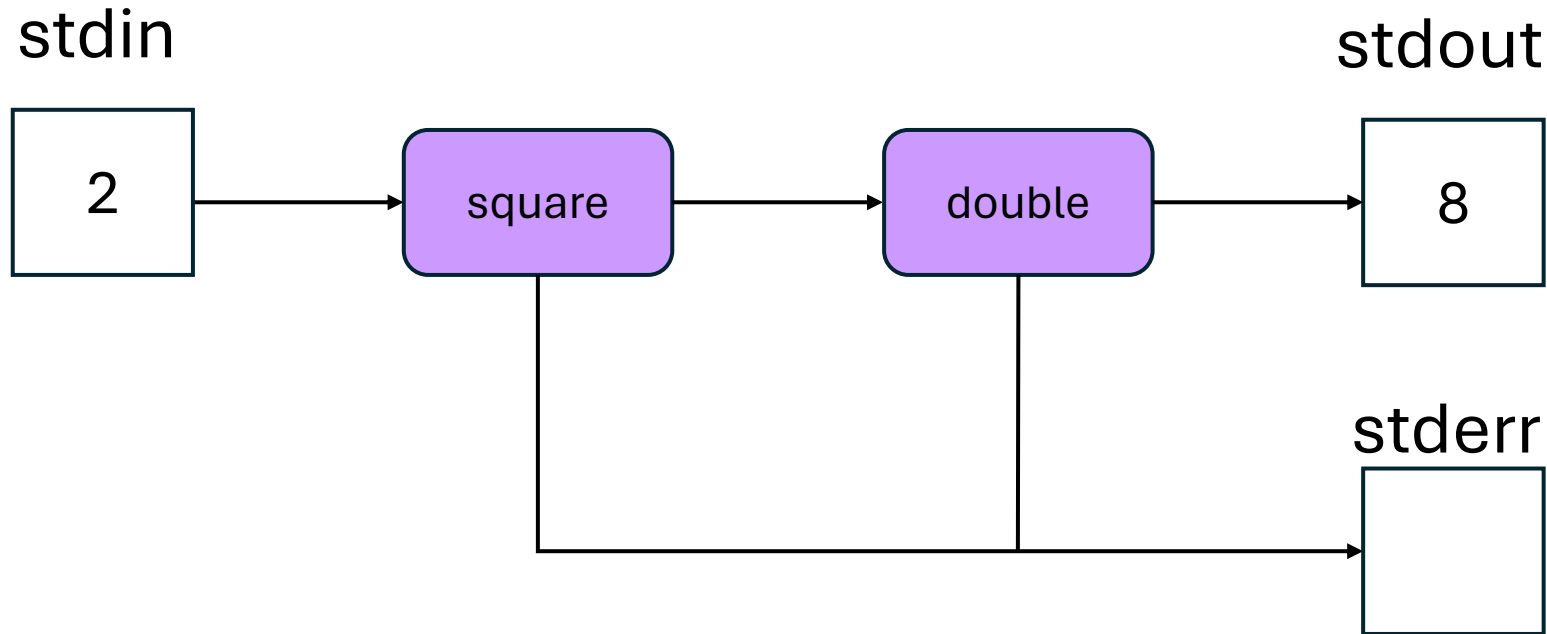
append

Input/output streams



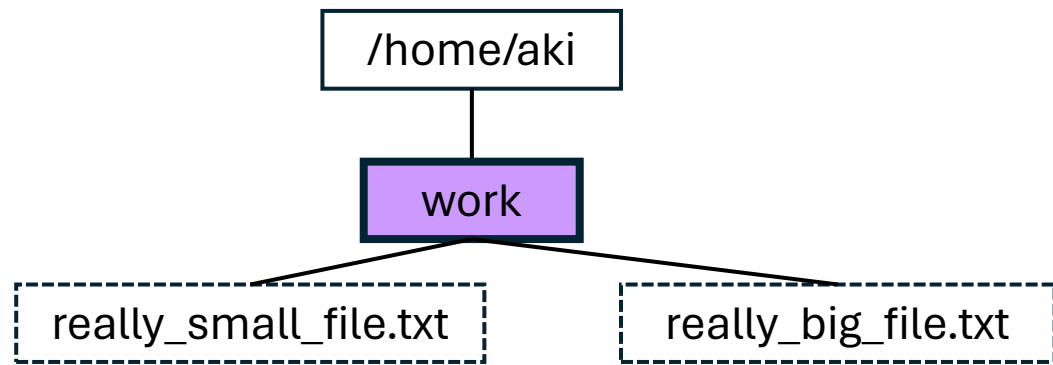
```
$ square a 2> err_log.txt
```

Input/output streams



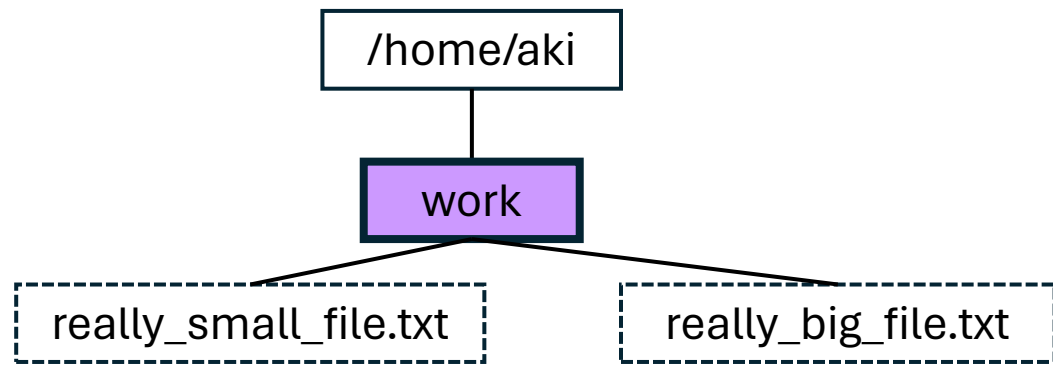
```
$ square 2 | double
```

Redirect stdout with pipe |



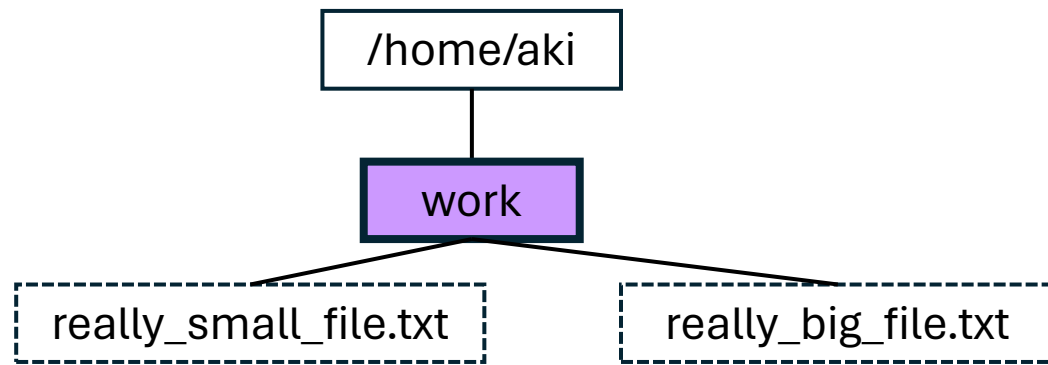
```
$ ls
really_big_file.txt
really_small_file.txt
```

