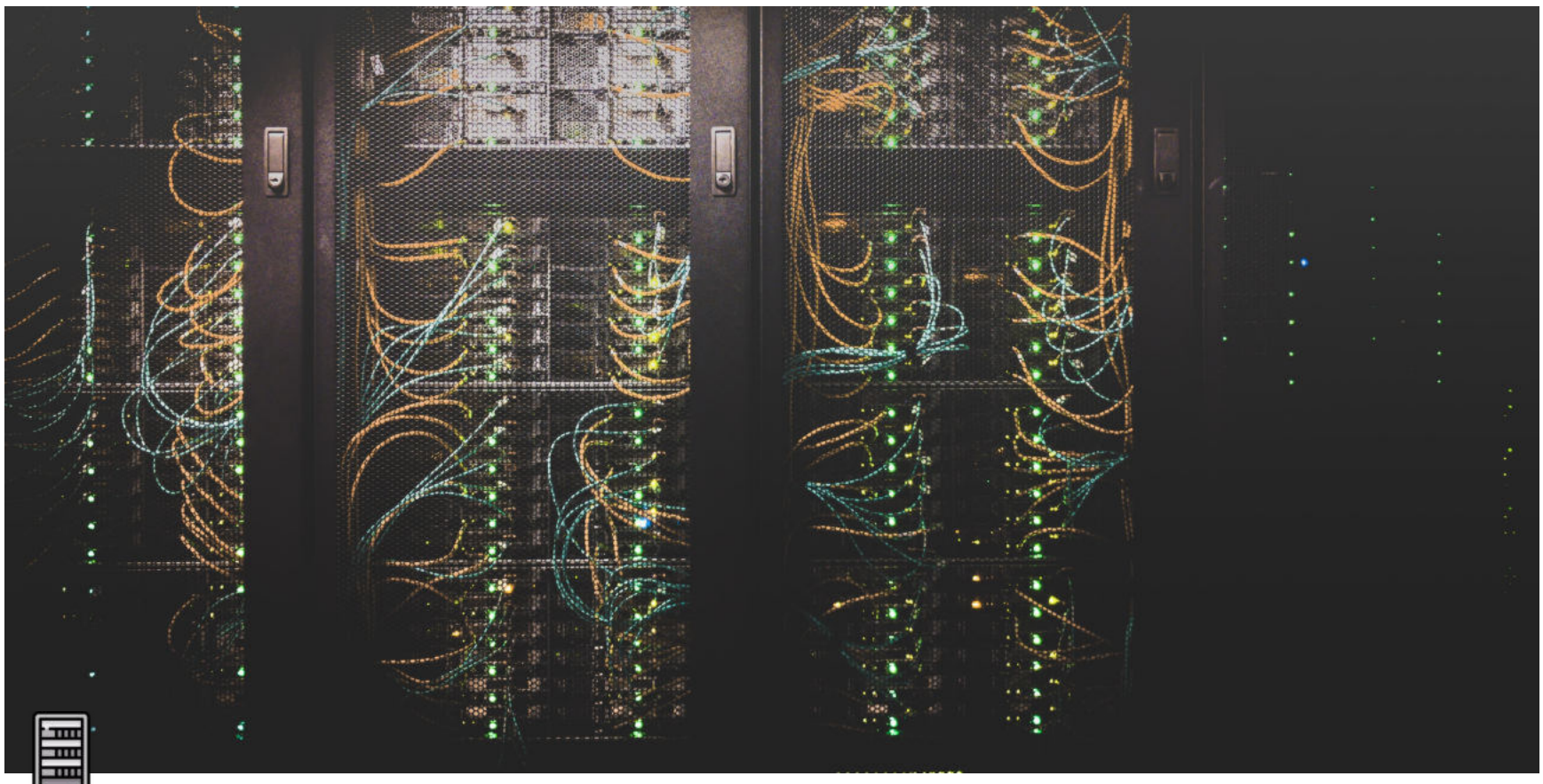


Launching a Virtual Machine

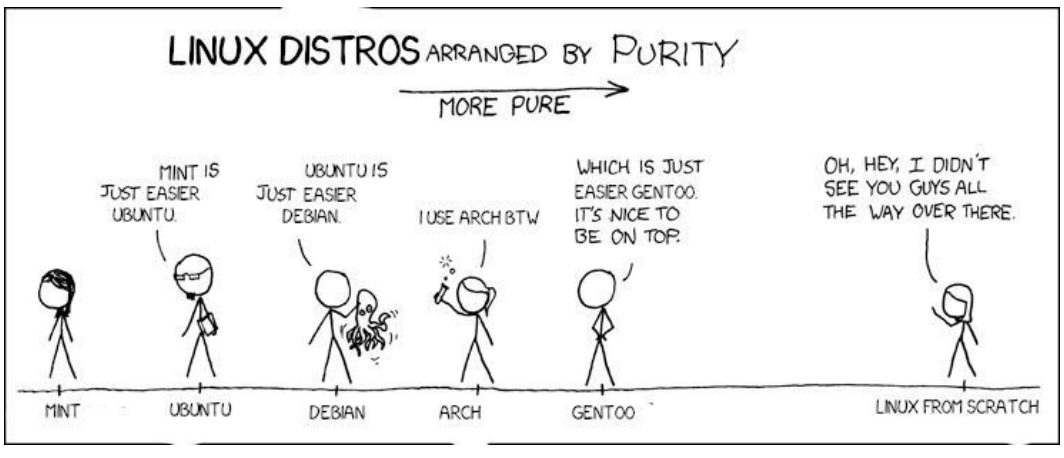
Type	📒 Lecture
Date	@December 22, 2021
Lecture #	1
Lecture URL	https://youtu.be/PHrN7yp1AJw
Notion URL	https://21f1003586.notion.site/Launching-a-Virtual-Machine-e493cecce6a743a9b37516d196c07c1d
Week #	1

Why do we need a Virtual Machine (VM)?

- Laptops usually come pre-installed with Windows
  - Unless, of course, you are an 🍎 person
- We wish to try Linux almost natively, without removing the existing OS
  - Also considering the fact that we do not wish to dual boot

Requirements

- An .iso image of the operating system we want
  - Ubuntu 20.04 is recommended, or just use arch btw



    - Can be downloaded here
- A Hypervisor
  - A **hypervisor**, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor

allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

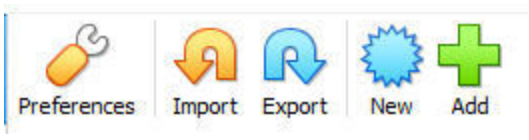
Source: https://www.vmware.com/topics/glossary/content/hypervisor.html

- Oracle VirtualBox
- VMWare Workstation Player
- or just use Windows Subsystem for Linux
- Atleast 20GB free space for the VM
- Some RAM ¯_(ツ)_/¯
  - 8GB+ recommended

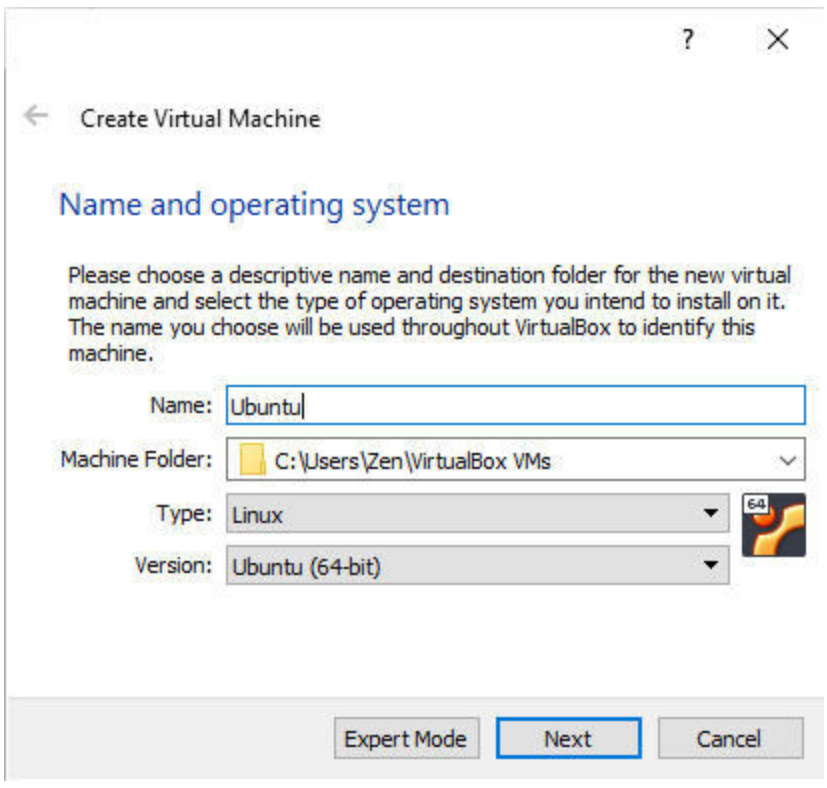
Steps

- Download the Ubuntu .iso file from https://ubuntu.com/download/desktop
- Download either VirtualBox or the VMWare Workstation Player
  - Install them
- Open VirtualBox / VMWare Workstation Player

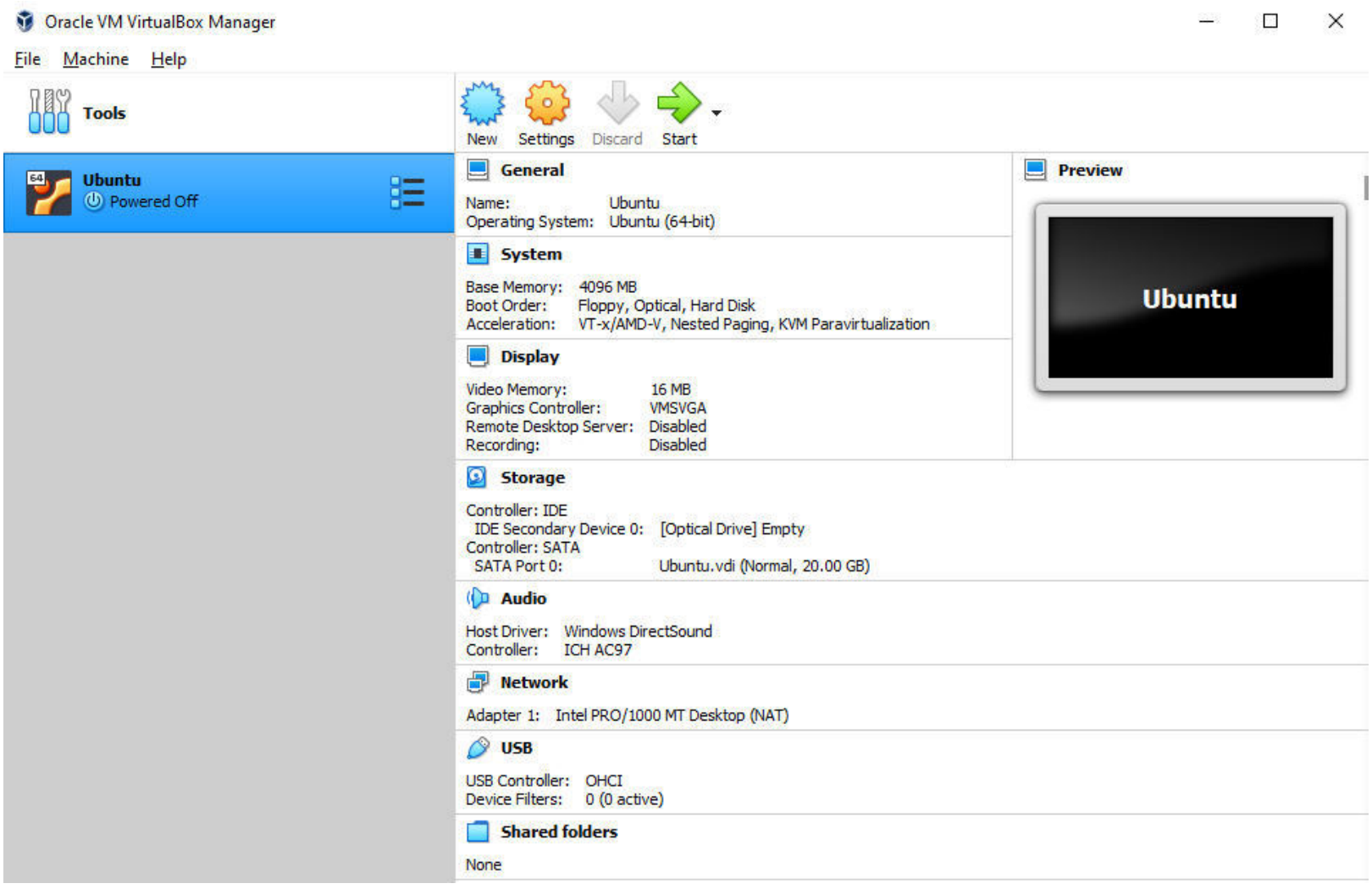
  (I will be using VirtualBox)
  - Click on the “New Button to create a new VM”



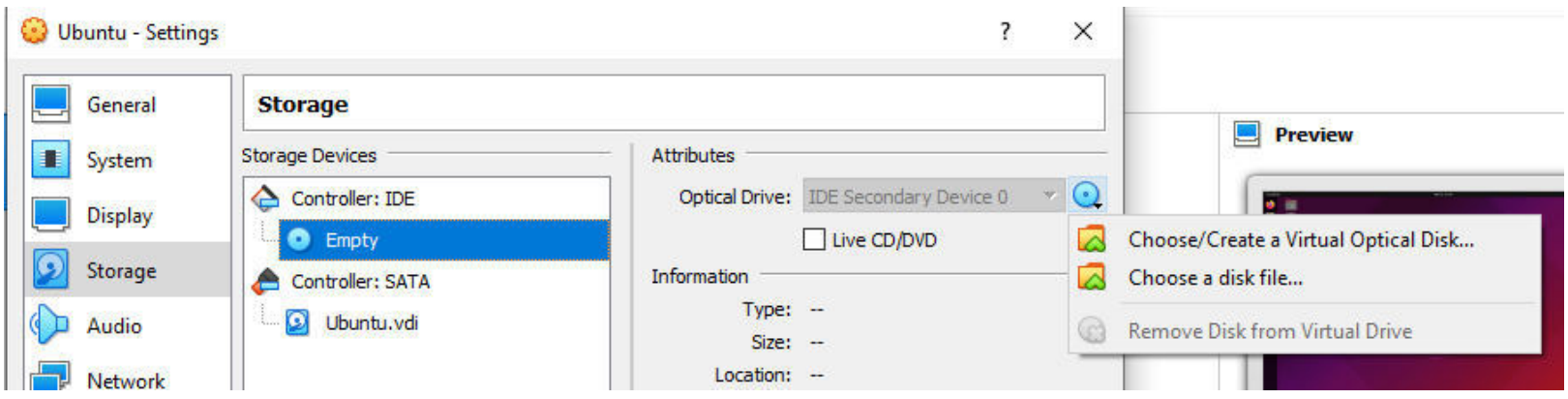
  - Add a name and choose the OS type and version



  - Adjust the RAM and Storage as per your liking, make sure to keep the minimum specs
  - Select the VM on the left menu and go to settings

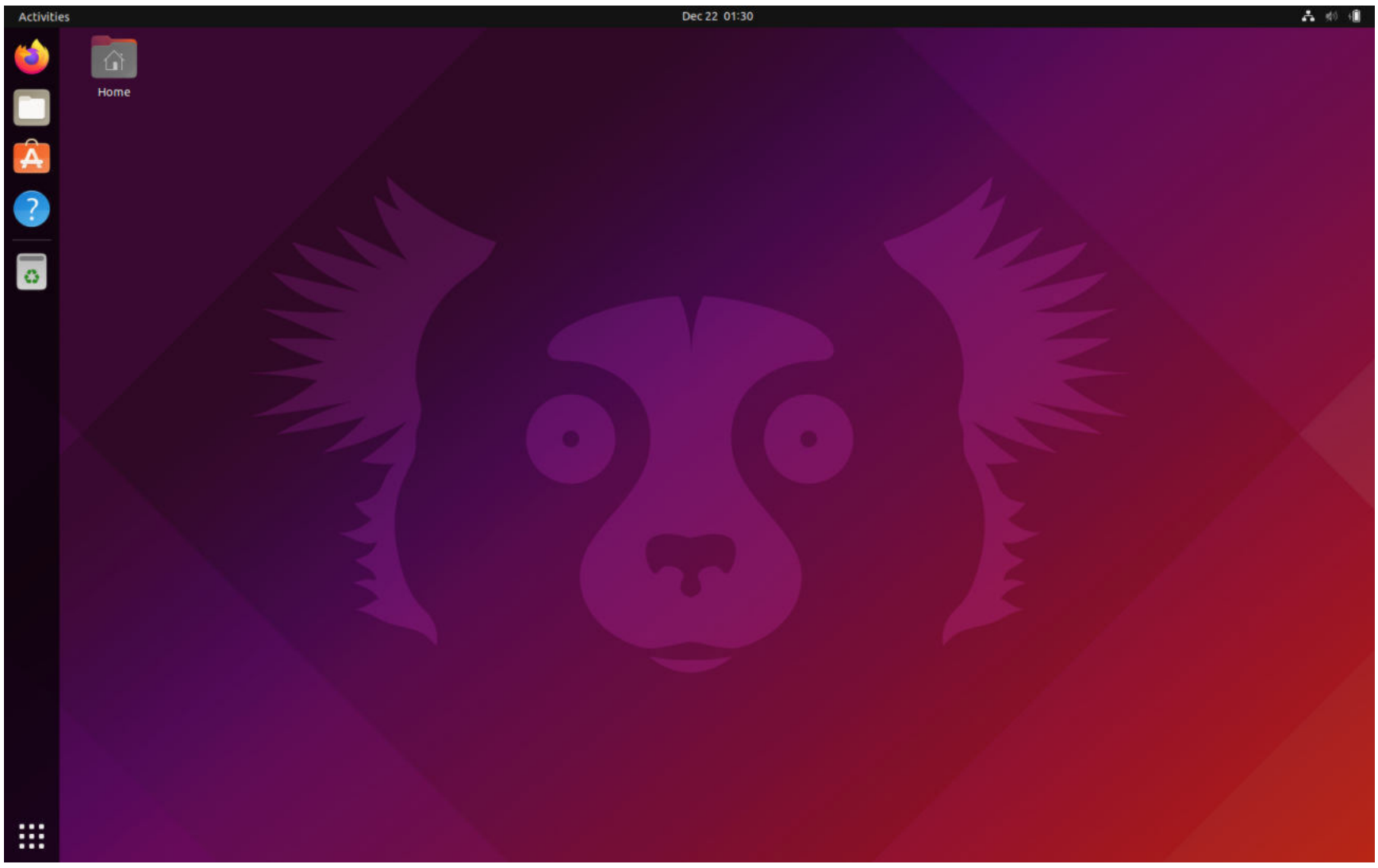


- Go to Storage → Storage Devices → Controller → Empty → Click on the CD icon and choose "**Choose/Create a Virtual Optical Disk...**" and choose your .iso file