Current directory: `/home/aki/work`



```
$ cat really_small_file.txt
```

Current directory: `/home/aki/work`

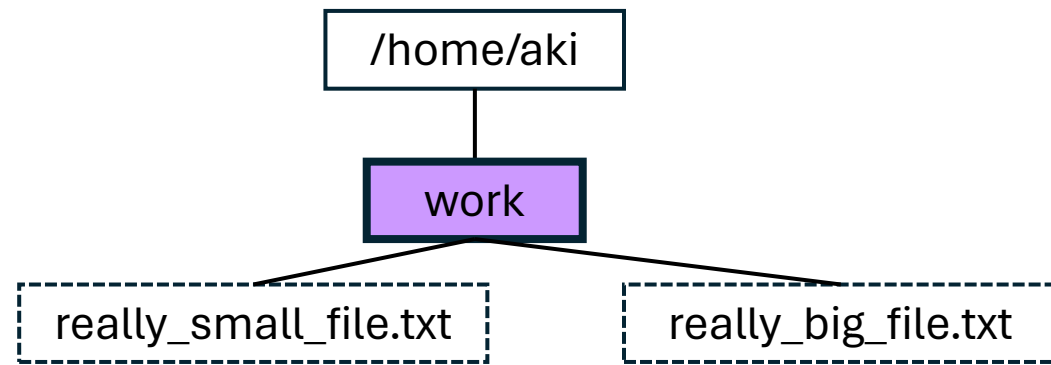


```
$ cat really_small_file.txt
```

```
Hello, I am a really small file 😊  
Don't cat the big file, or your computer will  
explode
```

Concatenate

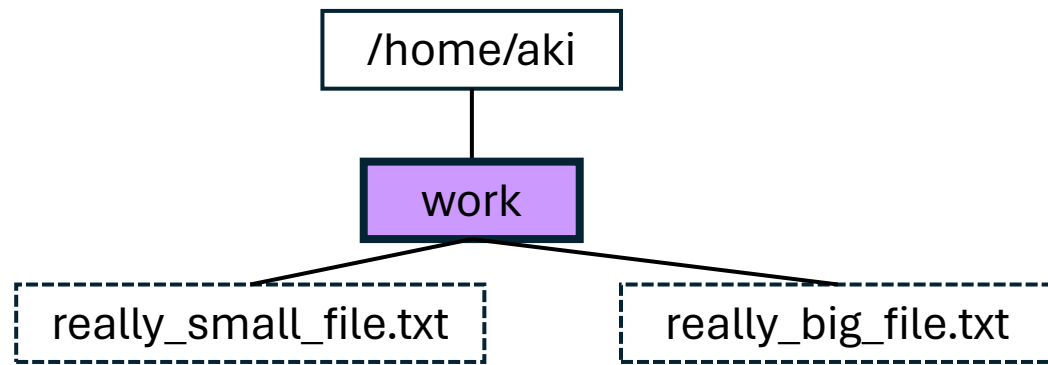
Current directory: `/home/aki/work`



```
$ cat really_small_file.txt
```

```
Hello, I am a really small file 😊  
Don't cat the big file, or your computer will  
explode
```

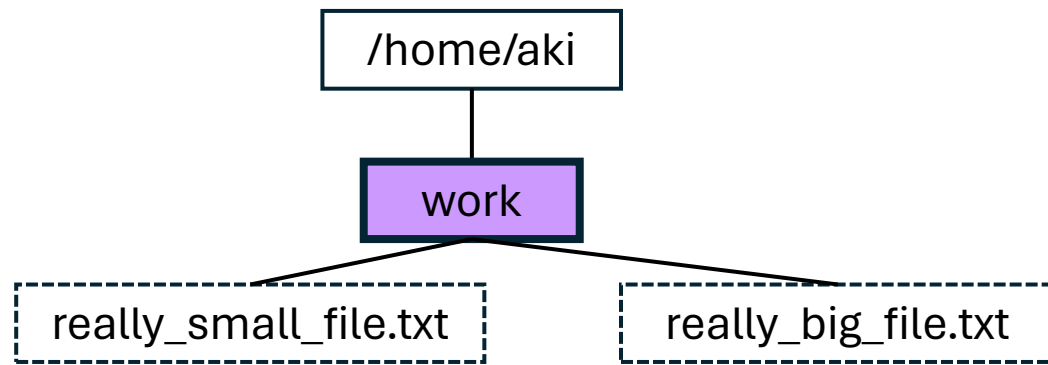
Current directory: `/home/aki/work`



```
$ head really_big_file.txt
I hope you didn't cat this file!
Qwjeiojqhweroijqfoqjwfoqwjfqw
Qwerqwoihrrjqwoihjrqiowjrqiowr
Qiodjfnuiohesguiewqfgqwiongqw ...
```

Output first part of file
Default: 10 lines
-n: number of lines

Current directory: `/home/aki/work`

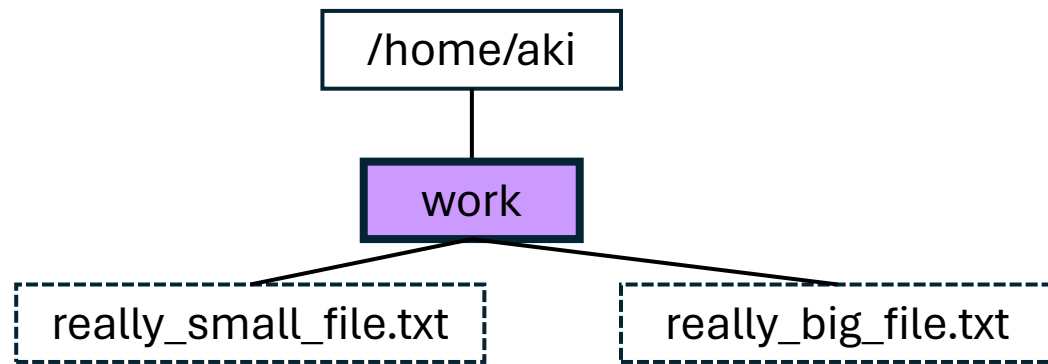


```
$ tail really_big_file.txt
```

```
...  
Asd0iahjdoiqwjf0i9qjwgoqwgoqwj  
Qwjogfijqwogijqwiogjqwiogjqwiog  
I lied. The file isn't that big 😊
```