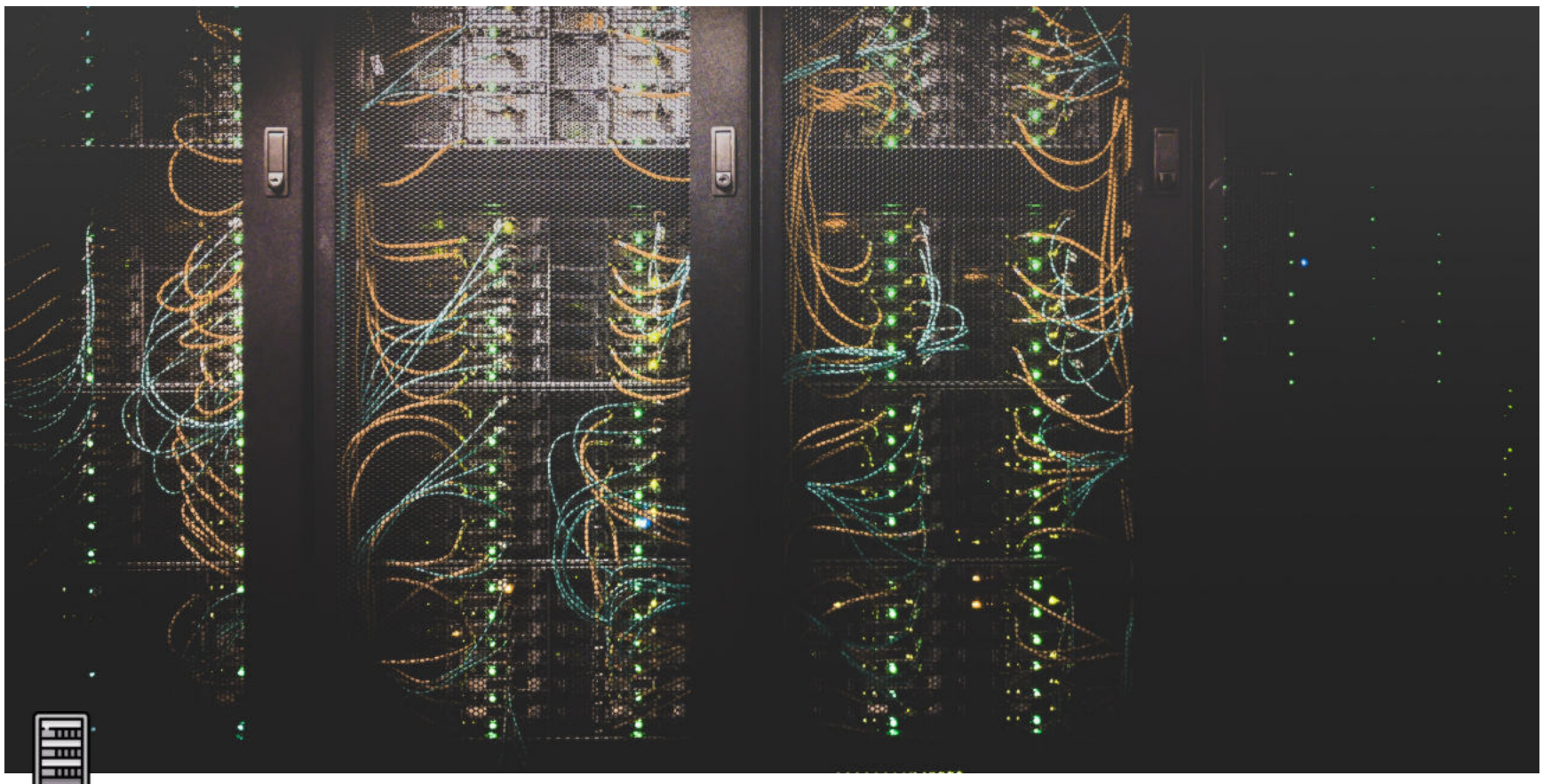


- Proceed with the installation
  - Uhh it's just Next, Next, Next .... Next ... Restart

When you install it correctly and get it up and running, you might see something like this ...



(this screenshot here is Ubuntu 21.10 and gosh that's a creepy monke)



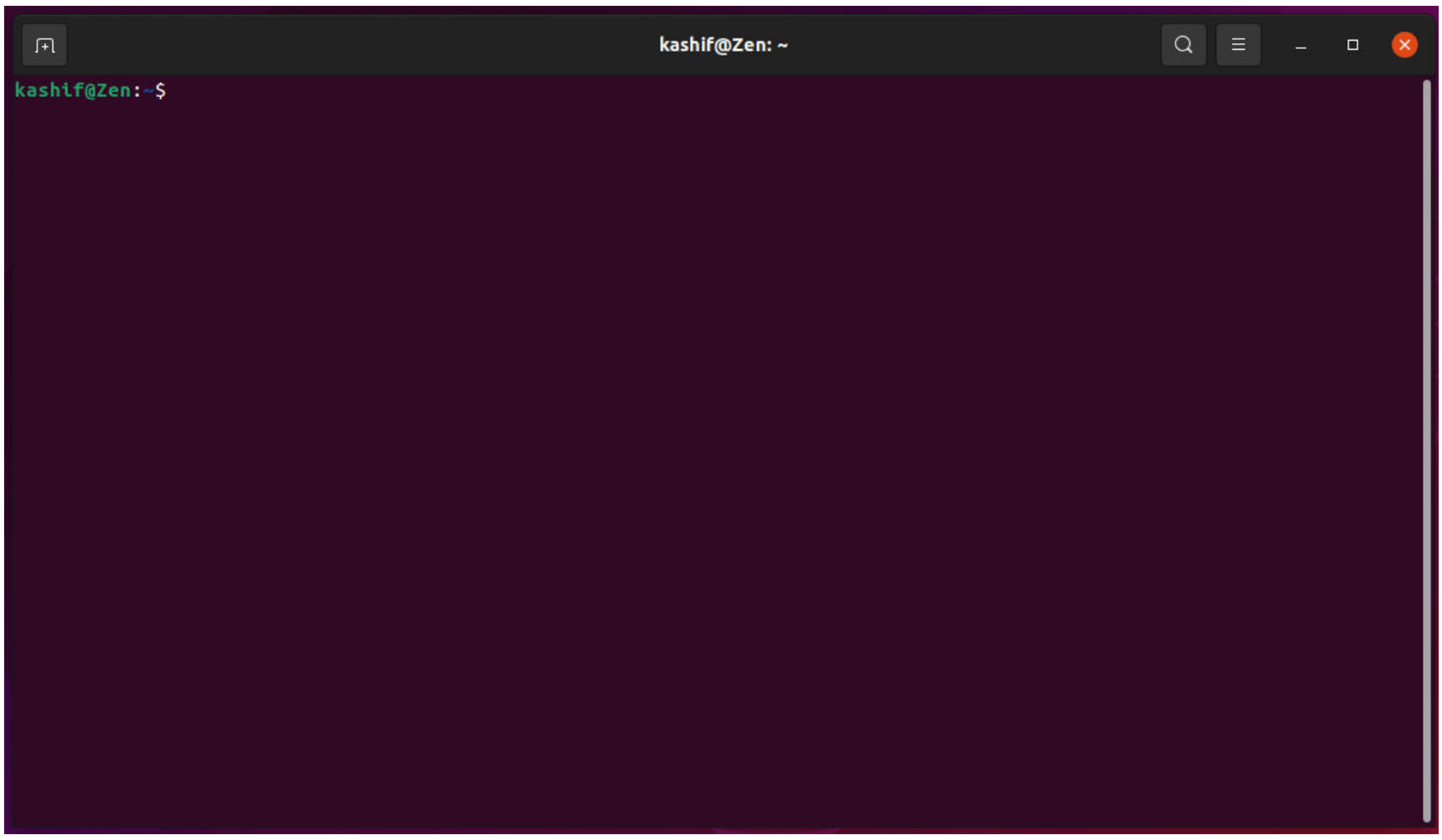
Command Line Environment

Type	📒 Lecture
Date	@December 22, 2021
Lecture #	2
Lecture URL	https://youtu.be/qrAnlpcMyYc
Notion URL	https://21f1003586.notion.site/Command-Line-Environment-f96a91e782a147a88894028d7848a2c4
Week #	1

Why use Command Line environment?

- To use linux at its max potential
- Combine commands to form powerful scripts
    - To automate using these scripts
- *To assert dominance over GUI plebs*

Terminal in Ubuntu



This is what the default terminal looks like in Ubuntu

To clear the command line

```
clear
```

- or you can press `Ctrl + L` to clear the terminal screen

To check which directory we currently are in

```
pwd
```

By default, you are placed in the home directory of the currently logged in user

To list all the files and folders in the current directory

```
ls
```

To view the currently running processes

```
ps
```

To know the OS, duh

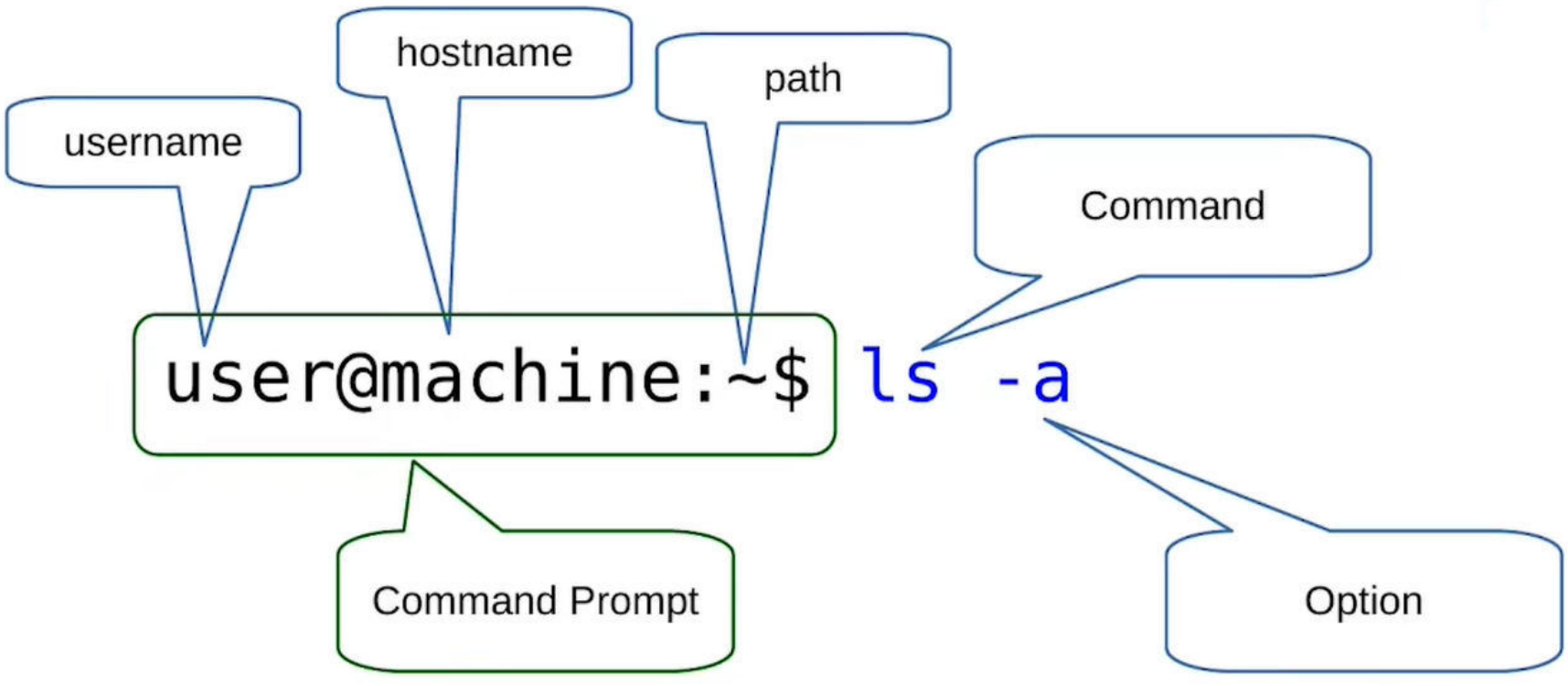
```
uname
```

To exit the shell

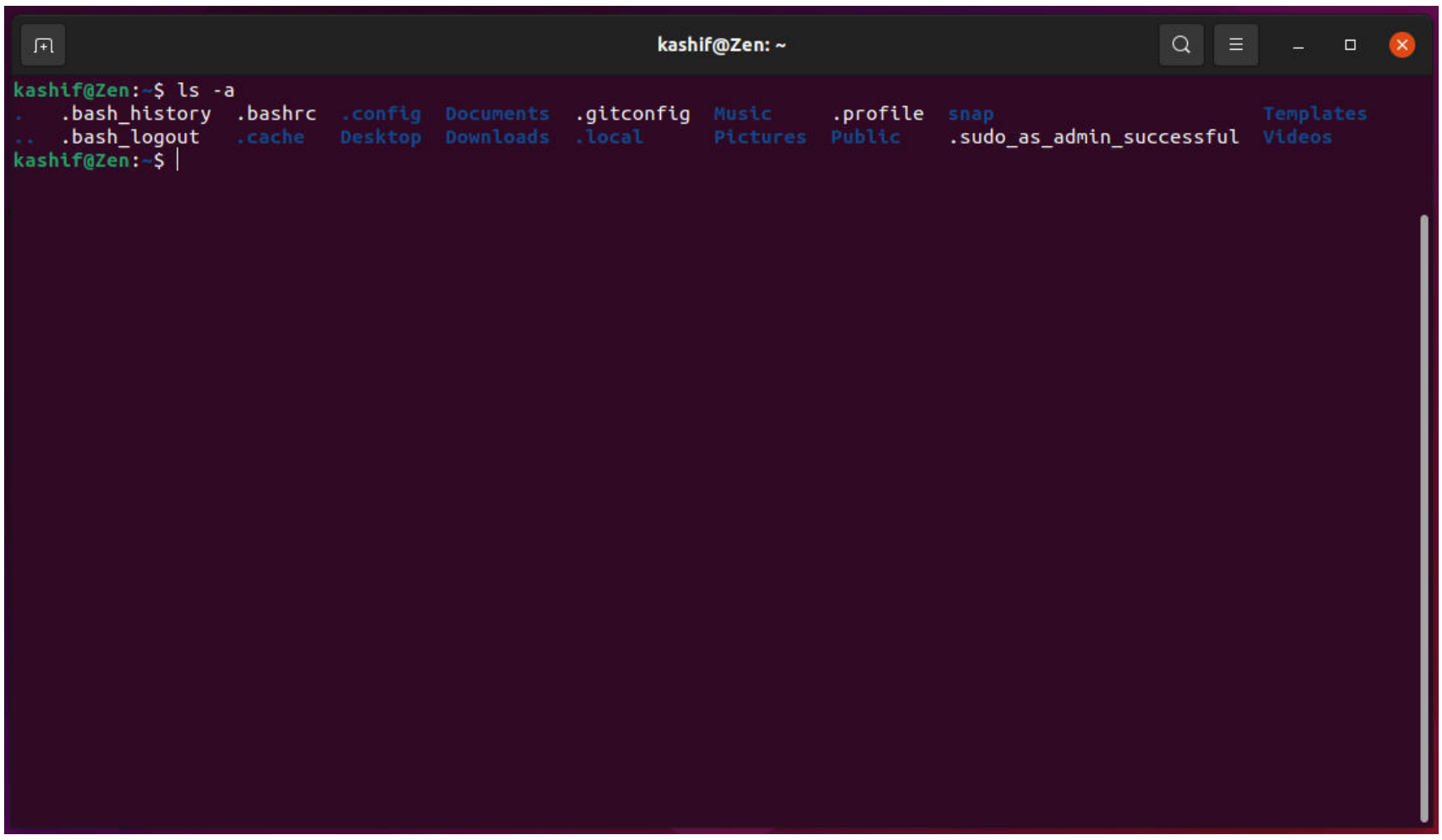
```
exit
```

- Or you can also press `Ctrl + D` to exit out of the terminal session

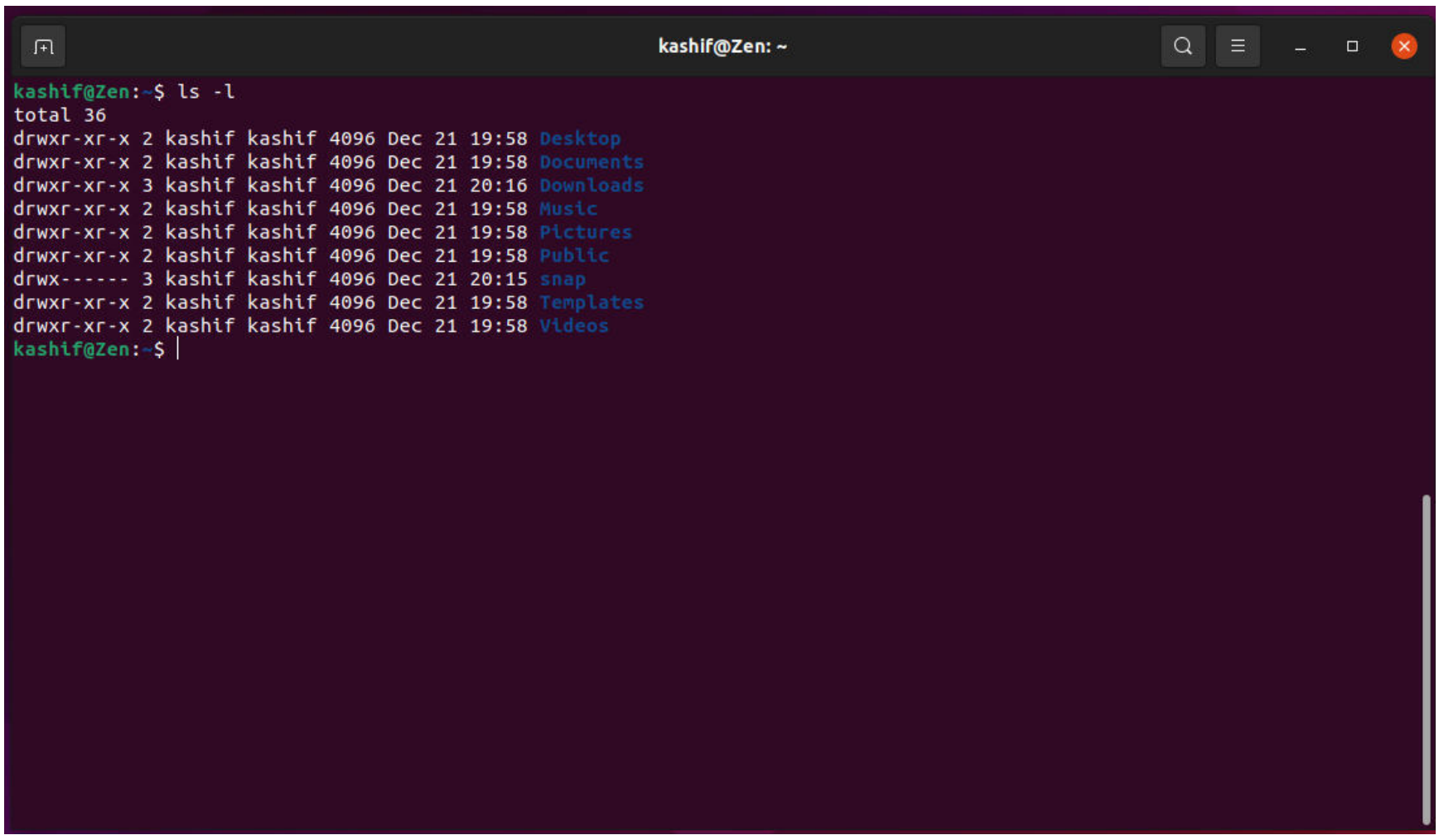
Anatomy of a typical command on the terminal