Output last part of file
Default: 10 lines
-n: number of lines


Current directory: `/home/aki/work`



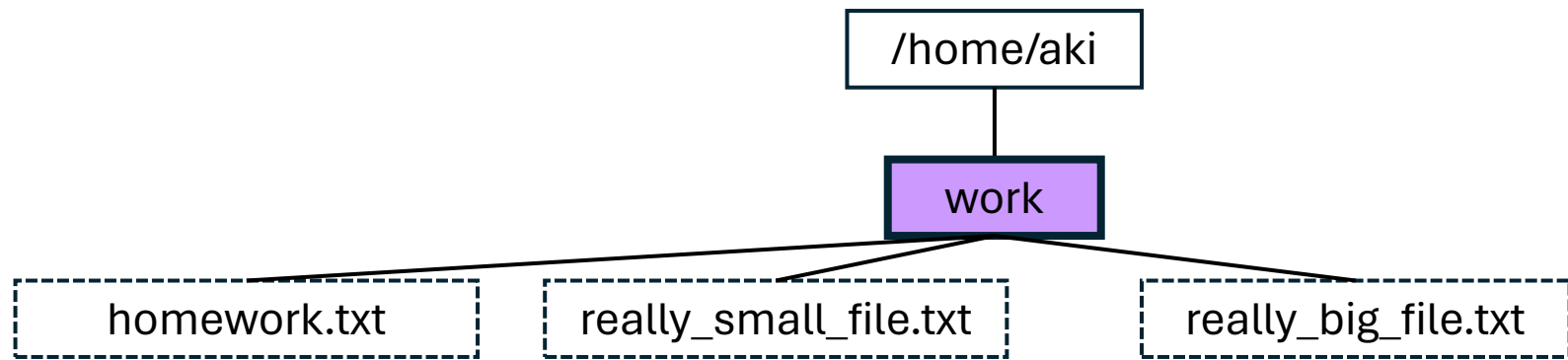
```
$ wc -l really_big_file.txt
15 really_big_file.txt
```

Word count
-l: number of lines
-n: number of words

Current directory: `/home/aki/work`

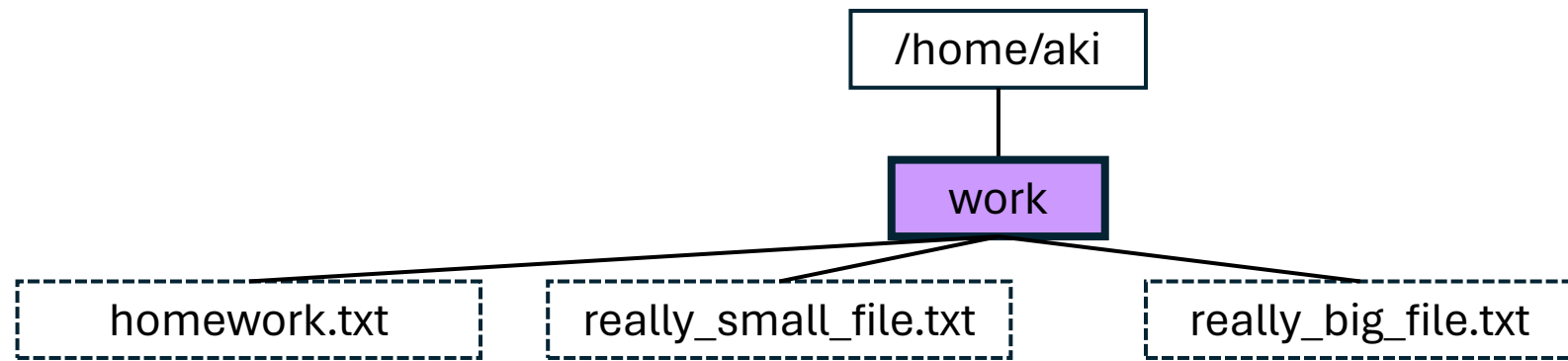
A man with dark hair and a beard is lying in bed, looking towards the camera. He is wearing a light blue shirt. In the foreground, a round analog clock is visible, showing a time around 10:10. A thought bubble is superimposed over the image, containing a terminal command.

```
$ wc -l really_big_file.txt  
15 really_big_file.txt
```



```
$ ls
really_big_file.txt
really_small_file.txt
homework.txt
```

Current directory: `/home/aki/work`

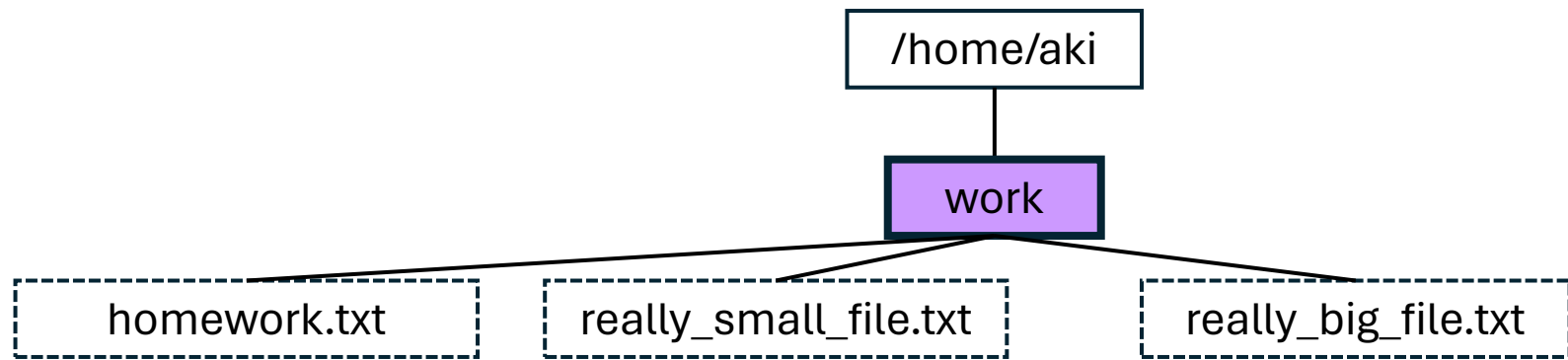


```
$ cat homework.txt
```

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

Current directory: `/home/aki/work`

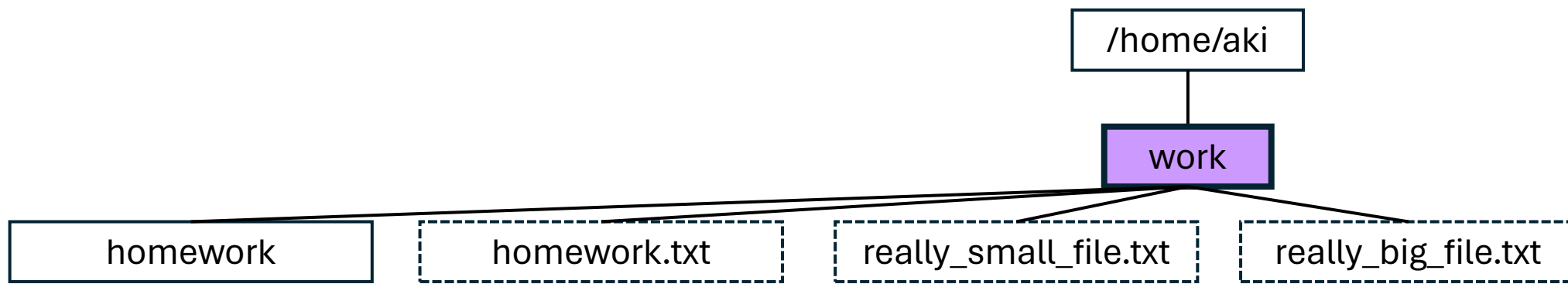


\$

Do this:

1. **Make new directory called homework**
2. Make a copy of `homework.txt` and move it to `homework` dir
3. Delete `really_big_file.txt` from `work` dir
4. Rename `really_small_file.txt` to `small.txt`
5. Write `DONE` at the end of `homework/homework.txt`

Current directory: `/home/aki/work`

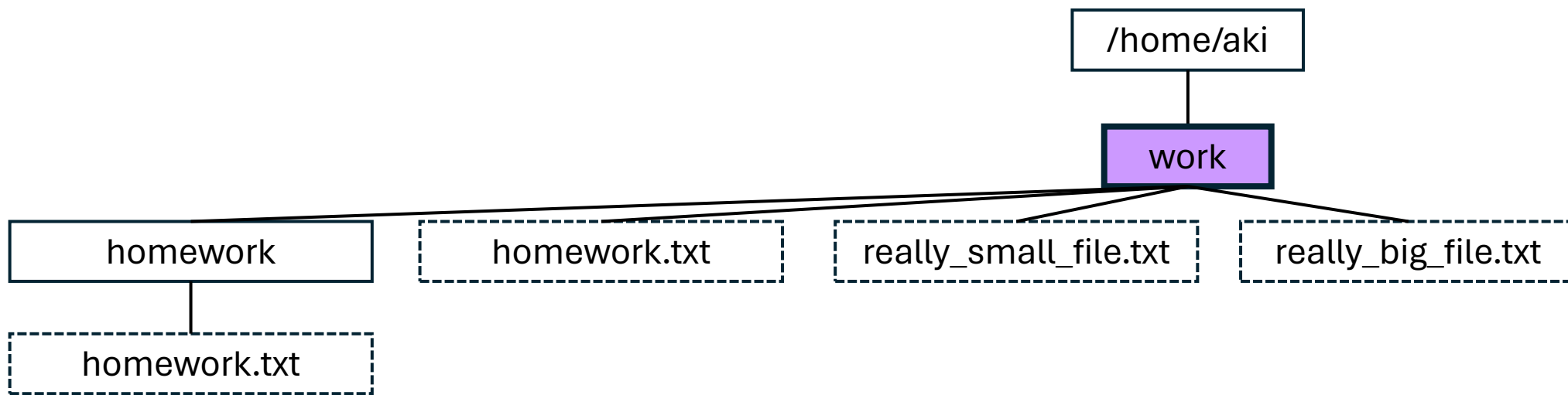


```
$ mkdir homework
```

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

Current directory: /home/aki/work

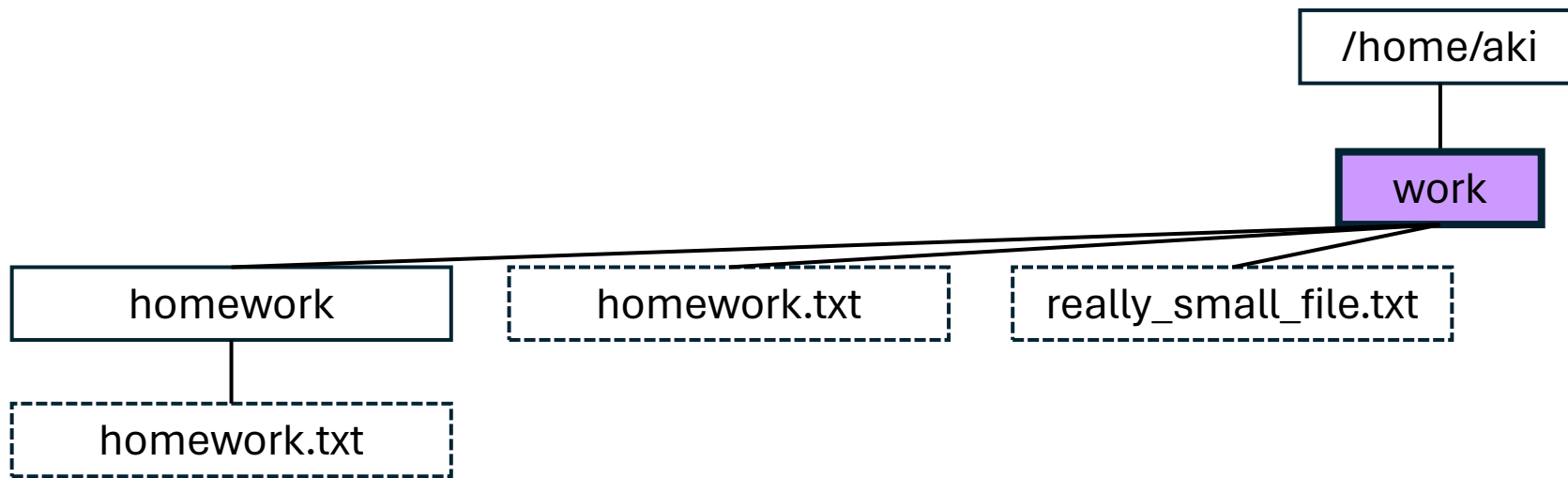


```
$ cp homework.txt homework/
```

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

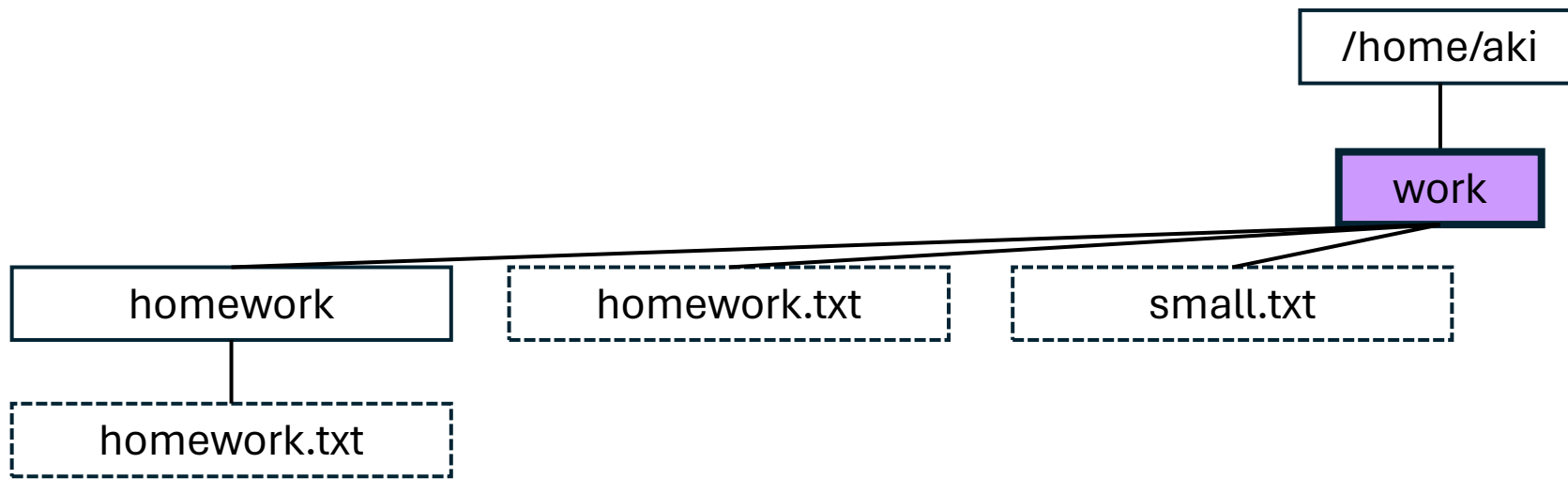
Current directory: /home/aki/work



```
$ rm really_big_file.txt
```