- To display all the files, press `ls -a`



- To display the files in a list, press `ls -l`



These two flags are the most commonly used ones, ***we can also combine them as one flag***

like, `ls -al`

To get help on any command

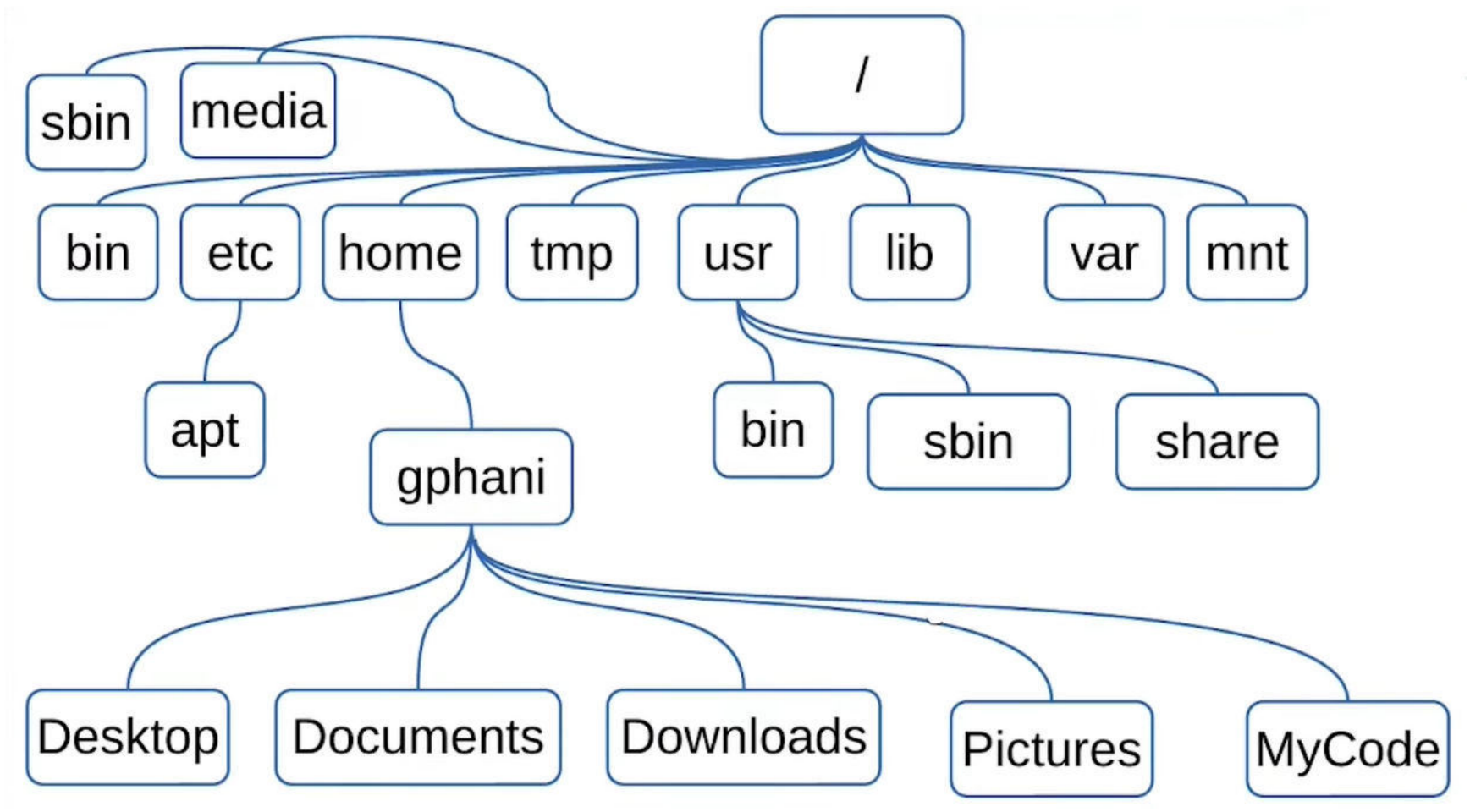
```
man <command-name>
```

- For example

  ```
  man ls
  ```

  gives us the manual for the command `ls`

Filesystem in Linux



Traversing the tree

`/` is the root of the file system

`/` is also the delimiter for sub-directories

- `.` is the current directory
- `..` is the parent directory

Path for traversal can be absolute or relative

To change directory

```
cd <location>
```

Examples

- `cd` without any arguments will take us to the home directory of the currently logged in user
- `cd <folder-name>` will change the move us into the `folder-name`
- `cd ..` will takes us to the parent of the current directory
- `cd /` will takes us to the root directory

What does this *directory* do?

- `/bin` → Essential command binaries
- `/boot` → Static files for the bootloader
- `/dev` → Device files
- `/etc` → Host specific system configuration
- `/lib` → Essential shared libraries and kernel modules
- `/media` → Mount points for removable devices
- `/mnt` → Mount points
- `/opt` → Add on application software packages
- `/run` → Data relevant to running processes
- `/sbin` → Essential system binaries
- `/srv` → Data for services
- `/tmp` → Temporary files
- `/usr` → Secondary hierarchy
- `/var` → Variable data

`/usr` hierarchy

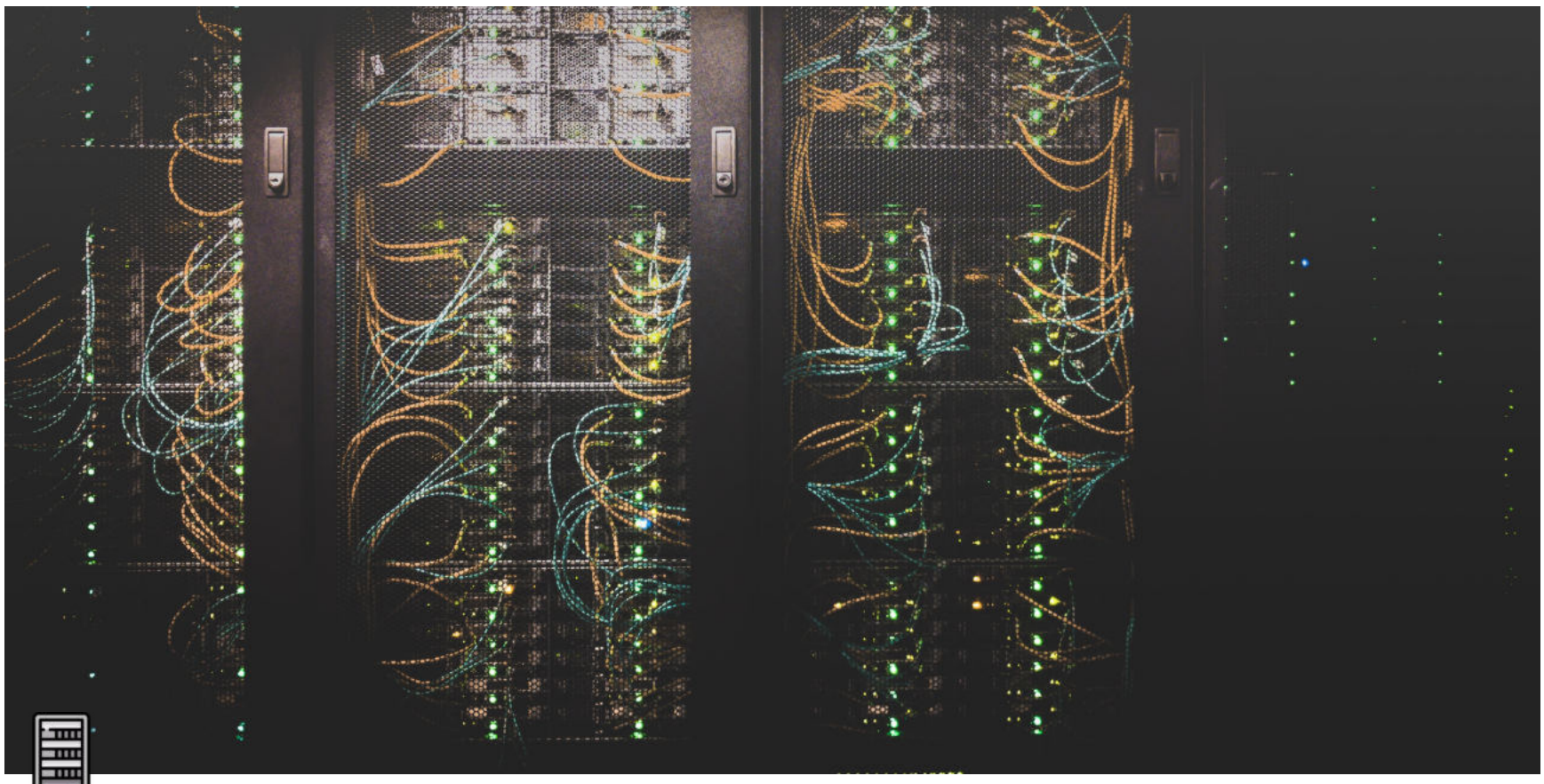
- `/usr/bin` → User commands
- `/usr/lib` → Libraries
- `/usr/local` → Local hierarchy
- `/usr/sbin` → Non-vital system binaries
- `/usr/share` → Architecture dependent data
- `/usr/include` → Header files included by C programs
- `/usr/src` → Source code

`/var` hierarchy

- `/var/cache` → Application cache data
- `/var/lib` → Variable state information
- `/var/local` → Variable data for `/usr/local`
- `/var/lock` → Lock files

- `/var/log` → Log files and directories
- `/var/run` → Data relevant to running processes
- `/var/tmp` → Temporary files preserved between reboots

	sharable	**unsharable**
static	/usr /opt	*/etc* */boot*
variable	*/var/mail*	/var/run /var/lock



Simple Commands in Linux - 1

Type	📒 Lecture
Date	@December 22, 2021
Lecture #	3
Lecture URL	https://youtu.be/DIpBEmRDHnw
Notion URL	https://21f1003586.notion.site/Simple-Commands-in-Linux-1-3e5b2c25e6d04a7499afcb89af041b20
Week #	1

Some basic commands

- `date` → Date and time

```
kashif@Zen:~/Desktop$ date
Wednesday 22 December 2021 12:05:01 PM IST
```

  - `date -R` → Gives the date in RFC5322 standard
- `cal` → Calendar of a month

```
kashif@Zen:~/Desktop$ cal
   December 2021
Su Mo Tu We Th Fr Sa
            1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

- `free` → Memory statistics

```
kashif@Zen:~/Desktop$ free
               total        used        free      shared  buff/cache   available
Mem:         4020444      589808     2725032       36244      705604     3170416
Swap:         945368           0      945368
```

  - `free -h` → Makes the output human readable
- `groups` → Groups to which the user belongs

```
kashif@Zen:~/Desktop$ groups
kashif adm cdrom sudo dip plugdev lpadmin lxd sambashare
```

idk what the junk is this

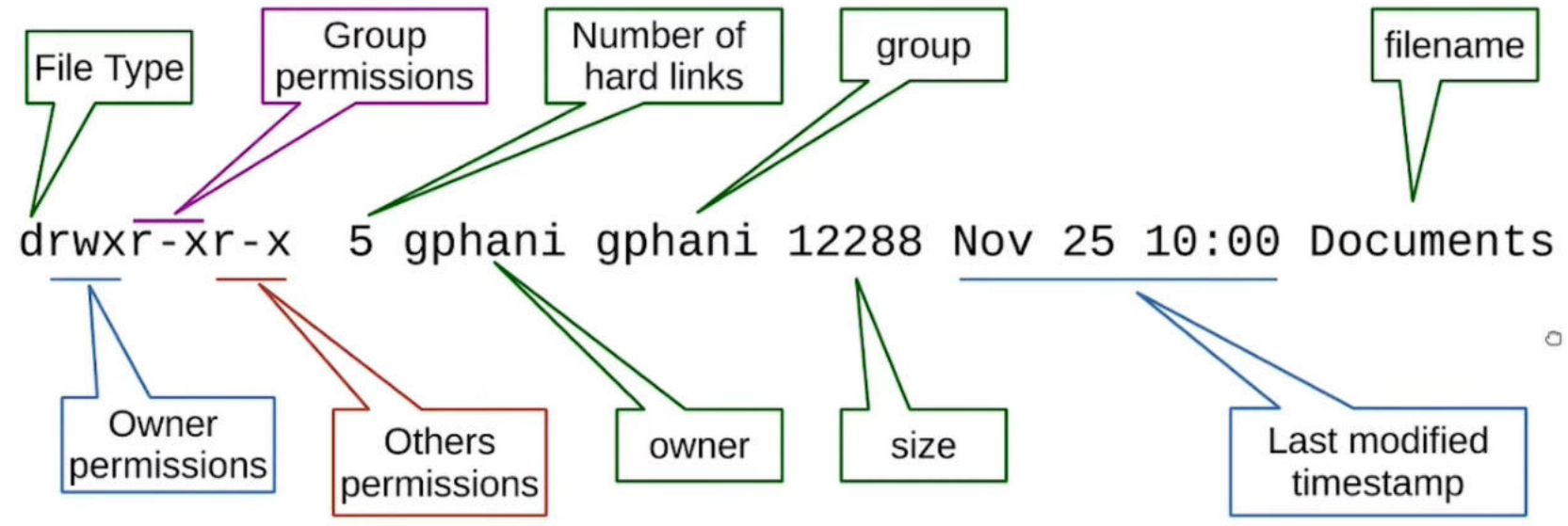
- `file` → What type of a file it is

```
kashif@Zen:~$ file .bashrc
.bashrc: ASCII text
```

- `cd -` → To visit the previous directory we were in

```
kashif@Zen:~$ cd -
/home/kashif/Desktop
kashif@Zen:~/Desktop$
```

Typical output of `ls -l`



File types

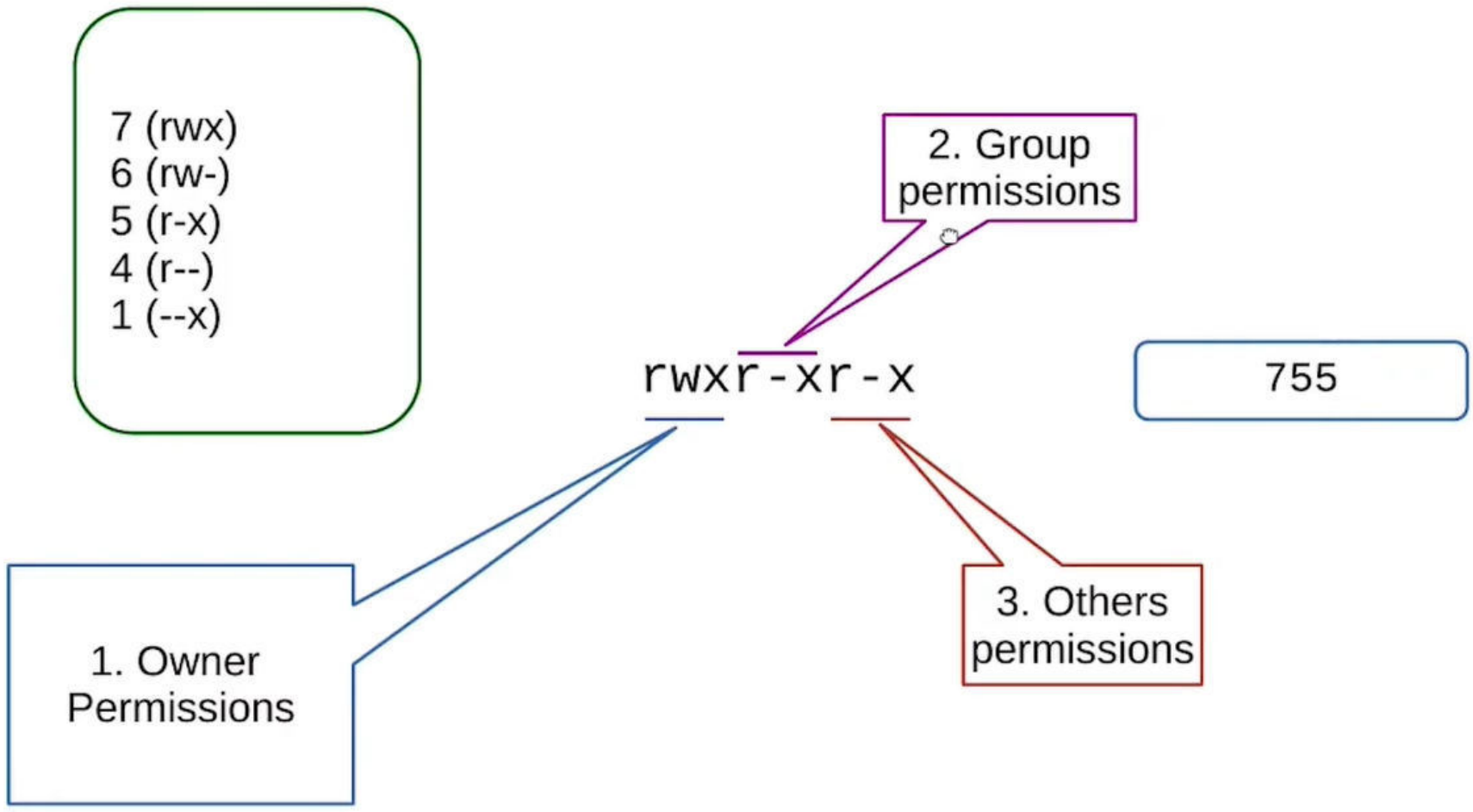
- `-` → Regular file
- `d` → Directory
- `l` → Symbolic link
- `c` → Character file
- `b` → Block file
- `s` → Socket file
- `p` → Named pipe

inode (Read: eye node)

`ls -l <name>`

- An entry in the filesystem table about the location in storage media

Permission string



To modify permissions

- Create a folder
  - `mkdir <folder-name>`
  - `chmod g-w <folder-name>` to remove the write permission from the group
  - Similarly, `chmod g-x <folder-name>` to remove the execute permission from the group
  - To add permission, `chmod g+w <folder-name>` to give write permission to the group
- So, a general structure of permission syntax is something like ...
  - `chmod <user-group><plus/minus><r/w/x> <folder-name/file-name>`
    - Where `<user-group>` are ...
      - `u` → User
      - `g` → Group
      - `o` → Others
    - `<plus/minus>` are ...
      - `-` → To remove permission
      - `+` → To add permission
    - `<r/w/x>` are ...
      - `r` → Read
      - `w` → Write
      - `x` → Execute
- We can also use numerical values for permissions
  - `chmod 700 <folder-name>` to give the `rwx` permission to user only

`touch` command

- Used to modify the timestamp of a file or folder
  - If a file does not exist, it will be created
- `touch <file-name>` to create a new file
  - `chmod 700 <file-name>` to give `rwx` permission to user only

`cp` command

- `cp <file-name> <new-name>` to copy a file to a new name
  - `cp <file-name> <new-path>` can be used to copy a file to a new path

`mv` command

- `mv <file-name> <new-path>` to move a file to a new path
  - `mv <file-name> <new-file-name>` can be used to rename a file

Also, use quotation marks if the file name includes a space

`rm` command

- `rm <file-name>` to remove a file
  - **IT WILL NOT ASK FOR YOUR CONFIRMATION**
    - Just straight up delete

    - This is the default behaviour
  - We can pass `-i` flag for the confirm remove prompt

Alias

- We can also set an alias for long commands, for example ...
  - `alias ll="ls -altrhF"`

Know current user

```
whoami
```

Read a text file, page-by-page

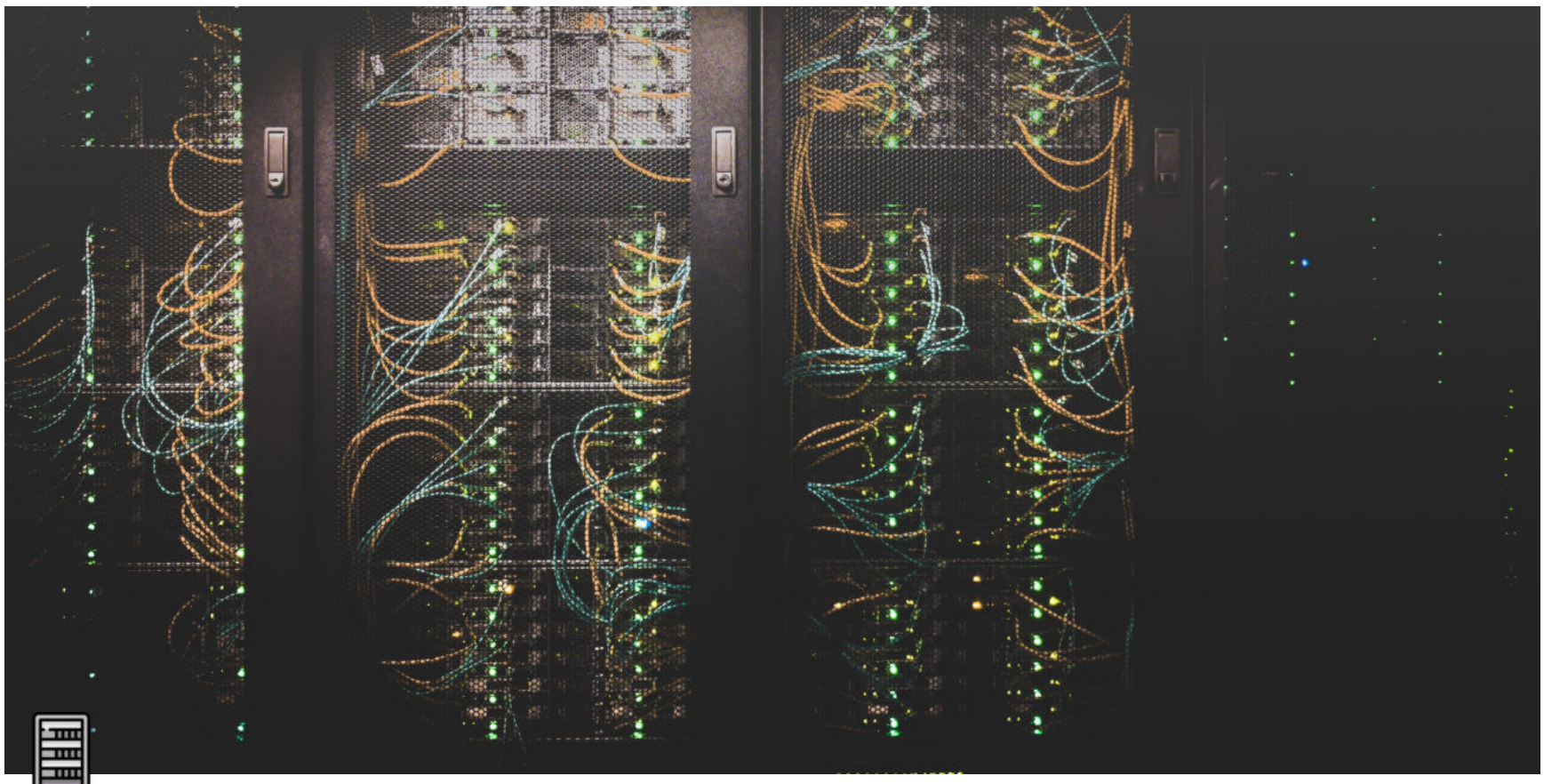
```
less <file-name>
```

To know the type of a file

```
file <file-name>
```

Some commands

- `chmod` → Change permissions of a file
- `touch` → Change modified timestamp of a file
- `cp` → Create a copy of a file
- `mv` → Rename/Move a file
- `mkdir` → Create a directory
- `rm` → Remove a file



Simple Commands in Linux - 2

Type	📒 Lecture
Date	@December 22, 2021
Lecture #	1
Lecture URL	https://youtu.be/lpe6AKk5GIk
Notion URL	https://21f1003586.notion.site/Simple-Commands-in-Linux-2-0cd2a51884c14be1a43ad24cd99c6692
Week #	2

`ls`

- Short and Long form of options
- Interpretation of directory as an argument
- Recursive listing
- Order of options on command line

  `ls -l` **will display the files & folders in the current directly, in a list**

```
kashif@Zen:~$ ls -l
total 40
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Desktop
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Documents
drwxr-xr-x 3 kashif kashif 4096 Dec 21 20:16 Downloads
drwxr-xr-x 2 kashif kashif 4096 Dec 22 12:35 level1
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Music
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Pictures
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Public
drwx------ 3 kashif kashif 4096 Dec 21 20:15 snap
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Templates
drwxr-xr-x 2 kashif kashif 4096 Dec 21 19:58 Videos
```

If we add the name of a folder after `ls -l`*, it will display the files & folders of that folder*

Commands to work with text files

- `less <file-name>` → To read text files, page-by-page
  - Press `q` to exit
- `cat <file-name>` → To read the entire text file, and dump them on the terminal window

    - Depending on the situation, `less` can be a better way to display the contents of a file on the terminal window rather than `cat`
- `more <file-name>` → Works similarly to `less`
    - View contents of the file page-by-page
- `head <file-name>` → Displays the first 10 lines of a file
    - Takes an optional flag, `-n` followed by an integer to display that number of lines
- `tail <file-name>` → Works the opposite of `head` command
- `wc <file-name>` → Displays the number of lines (or newlines), words and bytes in a file
    - Takes optional flags like `-l` to display only the # of lines
- `which <command>` → A command to locate the commands, *heh*

- `whatis <command>` → Gives a brief description of a command
- `apropos <keyword>` → Takes in the keyword and returns a list of commands that contain the keyword in their name or in their description

```
kashif@Zen:/etc$ apropos who
w (1)                - Show who is logged on and what they are doing.
who (1)              - show who is logged on
whoami (1)           - print effective userid
```

- `help` → Displays the reserved keywords
    - Can pass a command name as optional parameter and it'll display it's help
    - Works with `shell keyword`
- `info` → Open a file (which is similar to a webpage) that lists all the commands, and we can move the cursor to any command and press ***Enter*** to get the manual of that command
    - Use the `<` button to go back, how do you press the `<` button? `Shift + ,`
- `type` → Know the type of a command

```
kashif@Zen:/etc$ type which
which is /usr/bin/which
kashif@Zen:/etc$ type who
who is hashed (/usr/bin/who)
kashif@Zen:/etc$ type pwd
pwd is a shell builtin
kashif@Zen:/etc$ type ls
ls is aliased to `ls --color=auto'
kashif@Zen:/etc$ type type
type is a shell builtin
```

What are aliases?

Aliases are a nickname given to a command, which may want for our convenience

How to set an alias?

```
alias <name>='<your-command>'

# Example
alias ll='ls -al'
```

How to remove an alias?

```
unalias <name>

# Example
unalias ll
```

OK, How do I know all of my aliases?

```
# Type 'alias' in the bash shell
alias
```

Multiple arguments

- Second argument
- Interpretation of last argument
- Recursion is assumed for `mv` but not for `cp`

Links

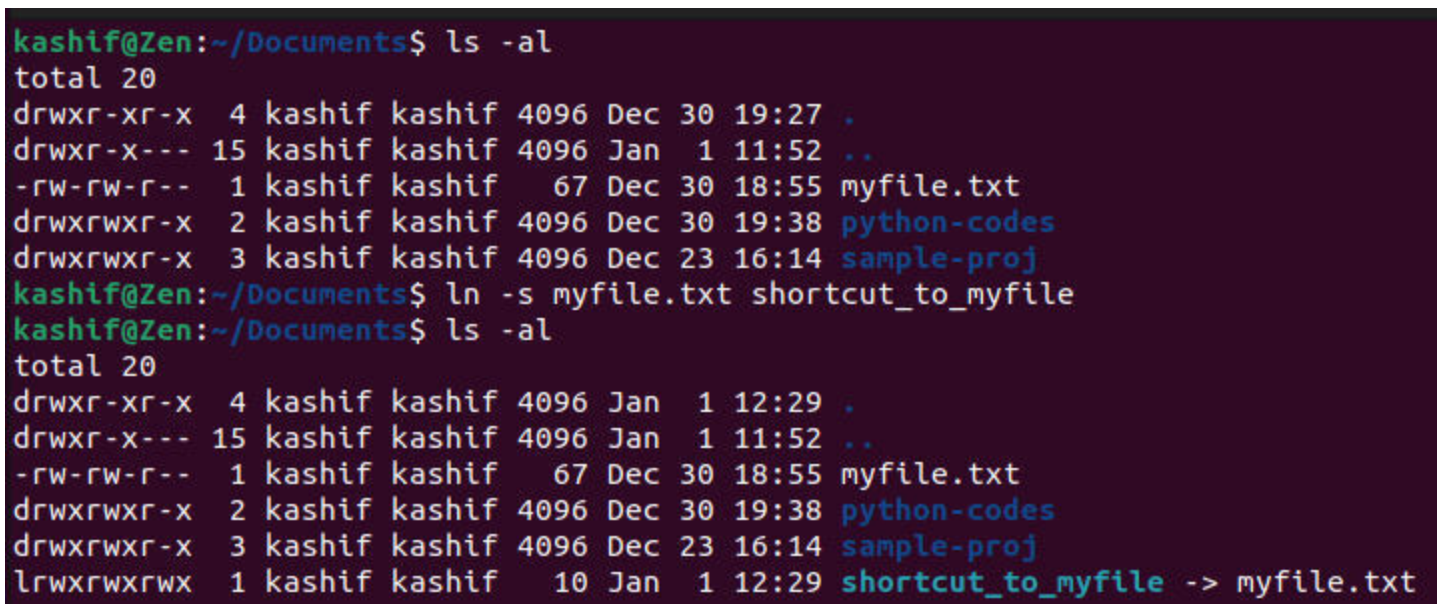
- Hard links
- Symbolic links
  - It is a bit analogous to desktop shortcuts in Windows

How to make a symbolic link?

- The command `ln` is used, with the `-s` flag

```
ln -s <source> <destination>
```

Example

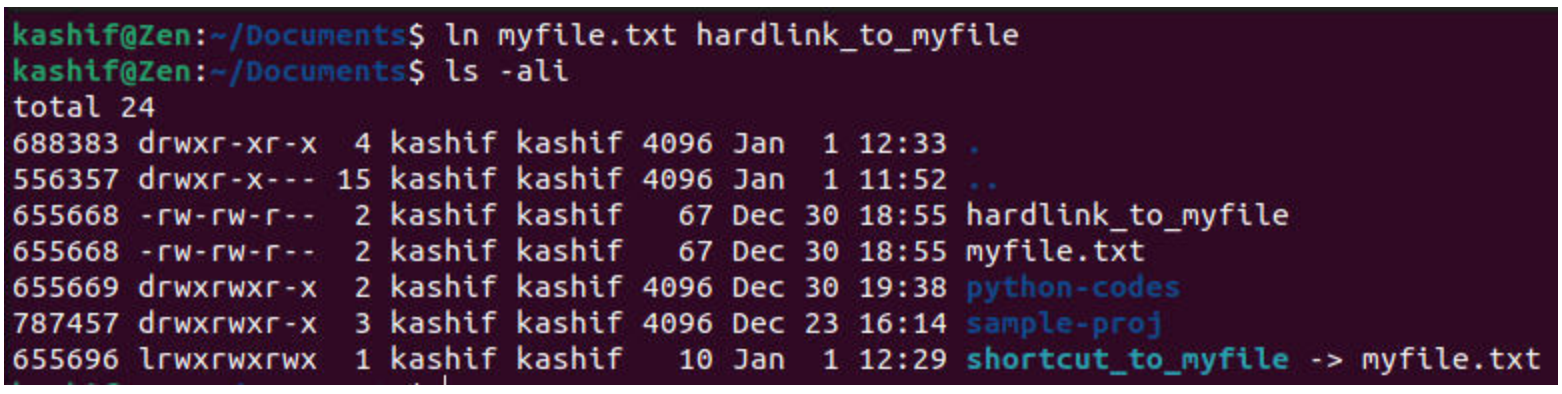


How to create a Hard link?

Just don't pass the `-s` flag to the `ln` command

```
ln <source> <destination>
```

Example



Notice the `inode` number for the hardlink is the same as the file

Size of files

- `ls -s` lists the files with file sizes

- `stat <file-name>` lists various other details as well as the file size
- `du <file-name>` shows the size of the file
  - Takes an optional `-h` flag to format the output in human readable form

In-memory filesystem

- `/proc`
- `/sys`

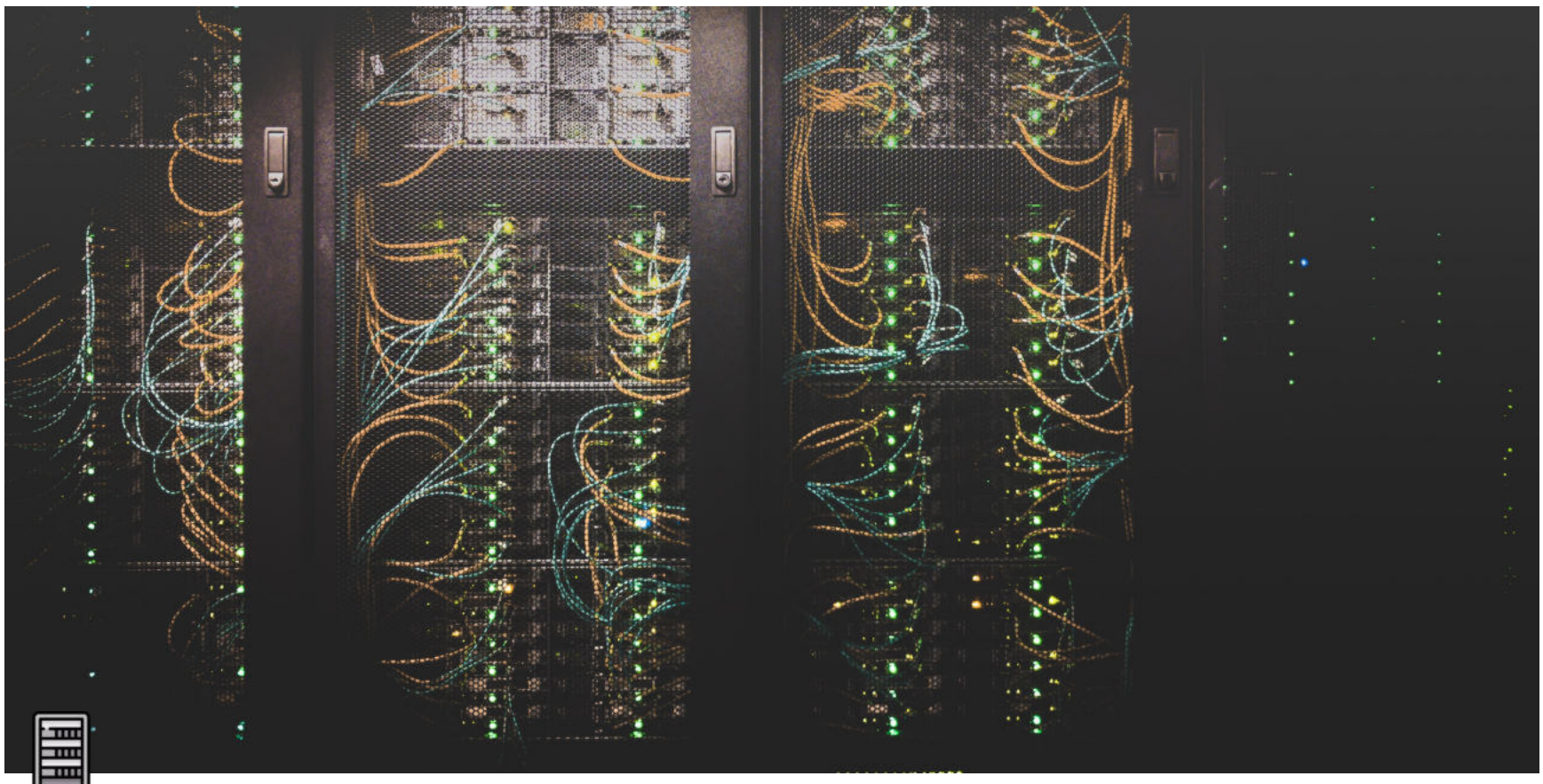
These directories are not stored on your HDD/SSD

They are located in your system memory

These contain the info about the system, so *no* `rm -rf` *-ing around as* `root`

A few commands

- `free` → To display the system memory stats, like used memory, free memory
  - Takes an optional `-h` flag to format the output in human readable form
- `df -h` → To know about the partitions, in human readable form



$hell variables

Type	📒 Lecture
Date	@December 22, 2021
Lecture #	2
Lecture URL	https://youtu.be/pPRge8Yxbso
Notion URL	https://21f1003586.notion.site/hell-variables-7c2117f90cee4aa7aec3ee410038f32a
Week #	2

`echo`

`echo` command prints a string or a environment variable

Example:

```
echo $HOME
echo Hello, World
```

Frequently used shell variables

- `$USERNAME`
- `$HOME`
- `$HOSTNAME`
- `$PWD`
- `$PATH`

Special shell variables

- `$0` → Name of the shell
- `$$` → Process ID of the shell
- `$?` → Return code of previously run program
- `$-` → Flags set in the bash shell

Process control

- Use of `&` to run a job in the background
- `fg`

- `coproc`
- `jobs`
- `top`
- `kill`

Program exit codes

- 0 → success
- 1 → failure
- 2 → misuse
- 126 → command cannot be executed
- 127 → command not found
- 130 → processes killed using `Ctrl + C`
- 137 → processes killed using `kill -9 <pid>`

Flags set in `bash`

- `h` → locate and hash commands
- `B` → brace expansion enabled
- `i` → interactive mode
- `m` → job control enabled
- `H` → ! style history substitution enabled
- `s` → commands are read from stdin
- `c` → commands are read from arguments

`echo`

```
echo Hello World
echo "How do you do?"
```

it's a start

We can enclose the string in quotes as well

Make sure to match the quotes properly

Some common `echo` commands

```
echo $USERNAME
echo $USER
echo $PWD
echo $HOME
echo $HOSTNAME
echo $PATH
```

We can enclose the shell variable (marked as $) in double quotes, and it'll replace the value

However, if we use single quote around the shell variable, the shell variable is printed as it is

We can also escape the shell variable by using a backslash in front of it

Example → `echo "User is \$USERNAME"` will display `User is $USERNAME`

To view all the shell variables

```
printenv
env
```

`printenv` prints all or part of the environment

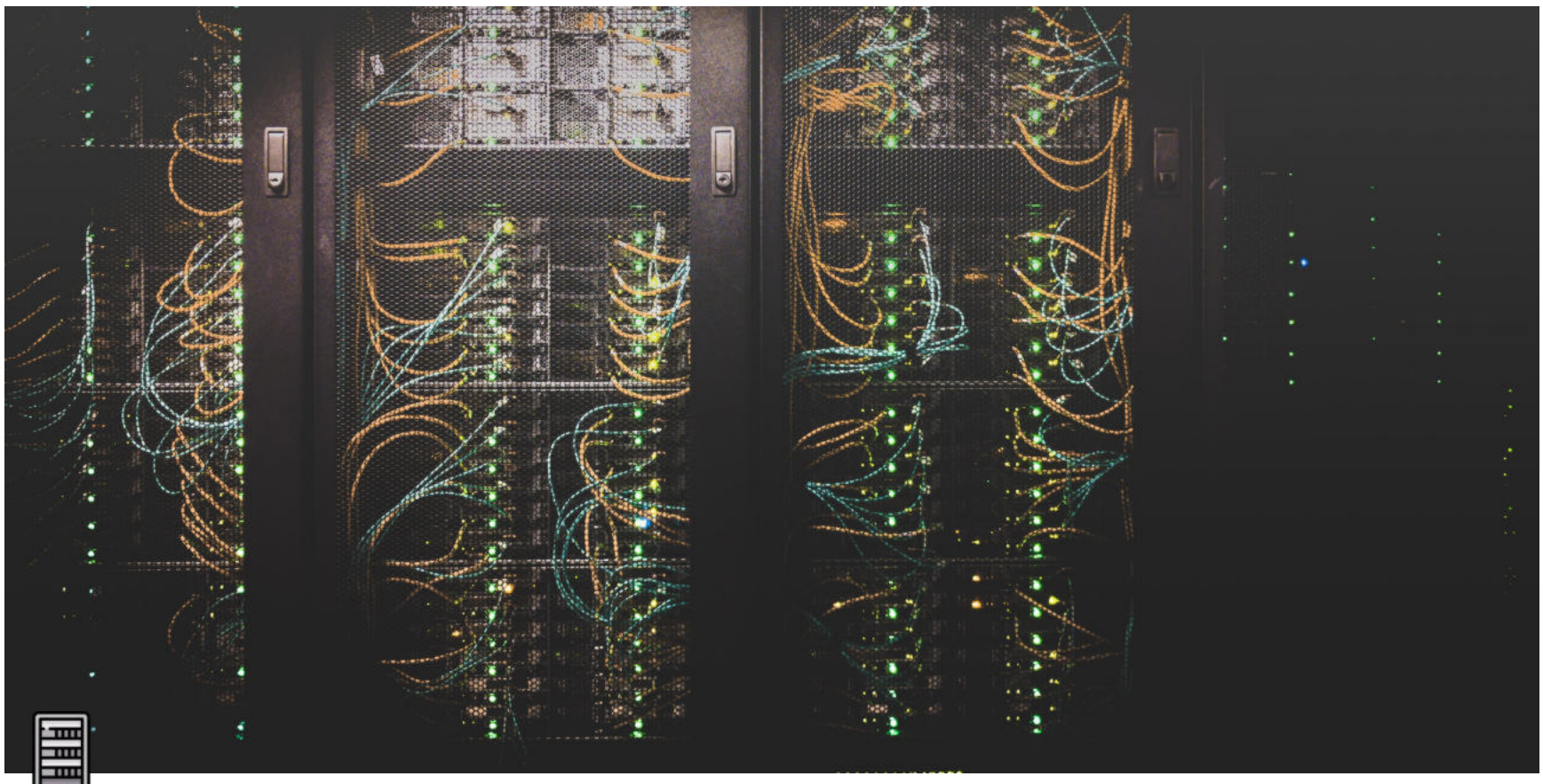
```
printenv HOME
```

`set`

`set` allows you to change the values of shell options and set the positional parameters, or to display the names and values of shell variables. (Source)

To escape any aliases set already

```
\<alias>

# Example
\ls
\date
```



Linux Process Management

Type	📒 Lecture
Date	@December 22, 2021
Lecture #	3
Lecture URL	https://youtu.be/2aThmDRvSWU
Notion URL	https://21f1003586.notion.site/Linux-Process-Management-d73af14458004045b281f5995be7cf32
Week #	2

`sleep` command

- It is a delay for a specified amount of time
  - The time is specified in seconds

Example

```
# The following command will make the prompt sleep for 5 seconds
sleep 5
```

Coprocess

- The shell command → `coproc`
- A Coprocess is executed asynchronously in a subshell
  - In simple words, this command is used to run a process without actually losing control of the prompt
  - As we don’t lose prompt access, we can execute other commands

Usage

```
coproc sleep 20
```

When the above mentioned process is completed, the prompt will output a message saying `Done`

We can run the `ps` command to view the running status of the aforementioned command

`kill`

- This command is used to kill an already running process

- `Ctrl + C` *is also quite handy*

Usage

```
kill -9 <PID>
```

How do I know the PID?

`ps` command, try `ps --forest` instead

`&` sign to put a process in the background

- We can also put the `&` (ampersand) sign at the end of a command to put that process in the background

Usage

```
sleep 30 &
```

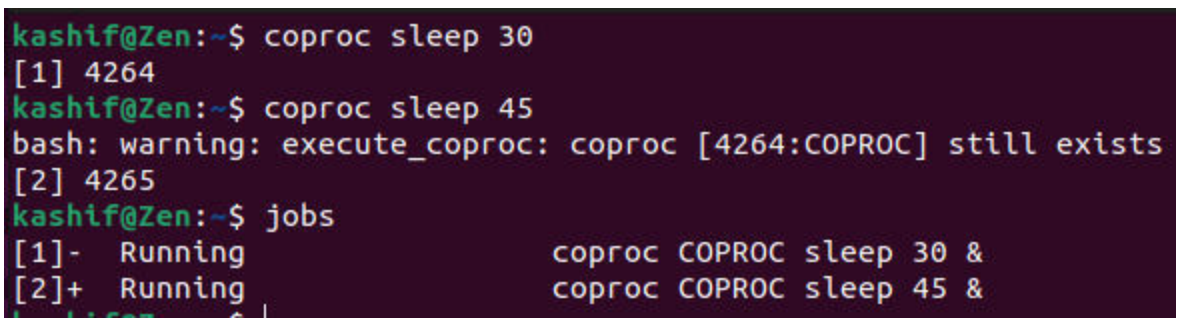
Alright, how do I bring it to the foreground?

```
fg
```

`jobs`

List the commands that are running in the background

Example



`top` command

It's like your Windows Task Manager, but in the Linux terminal

To exit out of `top`*, press* `Q` *or* `Ctrl + C`

Pressing `Ctrl + Z` while `top` is open suspends the process, then you can do your work and come back to it by using the `fg` command

`jobs` command will show the top process

Well, `Ctrl + Z` ***will suspend any running process***

$-

```
echo $-
```

The output of this shell variable would tell us about the bash shell we are currently using

To know what the output means, type `man bash` and match the flags

Launch a bash shell which is not interactive

```
bash -c "echo \$-"
```

```
bash -c "echo \$-; ps --forest;"
```

To know the PID of the new bash shell we launcher

```
bash -c "echo \$$; echo \$-; ps --forest;"
```

`history`

The following command displays out all the commands that have been run on the current shell, chronologically

```
history
```

The following command will run the `n`-th command that has been run, make sure to put `n` from the list of commands output from the above command

`n` is integer

```
!n
```

The following command will run the previous command that was executed on the bash shell

```
!!
```

Brace Expansion

So the output of `echo $-` had a `B` flag, it refers to Brace Expansion

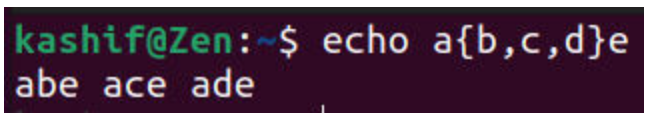
is this a JJK reference? ブレース展開

So, What is a Brace Expansion?

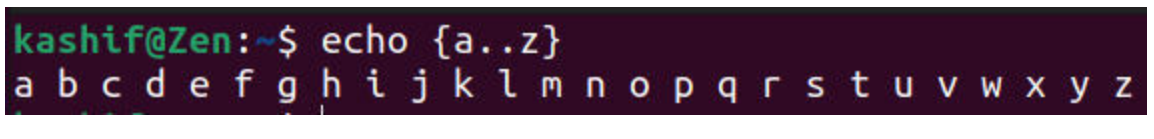
- Brace expansion is a mechanism by which arbitrary strings may be generated. Source

Example

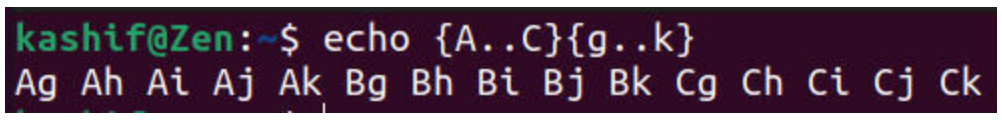
```
echo a{b,c,d}e
```




```
echo {a..z}
```




```
echo {A..C}{g..k}
```




```
# The * is a wildcard character, it expands to all the files in the current dir
echo *
```

Similarly ...

```
# The following command will return all the files folders that start with D
echo D*
```

Exit codes

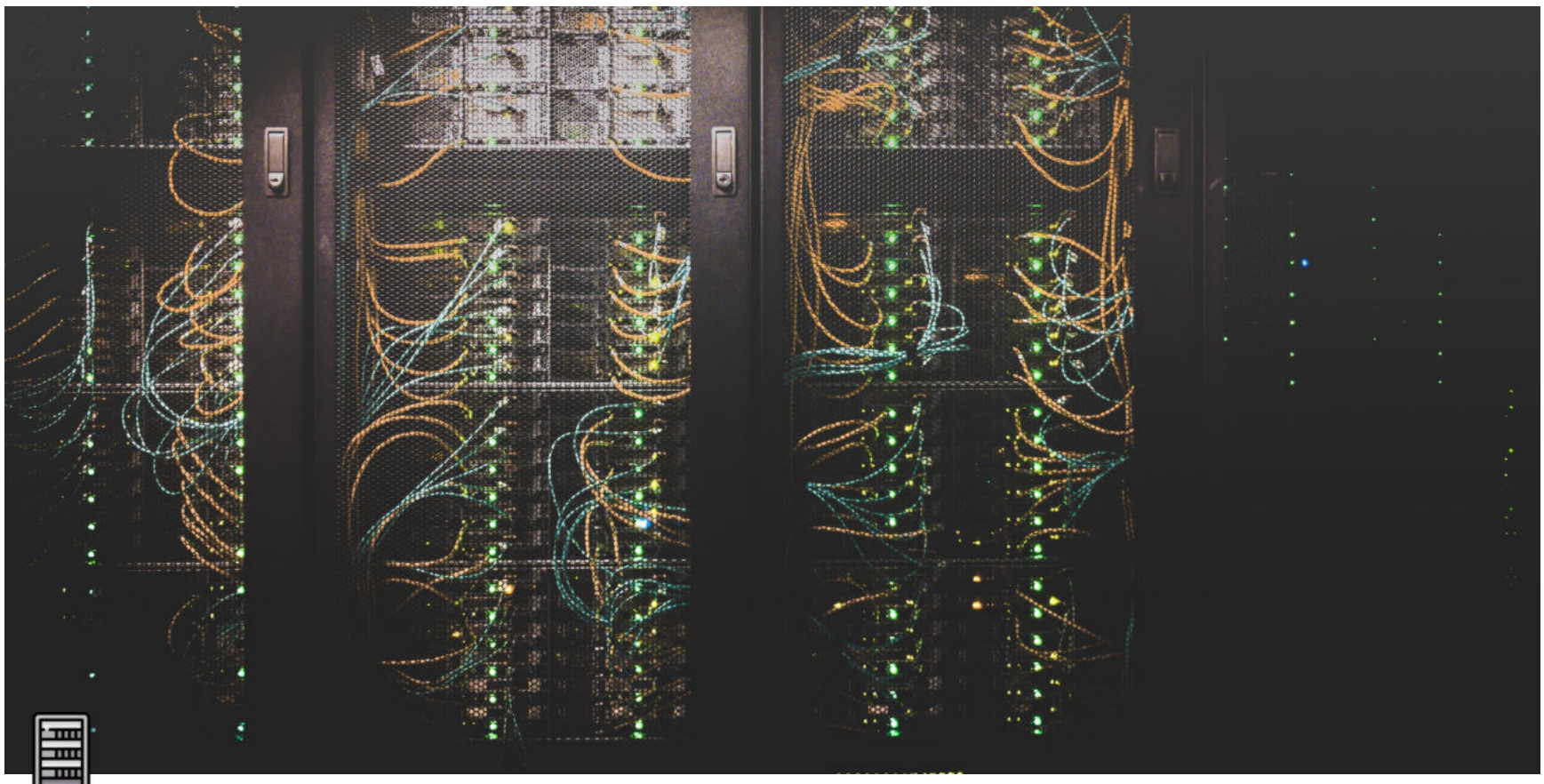
- `0` → All good, no error

- `1` → General errors, misc. errors
- `2` → Misuse of shell built-in
- `126` → Command invoked cannot execute
- `127` → Command not found
- `128` → Invalid argument to exit
  - `exit` takes int from `0 - 255`
- `128+n` → Fatal error signal "n"
  - When a process running in another place (like in another shell) was killed from somewhere else (killed from not the same shell)
  - `kill -9 $PPID` → `$?` returns 137 $(127 + 9)$
- `130` → Script terminated by `Ctrl + C`
- `255*` → Exit status out of range

If the exit code is out of range, bash takes the modulo of the exit code w.r.t. 256

Example

```
# Create a new bash subshell and exit with the status code 300
bash -c "exit 300"

# Check the exit code
echo $?

# The output will be ...
44
```



Combining commands and files

Type	📒 Lecture
Date	@January 9, 2022
Lecture #	1
Lecture URL	https://youtu.be/Lcx9UsS7y8Y
Notion URL	https://21f1003586.notion.site/Combining-commands-and-files-2476c1091a704743840ae8b76ab078c9
Week #	3

Executing multiple commands

- `command1; command2; command3`
  - Each command will be executed one after the other
- `command1 && command2 && command3`
  - This works as a logical AND
  - The subsequent commands after `command-n` will not run if the previous command resulted in an error
- `command1 || command2 || command3`
  - This works as a logical OR
  - The subsequent commands after `command-n` will not run if the previous command resulted in a success

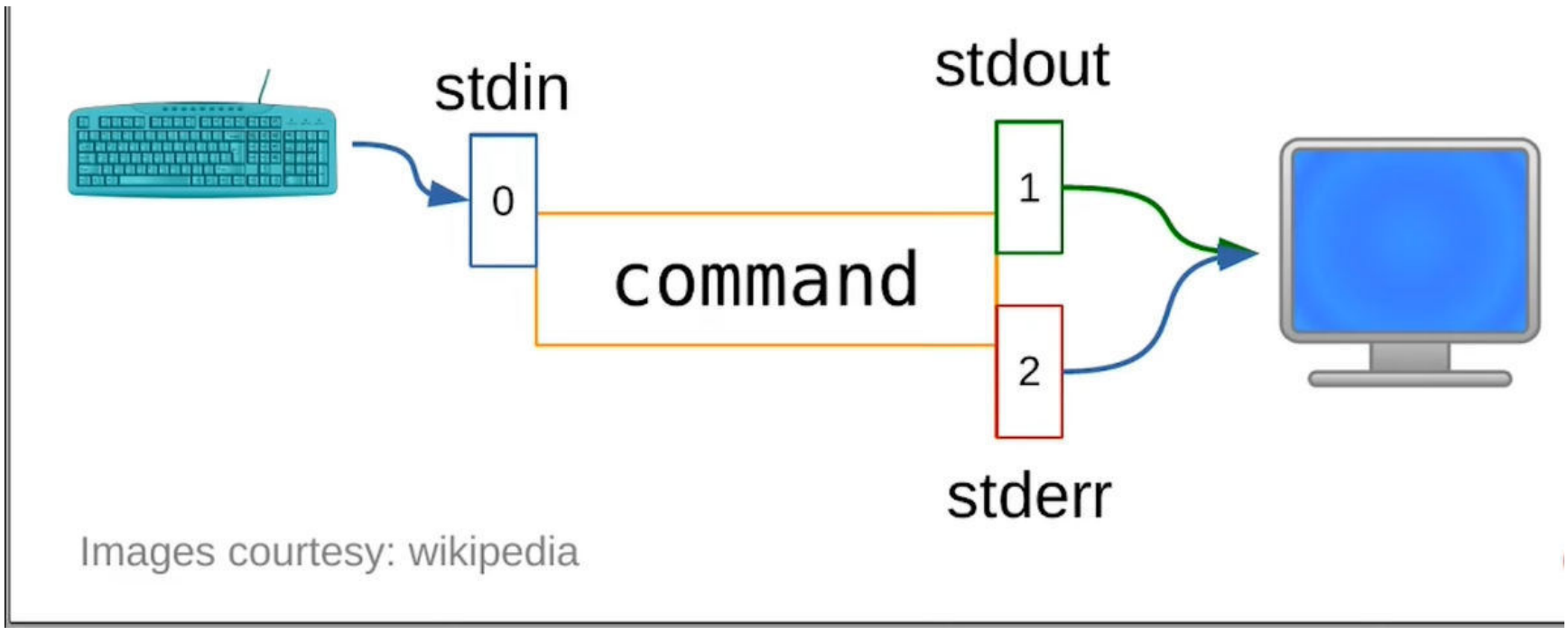
`(<command>)`

We can run any command enclosed within parentheses to execute them in a subshell, and returned back the result

We can execute a subshell within a subshell too