- Do this:
- 1. Make new directory called homework
 - 2. Make a copy of homework.txt and move it to homework dir
 - 3. Delete really_big_file.txt from work dir
 - 4. **Rename really_small_file.txt to small.txt**
 - 5. Write DONE at the end of homework/homework.txt

Current directory: /home/aki/work

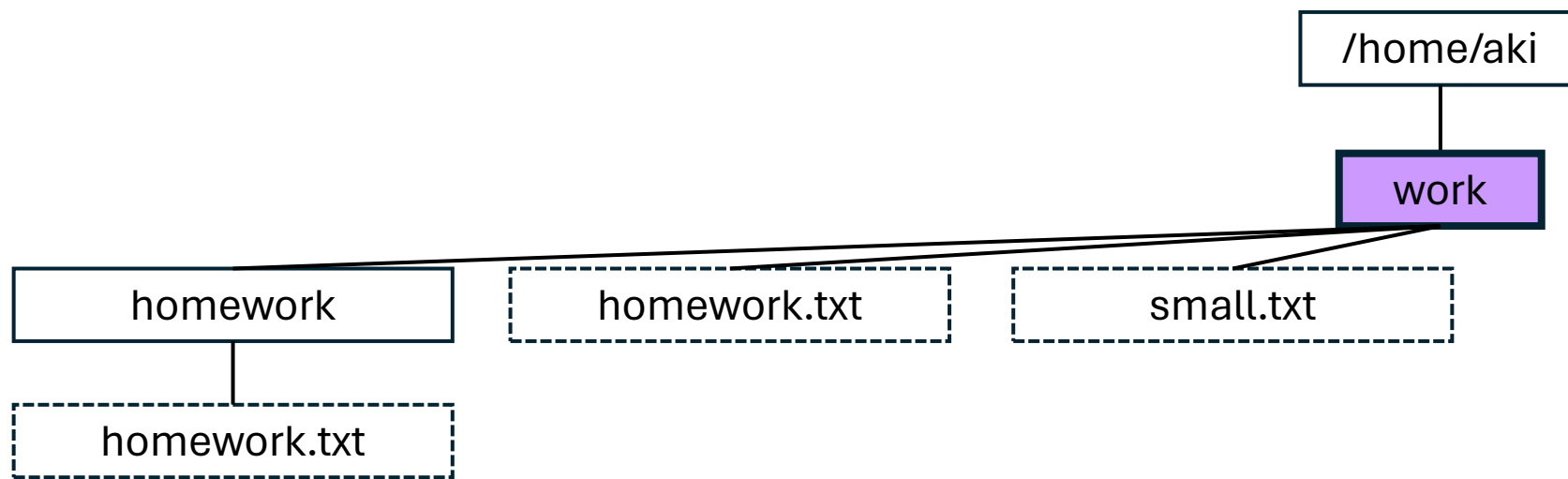


```
$ mv really_small_file.txt small.txt
```

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

Current directory: /home/aki/work



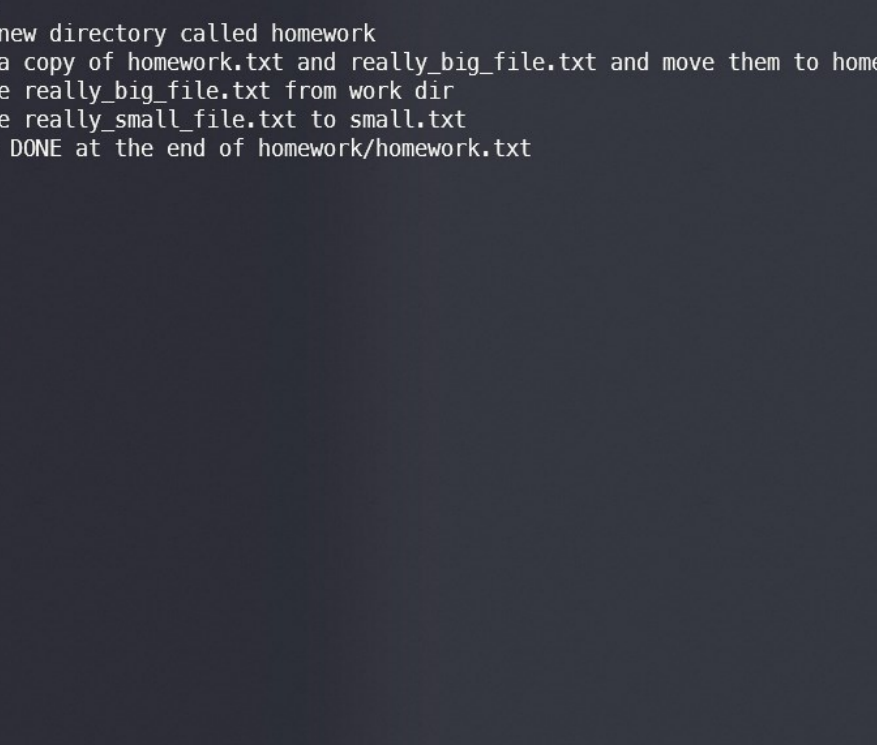
```
$ nano homework/homework.txt  
or  
$ vim homework/homework.txt
```

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

Current directory: /home/aki/work

nano



```
GNU nano 4.8 homework.txt
Do this:
1. Make new directory called homework
2. Make a copy of homework.txt and really_big_file.txt and move them to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

^G Get Help      ^O Write Out    ^W Where Is
^X Exit          ^R Read File    ^\ Replace
^K Cut Text      ^J Justify      ^C Cur Pos
^U Paste Text    ^T To Spell     ^_ Go To Line
```