```
kashif@Zen:~$ echo $BASH_SUBSHELL
0
kashif@Zen:~$ (echo $BASH_SUBSHELL)
1
kashif@Zen:~$ (echo $BASH_SUBSHELL; (echo $BASH_SUBSHELL))
1
2
```

File descriptors

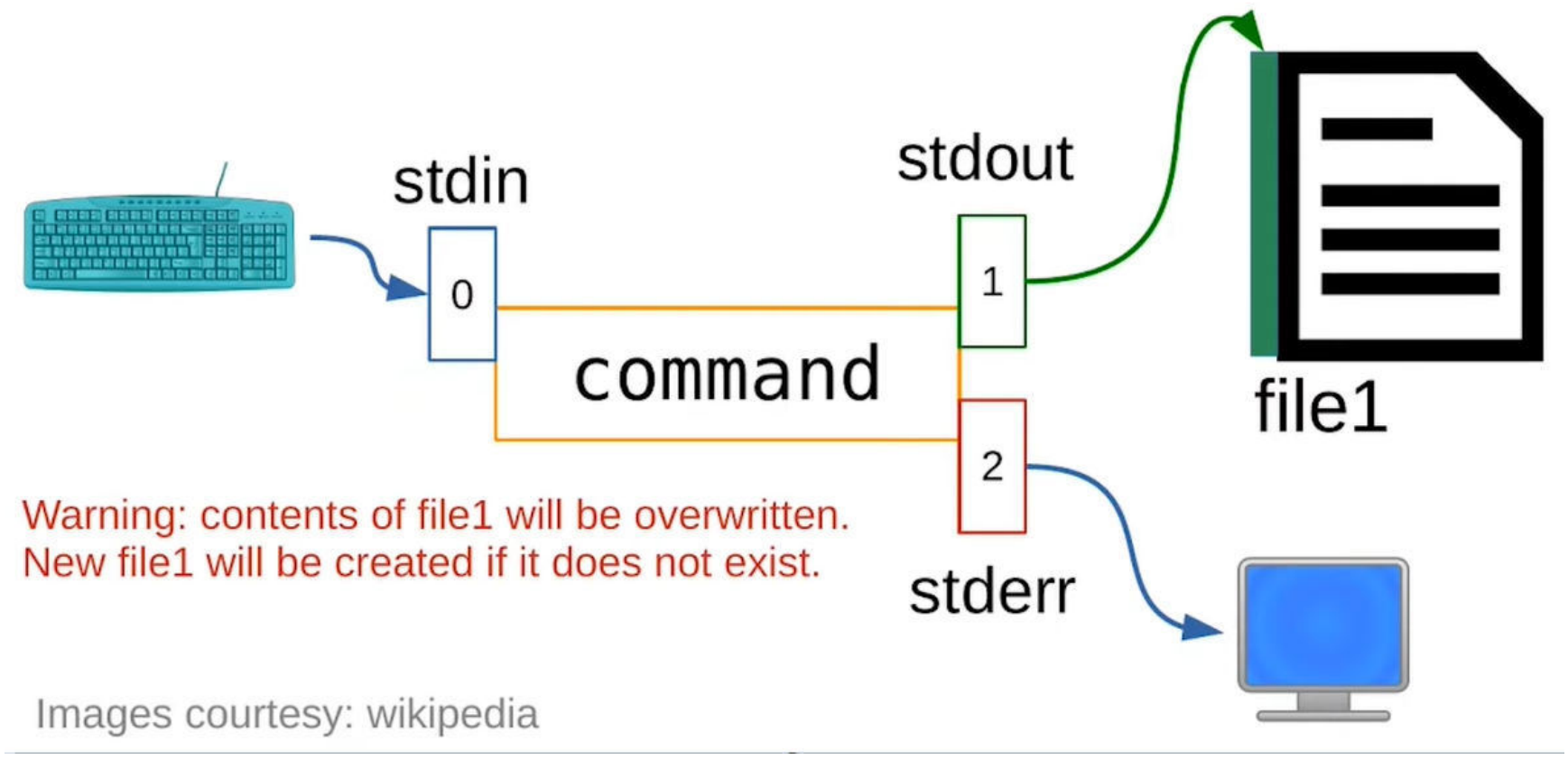


Every command in Linux has 3 file descriptors

- `stdin` (0)
  - It is a pointer to a stream that is coming from the keyboard (or the user input)
- `stdout` (1)
  - Points to the screen where the output is made
- `stderr` (2)
  - Points to the screen where the output is made

`command > file1`

- The output of the `command` should be written to `file1`



Create a file using `cat` command

`cat > filename`

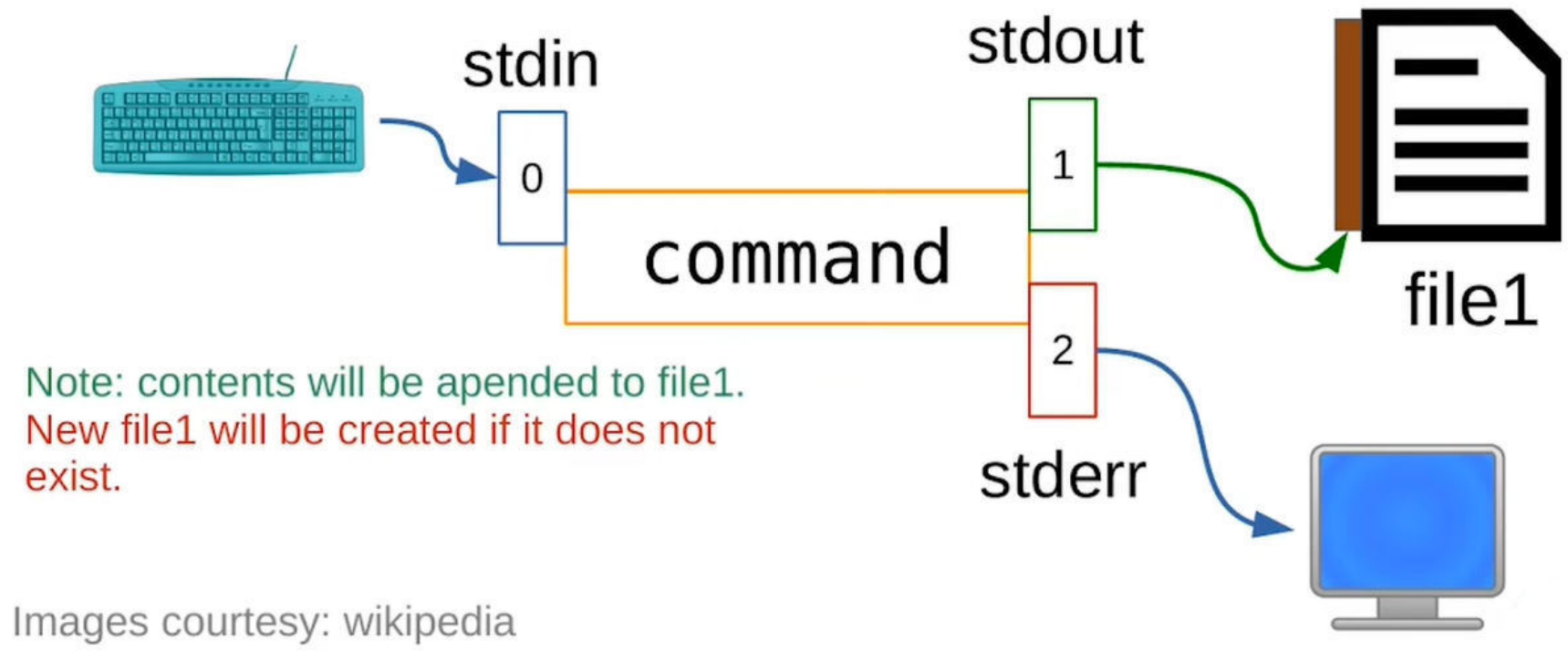
When we type this command, the `cat` command is supposed to receive the input from a file that is listed in the command line, but instead, we left that intentionally blank

So, the `cat` command, instead, reads the content from the `stdin`, i.e. the keyboard

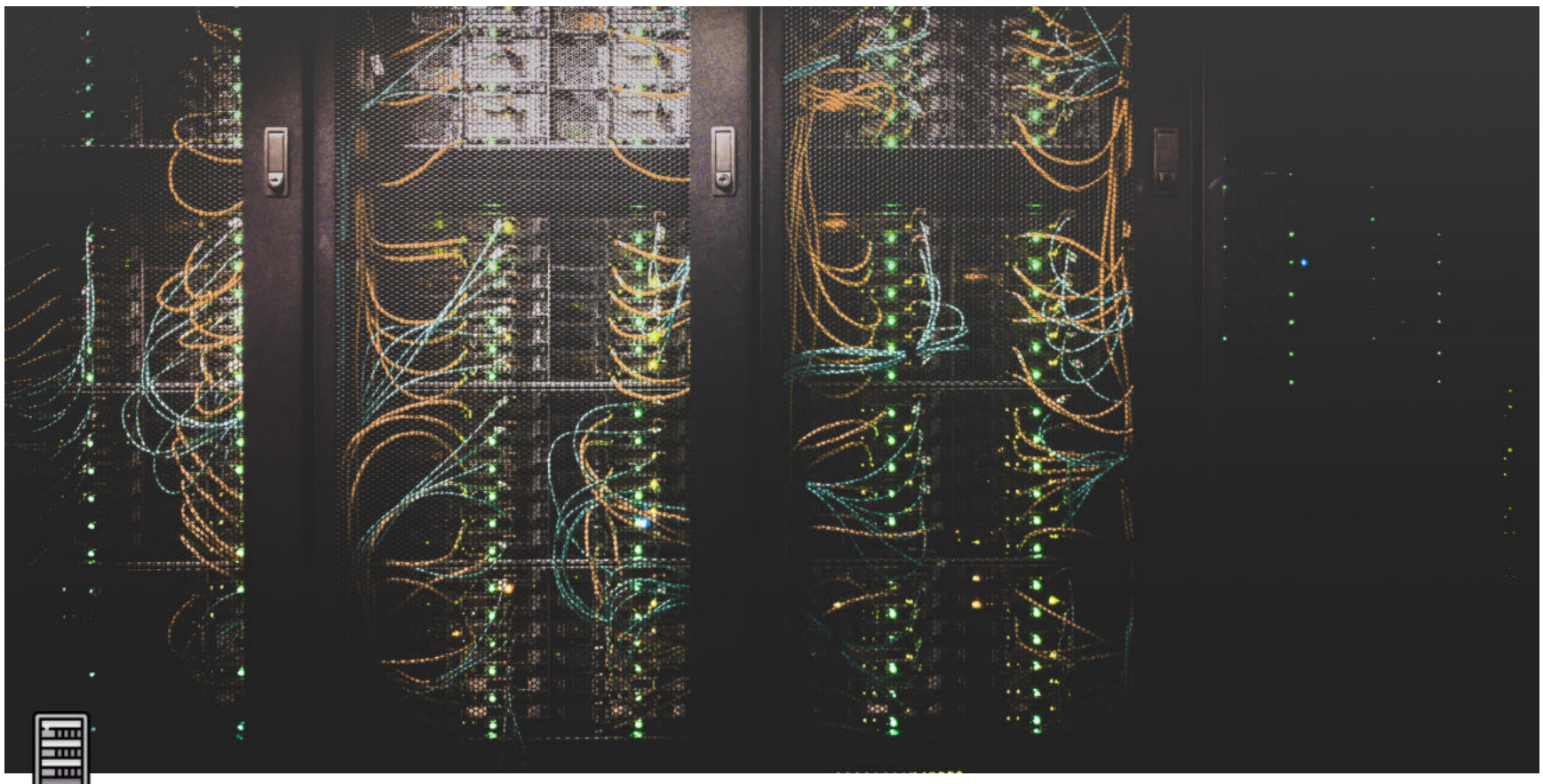
To exit, press `Ctrl + D`

`command >> file1`

- The output of `command` will be appended to `file1`



Similarly, we can use `>>` instead of `>` while creating a new file using the `cat` command

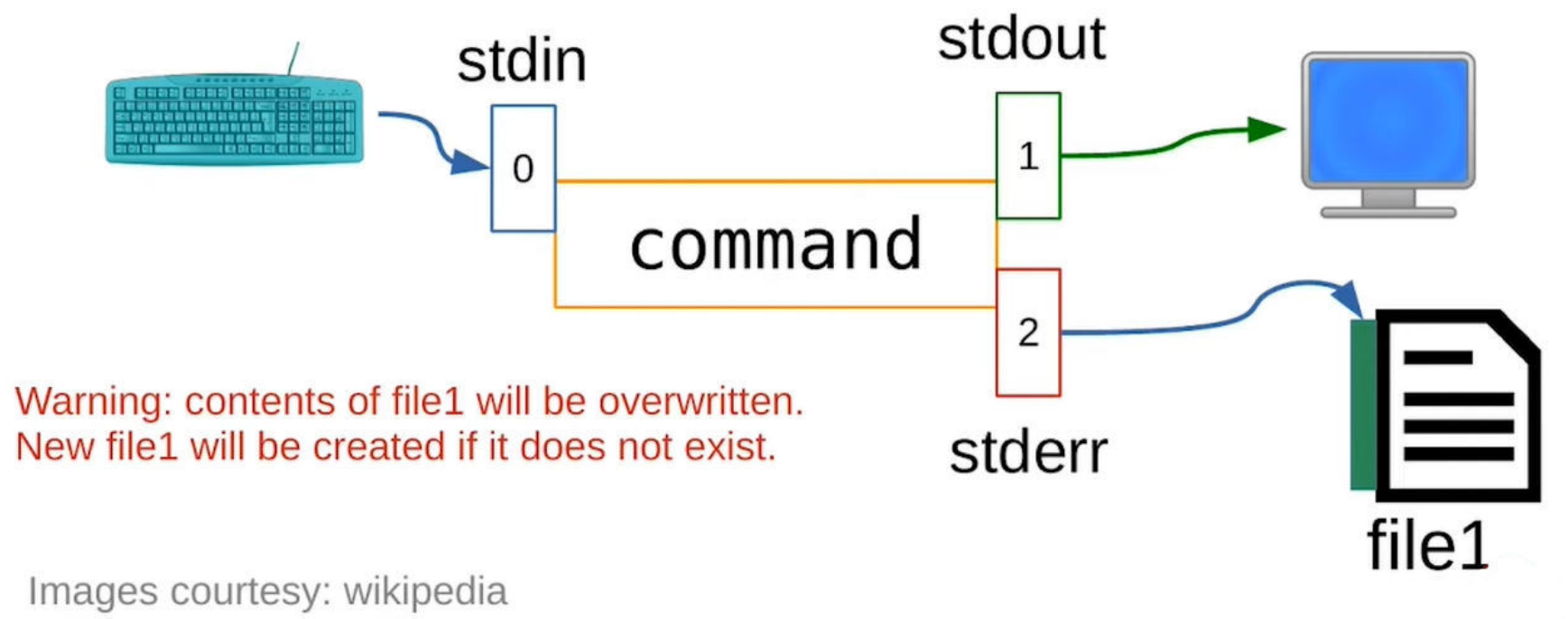


Redirections

Type	📒 Lecture
Date	@January 9, 2022
Lecture #	2
Lecture URL	https://youtu.be/BBh69kH_G_Y
Notion URL	https://21f1003586.notion.site/Redirections-734673f36f21448f99de25ccb092c8d4
Week #	3

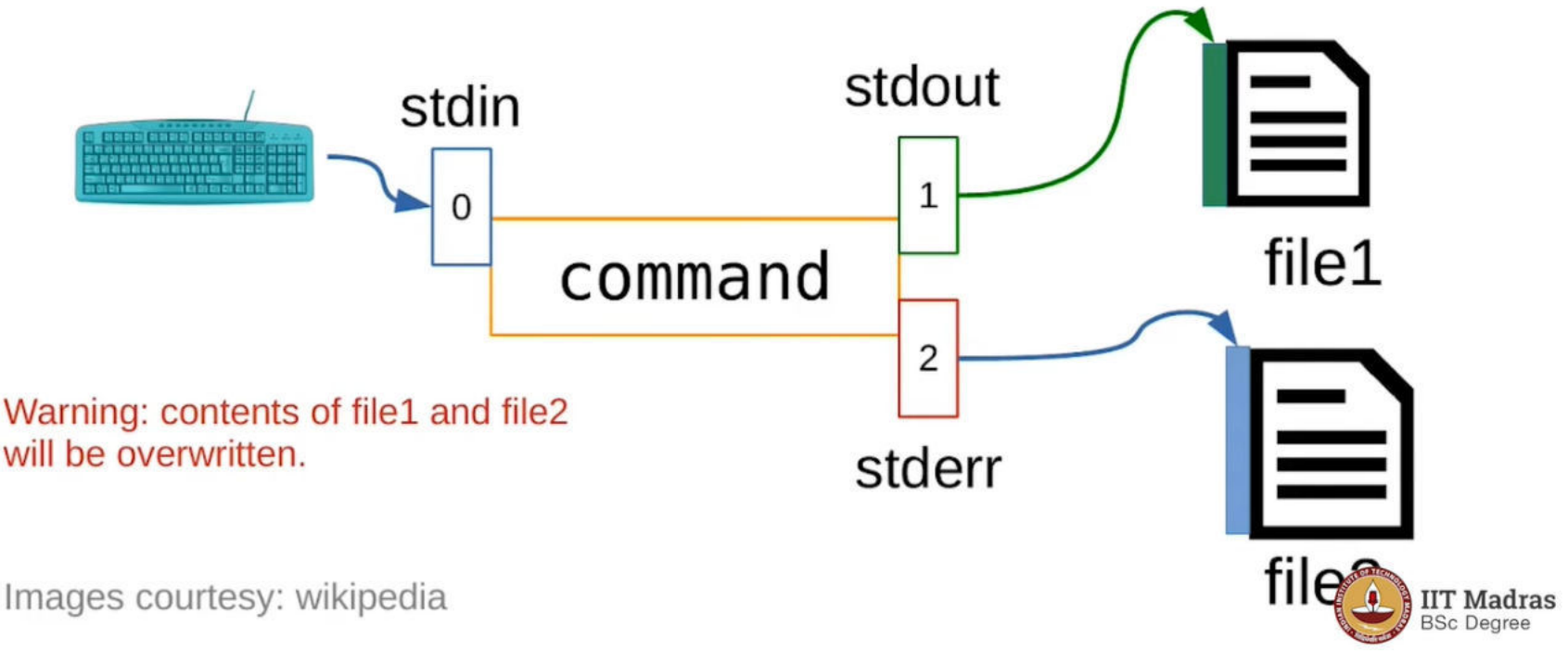
`command 2> file1`

- Redirect the output of the `command` to `stdout`, which is the display in this case
- Redirect the error of the `command` to `file1`



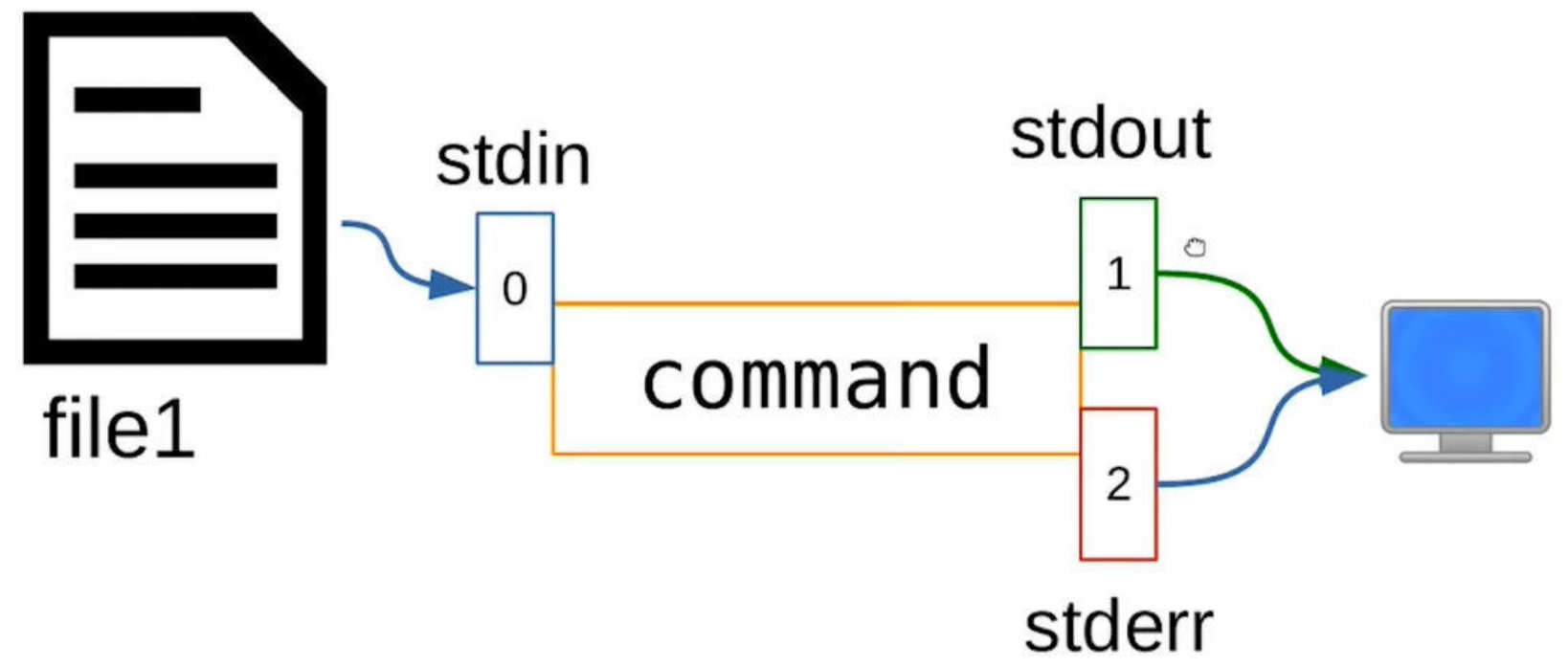
`command > file1 2> file2`

- Redirect the output of the `command` to the `stdout`, i.e. `file1`
- Redirect the error of the `command` to `stderr`, i.e. `file2`



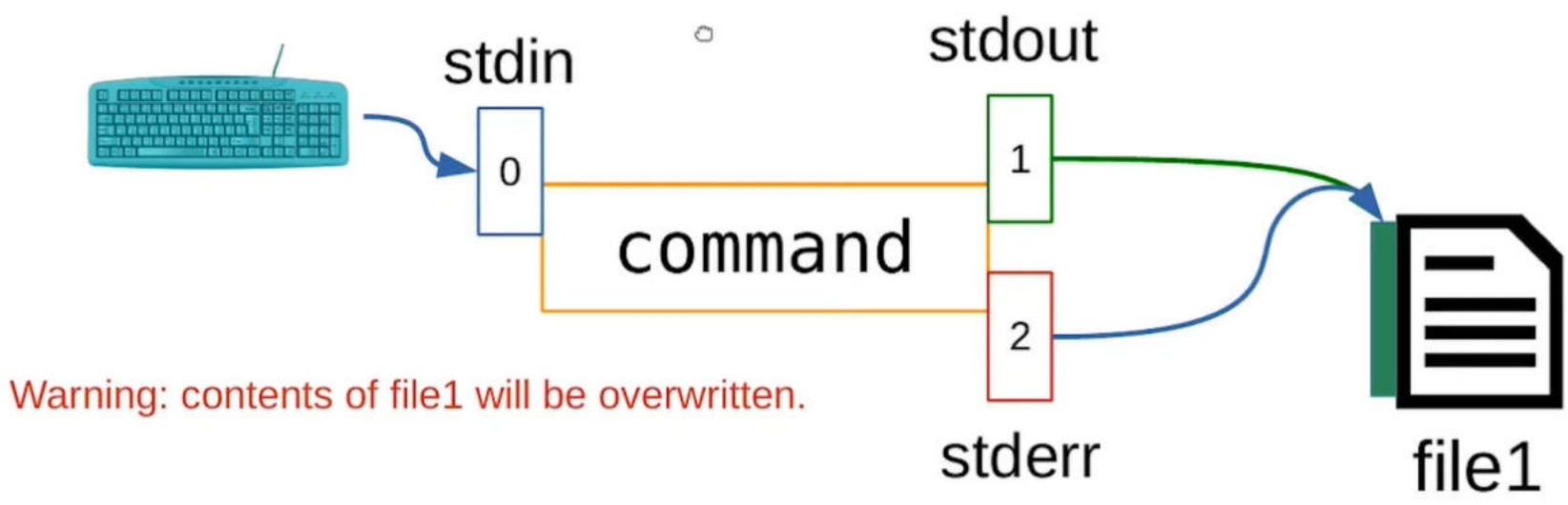
`command < file1`

- Any `command` which takes input from the keyboard, now takes input from `file1`



`command > file1 2>&1`

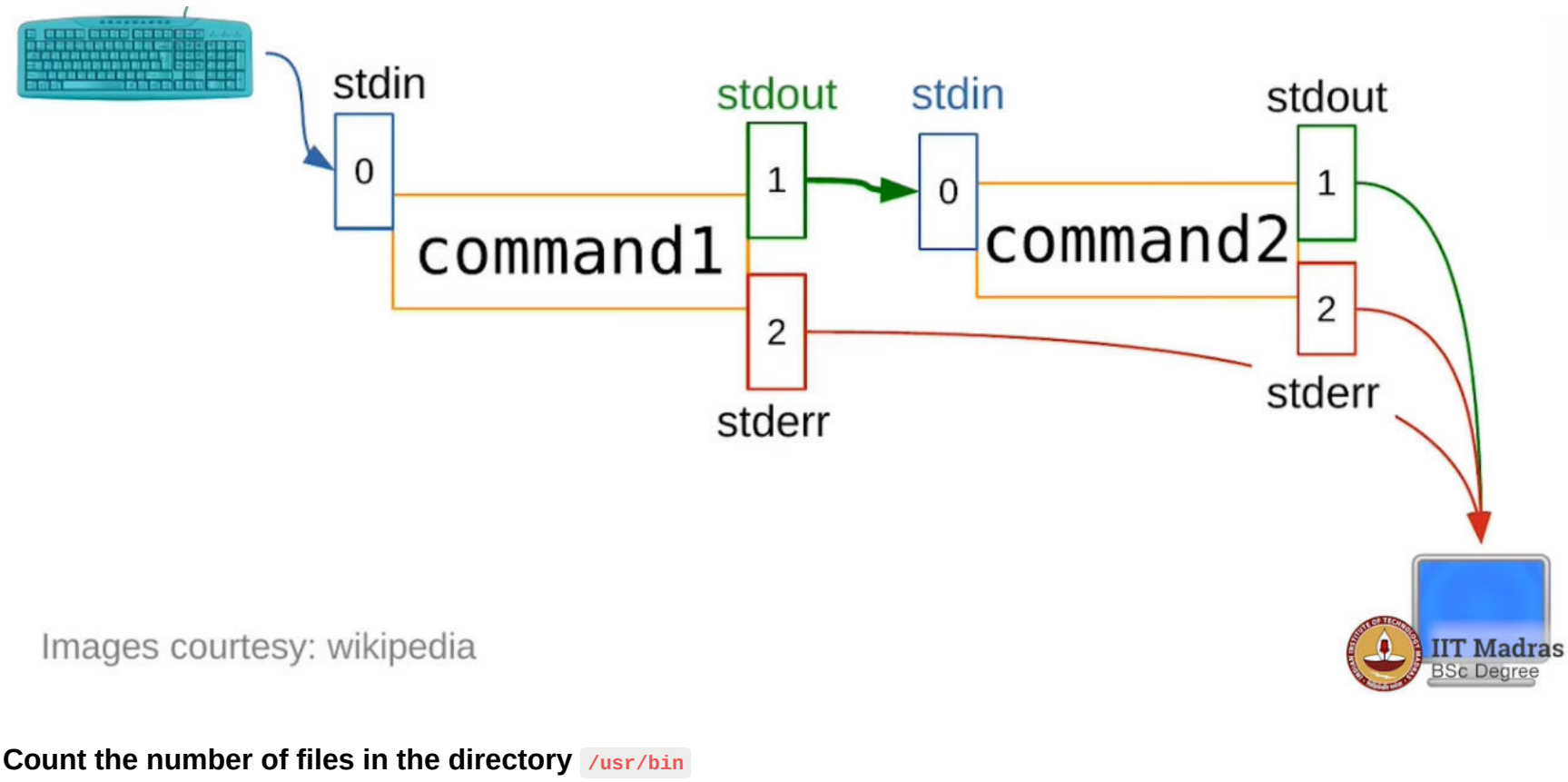
- The output of `command` is written to `file1`
- The error is redirected to stream 1, which is `stdout`



`command1 | command2` → **pipe operator** `|`

- The output of `command1` is sent to `command2` as input

- By default, the `stderr` will output to the display



Images courtesy: wikipedia

Count the number of files in the directory `/usr/bin`

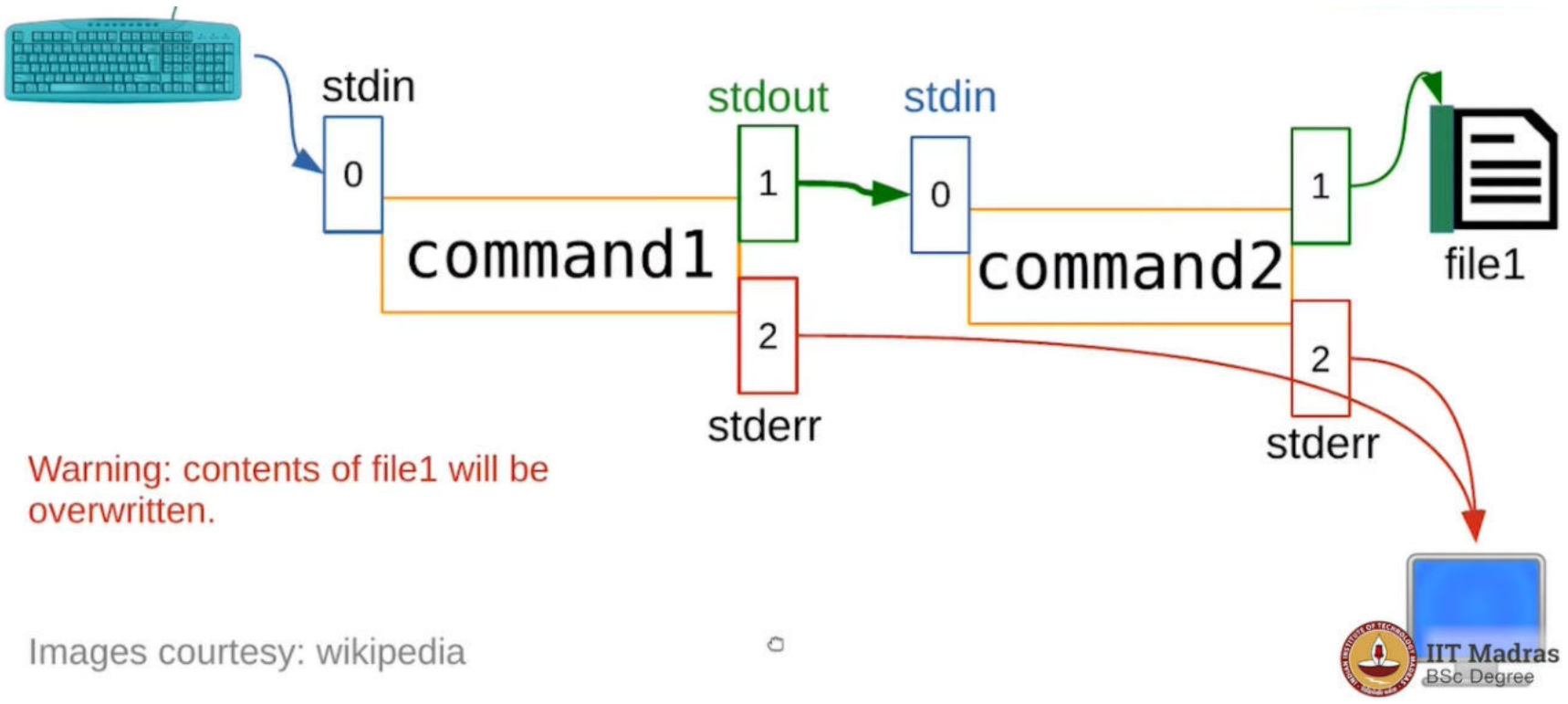
```
kashif@Zen:~$ ls /usr/bin/ | wc -l
1603
```

List the files of `/usr/bin` directory, but use the `less` command to scroll at ease

```
ls /usr/bin | less
```

`command1 | command2 > file1`

- The `stdout` of `command1` is mapped to `stdin` of `command2`
- The `stdout` of `command2` is written to `file1`
- The `stderr` is output to the display



Images courtesy: wikipedia

`/dev/null`

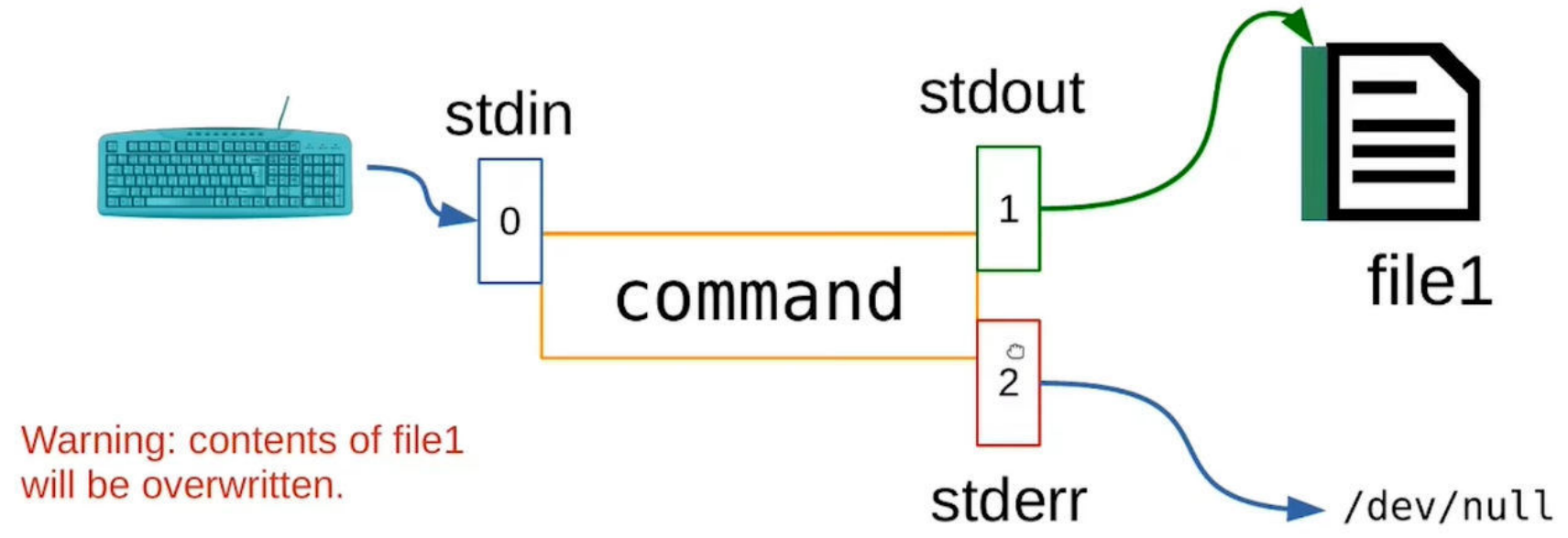
- A sink for output to be discarded
- Use → silent and clean scripts

So, a typical usage looks like ...

`command > file1 2> /dev/null`

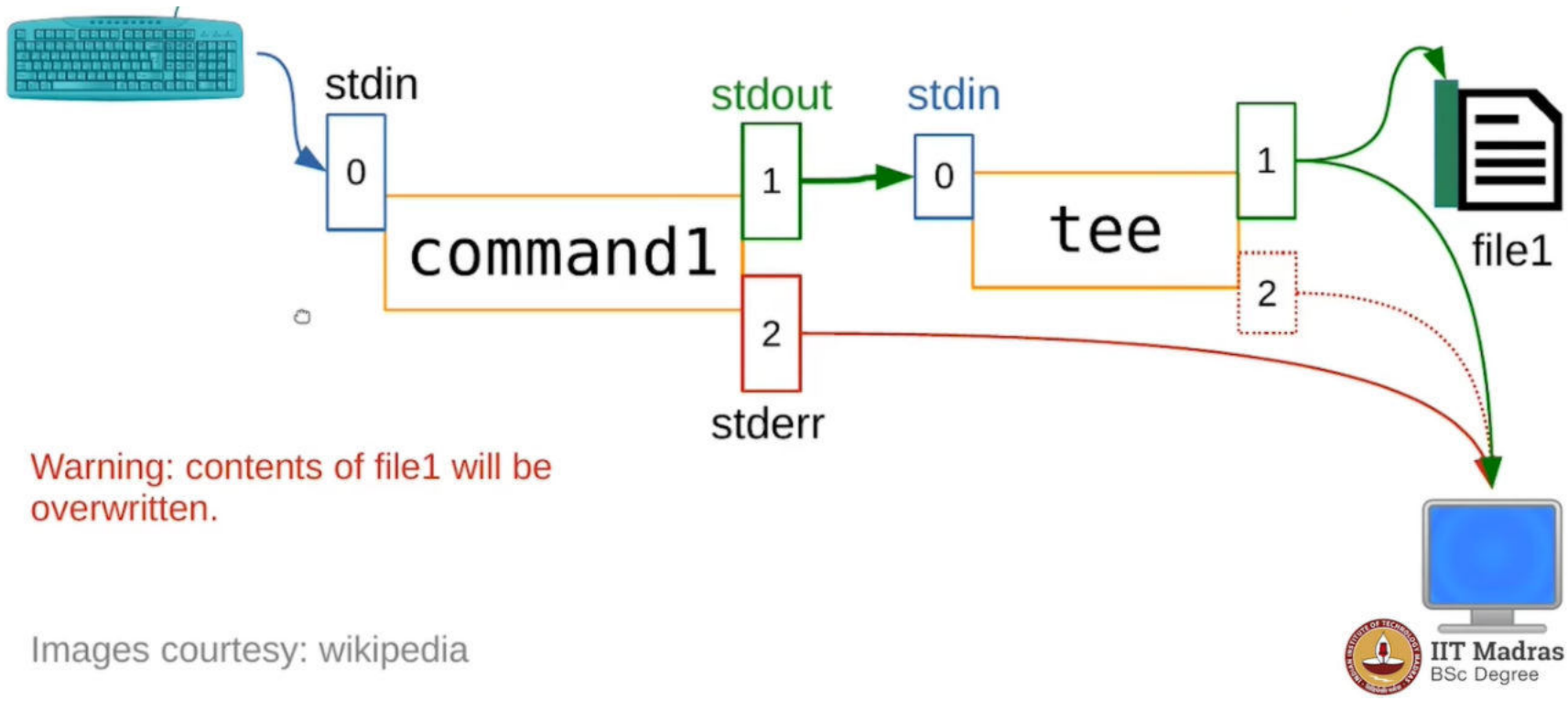
- The output of `command` is written to `file1`

- The `stderr` is written to `/dev/null`, ~~*which gets warped to another dimension*~~



`command1 | tee file1`

- The `tee` command splits the output into 2 streams, one stream is written to the `file1` another, one to the display
  - This command can write to multiple files as well

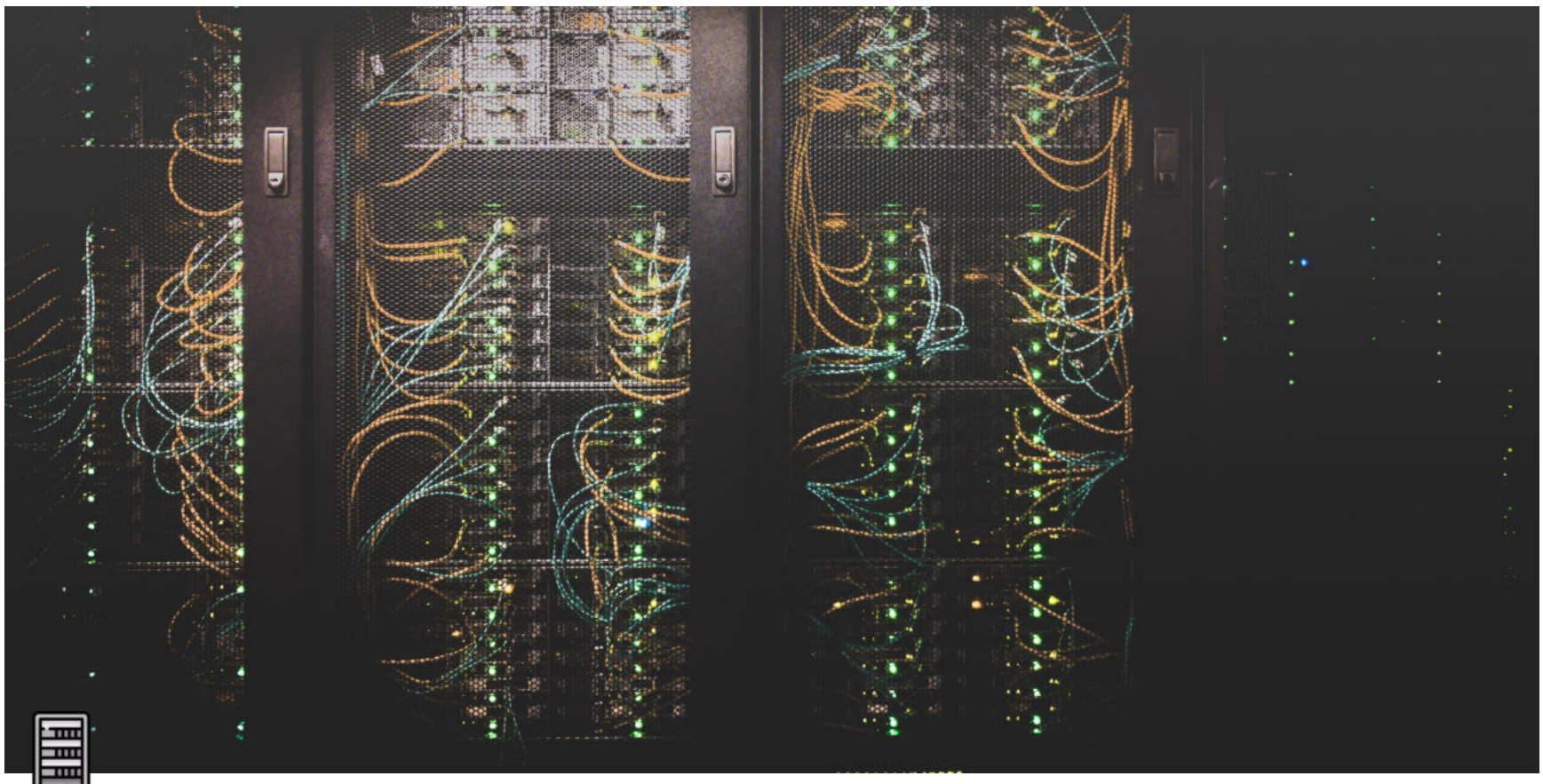


`diff` command

- This command compares files line-by-line

Usage

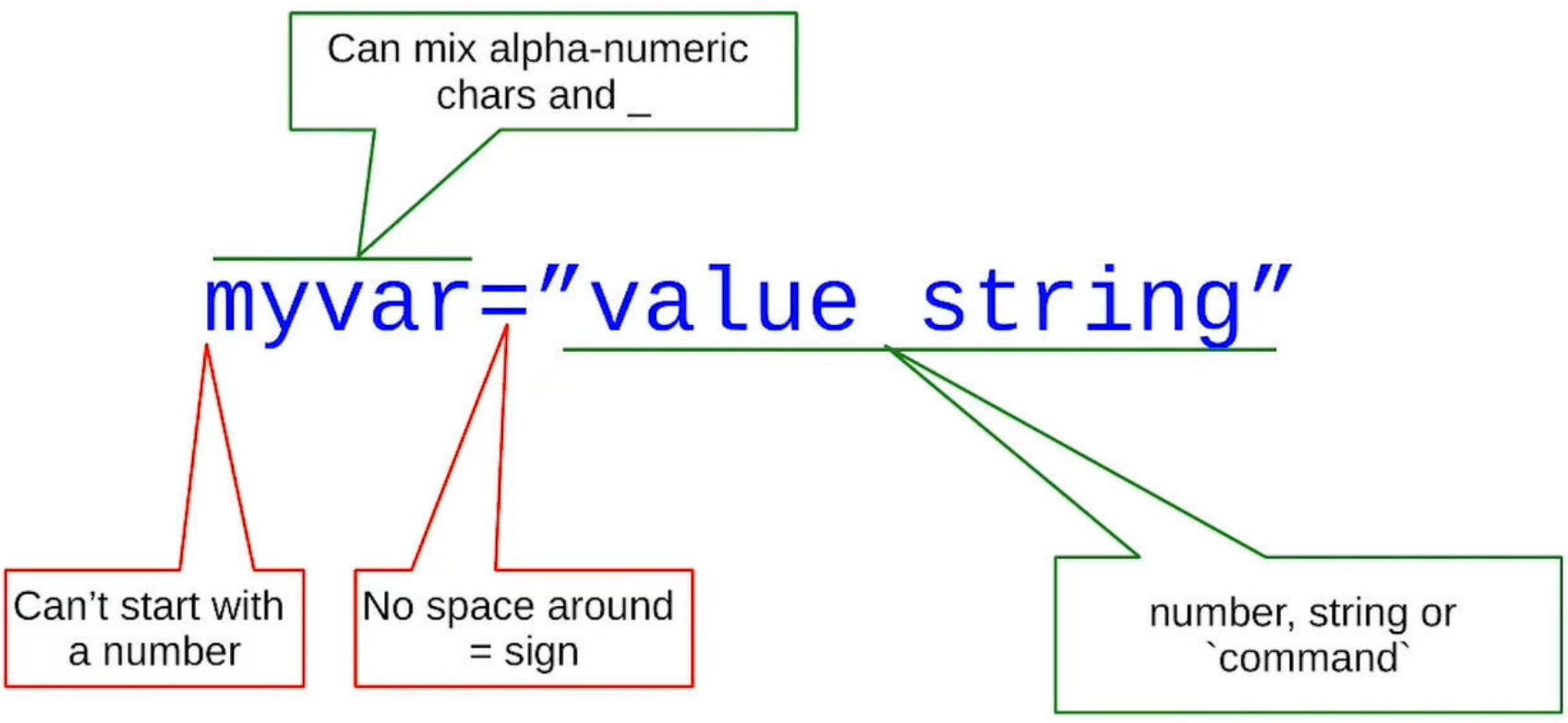
```
diff file1 file2
```



Shell variables

Type	📒 Lecture
Date	@January 9, 2022
Lecture #	3
Lecture URL	https://youtu.be/QX5XElFRpck
Notion URL	https://21f1003586.notion.site/Shell-variables-7fc4b392e99b4d59a13eebefdae88e1e
Week #	3

Creating a variable



Exporting a variable

Exporting means making the value of the variable available to a shell spawned by the current shell (wut)