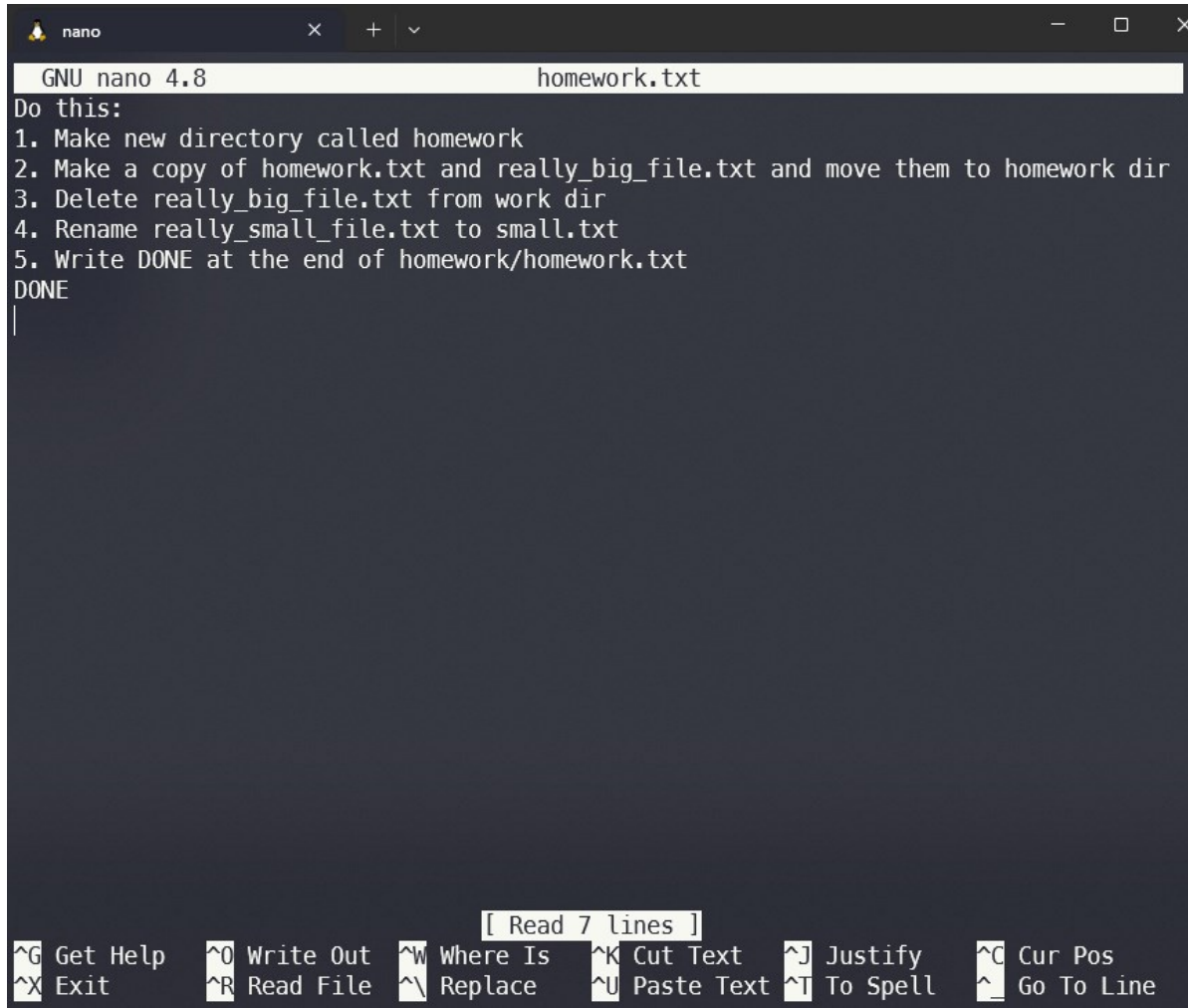
Basic, easy-to-use

vim

[illegible]

Powerful, steep learning curve

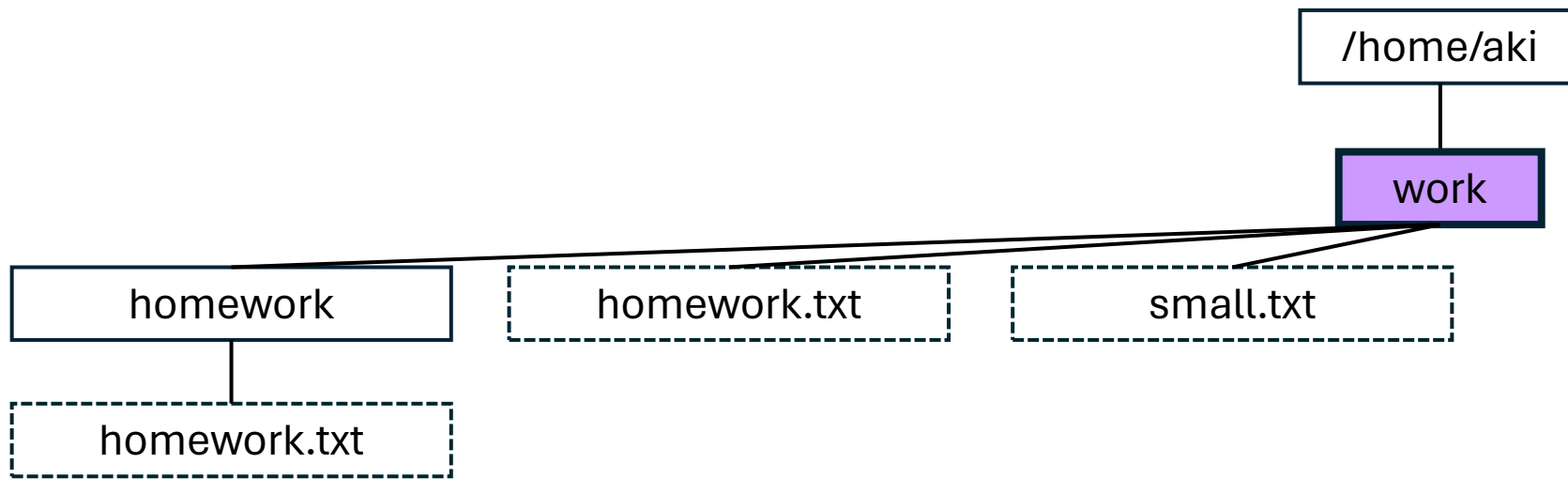
nano



```
GNU nano 4.8 homework.txt
Do this:
1. Make new directory called homework
2. Make a copy of homework.txt and really_big_file.txt and move them to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt
DONE
|

[ Read 7 lines ]
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

1. Move cursor with arrow keys to EOF
(end of file)
2. Type DONE
3. Ctrl (^) + X to exit
4. Y to save
5. Enter to not change file name



\$

Do this:

1. Make new directory called homework
2. Make a copy of homework.txt and move it to homework dir
3. Delete really_big_file.txt from work dir
4. Rename really_small_file.txt to small.txt
5. Write DONE at the end of homework/homework.txt

Current directory: /home/aki/work

Navigation

cd: change directory
ls: list directory contents
pwd: path to working directory

Reading text files

cat: output contents of file to stdout
head: output first 10 lines to stdout
tail: output last 10 lines to stdout

wc -l: output number of lines
wc -w: output number of words

Editing text files

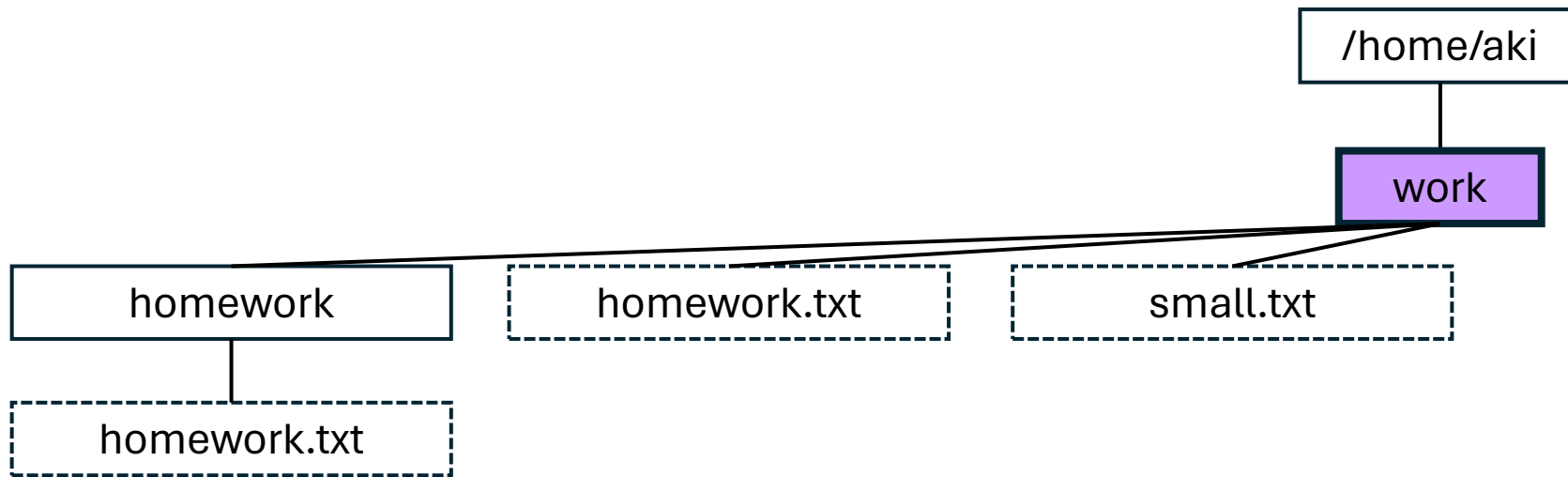
nano: basic, easy-to-use text editor
vim: powerful, steep learning curve

File manipulation

cp: copy file(s)
mv: move or rename a file
rm: permanently remove a file

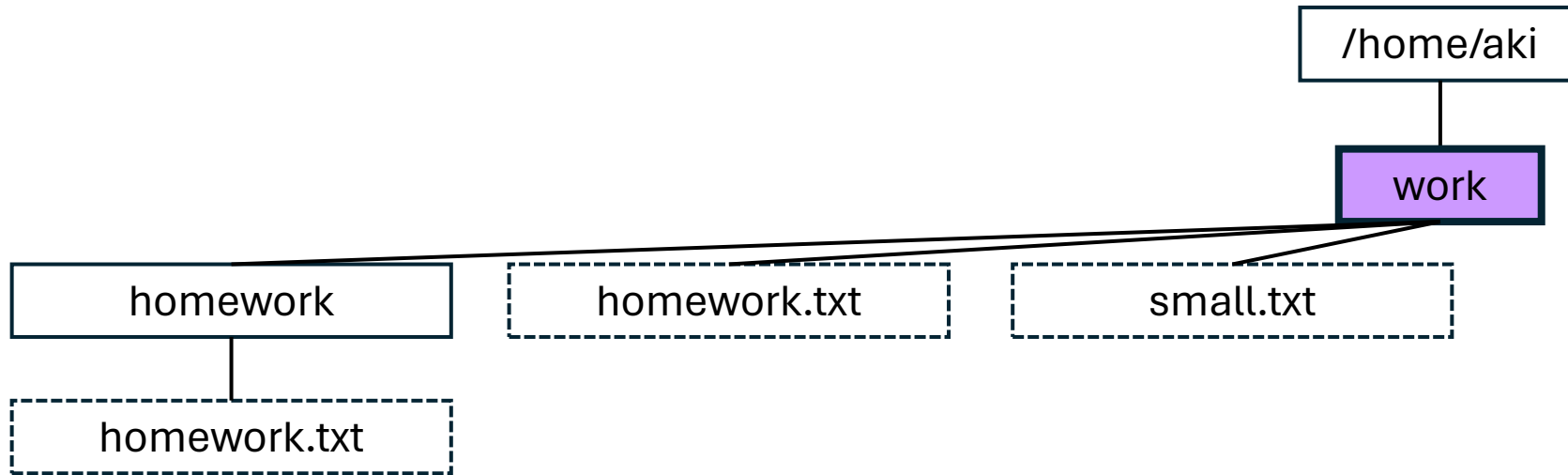
Looking stuff up

man: manual
Google 😊



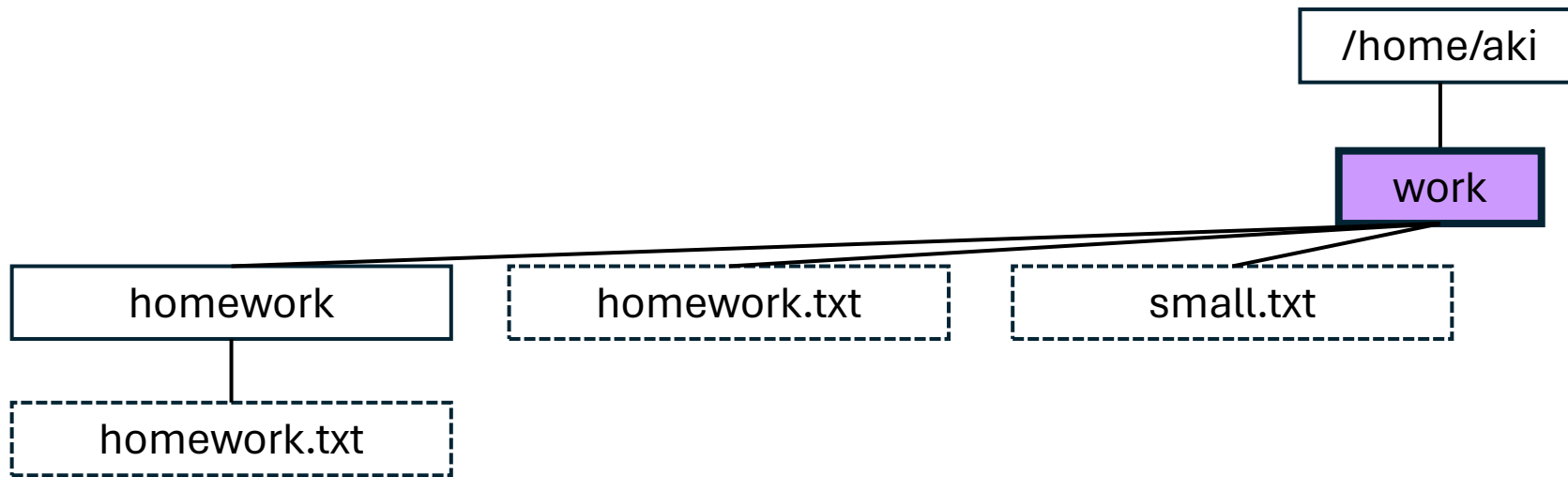
```
$ ls
homework
homework.txt
small.txt
```

Current directory: `/home/aki/work`



```
$ ls *  
homework  
homework.txt  
small.txt
```

Current directory: `/home/aki/work`



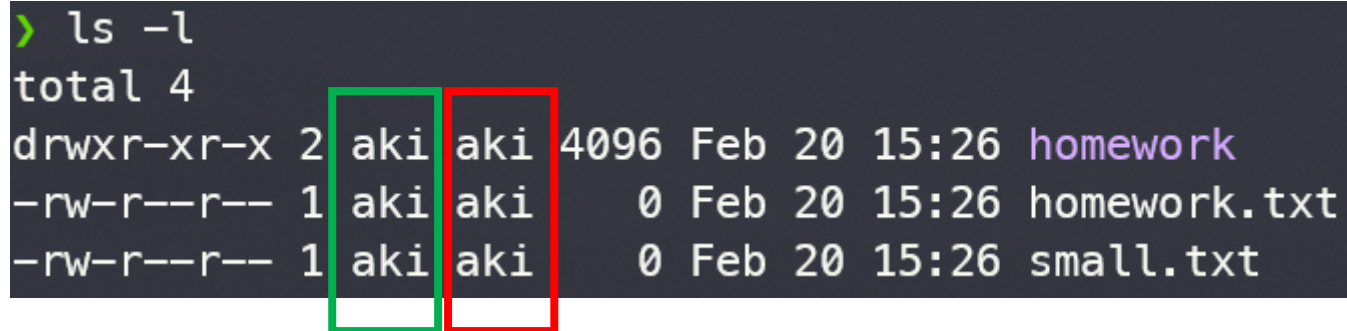
```
$ ls *.txt  
homework.txt  
small.txt
```

Current directory: `/home/aki/work`

UNIX file permissions

- Files have an **owner** and a **group** determining who has access to the file
 - Default: File is owned by the user or program who created it.
 - File group is usually one of the groups the owner belongs to.

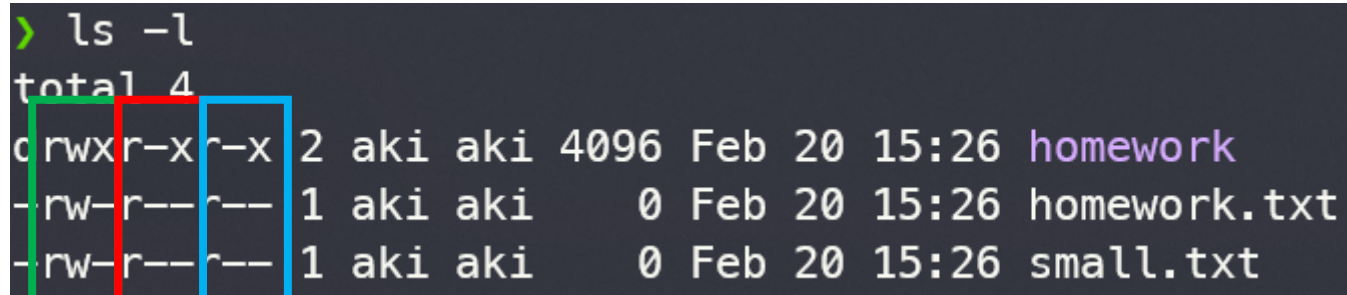
```
> ls -l
total 4
drwxr-xr-x 2 aki aki 4096 Feb 20 15:26 homework
-rw-r--r-- 1 aki aki  0 Feb 20 15:26 homework.txt
-rw-r--r-- 1 aki aki  0 Feb 20 15:26 small.txt
```



UNIX file permissions

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

```
> ls -l
total 4
drwxr-xr-x 2 aki aki 4096 Feb 20 15:26 homework
-rw-r--r-- 1 aki aki    0 Feb 20 15:26 homework.txt
-rw-r--r-- 1 aki aki    0 Feb 20 15:26 small.txt
```



UNIX file permissions

To change the mode (permissions) of a file/directory

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

```
$ chmod o+rw file.txt
```

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

```
$ chmod o+rw file.txt
```

Give others (o) read and write permissions

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

```
$ chmod u-w,g-w,o-w file.txt
```

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

```
$ chmod u-w,g-w,o-w file.txt
```

Remove write permissions for user, group, and others

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

```
$ chmod g=rx file.txt
```

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

UNIX file permissions

chmod

- **+**: add the following permissions
- **-**: remove the following permissions
- **=**: permissions are exactly as follows

```
$ chmod g=rx file.txt
```

Set group permissions to read and execute (not write)

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

UNIX file permissions

chmod

- Set all permissions with number
 - Use a Chmod Calculator
 - <https://chmod-calculator.com/>

- Files permissions are shown in the following order:
 - Owner (u)
 - Group (g)
 - Other/all (o)
- Each group has (or does not have) permission to:
 - Read (r)
 - Write (w)
 - Execute (x)

Owner

Read ☒

Write ☐

Execute ☐

Group

Read ☐

Write ☒

Execute ☐

Public

Read ☐

Write ☐

Execute ☒

Linux
Permissions:

What have we learned so far?

- Navigation
 - Reading text files
 - Editing text files
 - Manipulating files
 - Changing permissions
-
- READ THE DOCS

Let's talk bioinformatics

- Common file types

Let's talk bioinformatics

- Common file types
 - FASTQ
 - BED
 - VCF
 - MAF
 - SAM
 - BAM
 - GFF/GFT
 - CSV
 - TSV

Let's talk bioinformatics

- Common file types

- FASTQ
- BED
- VCF
- MAF
- SAM
- **BAM**
- GFF/GFT
- CSV
- TSV

Plain text file

Binary file

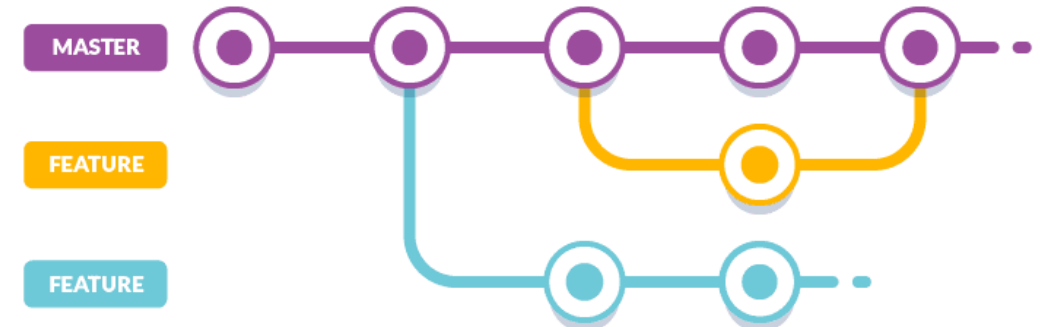
Git

- Popular version control system
- Created by Linus Torvalds in 2005



Git

- Popular version control system
- Created by Linus Torvalds in 2005
- What does Git do?
 - Manage projects with **Repositories**
 - **Clone** a project to work on a local copy
 - Control and track changes with **Staging** and **Committing**
 - **Branch** and **Merge** to allow for work on different parts and versions of a project
 - **Pull** the latest version of the project to a local copy
 - **Push** local updates to the main project



GitHub



- Not the same as Git
- GitHub makes tools that use Git
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018
- Other hosts exist
 - GitLab
 - Bitbucket
- Did you know: You have access to **free** GitHub pro as a student?
 - You should really look into it
 - <https://education.github.com/pack>

GitHub

- Most bioinformatic tools will be available through GitHub
- The rest of the workshop can be found in this repository:

<https://github.com/5Aki1/UNIX-Linux-Workshop>

```
$ git clone https://github.com/5Aki1/UNIX-Linux-Workshop
```

Slides will be available in the repository after the workshop. Run:

```
$ git pull
```