```
export myvar="value string"
```

OR

```
myvar="value string"
export myvar
```

Using variable values

`echo $myvar`

`echo ${myvar}`

`echo “${myvar}”`

Removing a variable

`unset myvar`

Removing value of a variable

`myvar=`

Test if a variable is set

`[[ -v myvar ]];`

`echo $?`

Return codes:

0 → success (variable `myvar` is set)

1 → failure (variable `myvar` is not set)

Test if a variable is *not* set

`[[ -z ${myvar+x} ]];`

Here, x can be any string

`echo $?`

Return codes:

0 → success (variable myvar is not set)

1 → failure (variable myvar is set)

Substitute default value

If the variable `myvar` is not set, use `“default”` as its default value

`echo ${myvar:-”default”}`

So, if `myvar` *is set, display its value else display “default”*

Reset value if variable is set

If the variable `myvar` is set, then set `“default”` as its value

`echo ${myvar:+”default”}`

So, if `myvar` *is set, change it’s value to “default” and display it else display null*

List of variable names

`echo ${!H*}`

List of names of shell variables that start with `H`

Length of string value

`echo ${#myvar}`

Display length of the string value of the variable `myvar`

If `myvar` *is not set, display 0*

Slice of a string value

`echo ${myvar:5:4}`

Display 4 chars of the string value of the variable `myvar`*, skipping first 5 chars*

Remove matching pattern

`echo ${myvar#pattern}` → *match once*

`echo ${myvar##pattern}` → *match max possible*

Keep matching pattern

`echo ${myvar%pattern}` → *match once*

`echo ${myvar%%pattern}` → *match max possible*

Replace matching pattern

`echo ${myvar/pattern/string}` → *match once and replace with* `string`

`echo ${myvar//pattern/string}` → *match max possible and replace with* `string`

Replace matching pattern by location

`echo ${myvar/#pattern/string}` → *match at beginning and replace with* `string`

`echo ${myvar/%pattern/string}` → *match at the end and replace with* `string`

Changing case

`echo ${myvar,}` → *change the first char to lower case*

`echo ${myvar,,}` → *change all chars to lower case*

`echo ${myvar^}` → *change first char to upper case*

`echo ${myvar^^}` → *change all chars to upper case*

Restricting value types

`declare -i myvar` → *only integers can be assigned*

`declare -l myvar` → *only lower case chars can be assigned*

`declare -u myvar` → *only upper case chars can be assigned*

`declare -r myvar` → *variable is read-only*

You can remove the restrictions by replacing the `-` *sign with a* `+` *sign*

However, `declare +r myvar` will ***NOT*** work

Indexed arrays

`declare -a arr` → *declare* `arr` *as an indexed array*

`$arr[0]="value"` → *set value of element with index 0 in the array*

`echo ${arr[0]}` → *value of the element at index 0 of array*

`echo ${#arr[@]}` → *number of elements in the array*

`echo ${!arr[@]}` → *display all the indices used*

`echo ${arr[@]}` → *display values of all elements of the array*

`unset 'arr[2]'` → *delete element with index 2 in the array*

`arr+=("value")` → *append an element with a value to the end of the array*

Associative arrays

Kind of like Hash maps?

`declare -A hash` → *declare* `hash` *as an associative array*

`$hash["a"]="value"` → *set value of element with index "a" in the array*

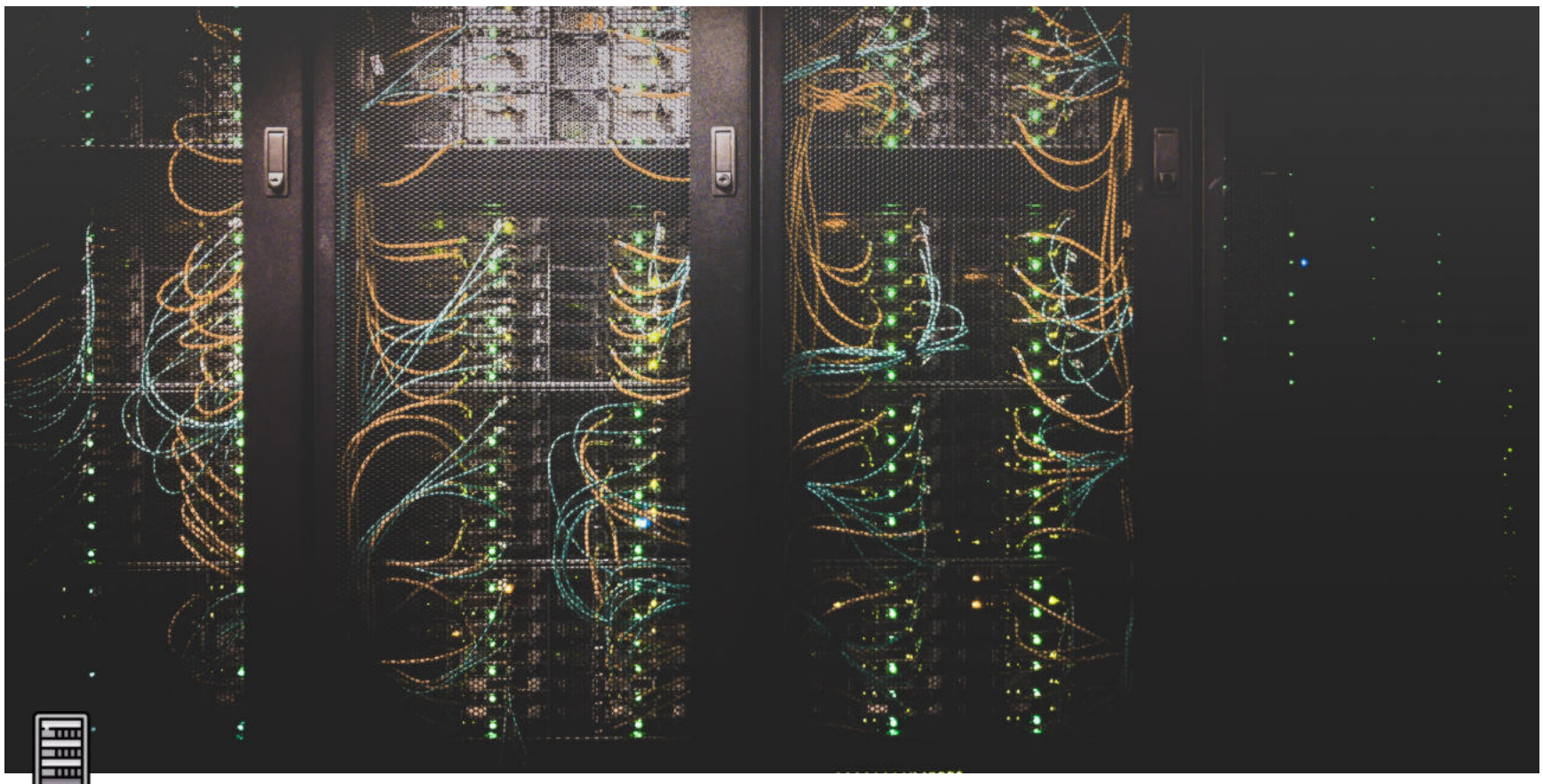
`echo ${hash["a"]}` → *value of element with index (or key?) "a" in the array*

`echo ${#hash[@]}` → *number of elements in the array*

`echo ${!hash[@]}` → *display all indices used*

`echo ${hash[@]}` → *display values of all elements of the array*

`unset 'hash["a"]'` → *delete element with index "a" in the array*



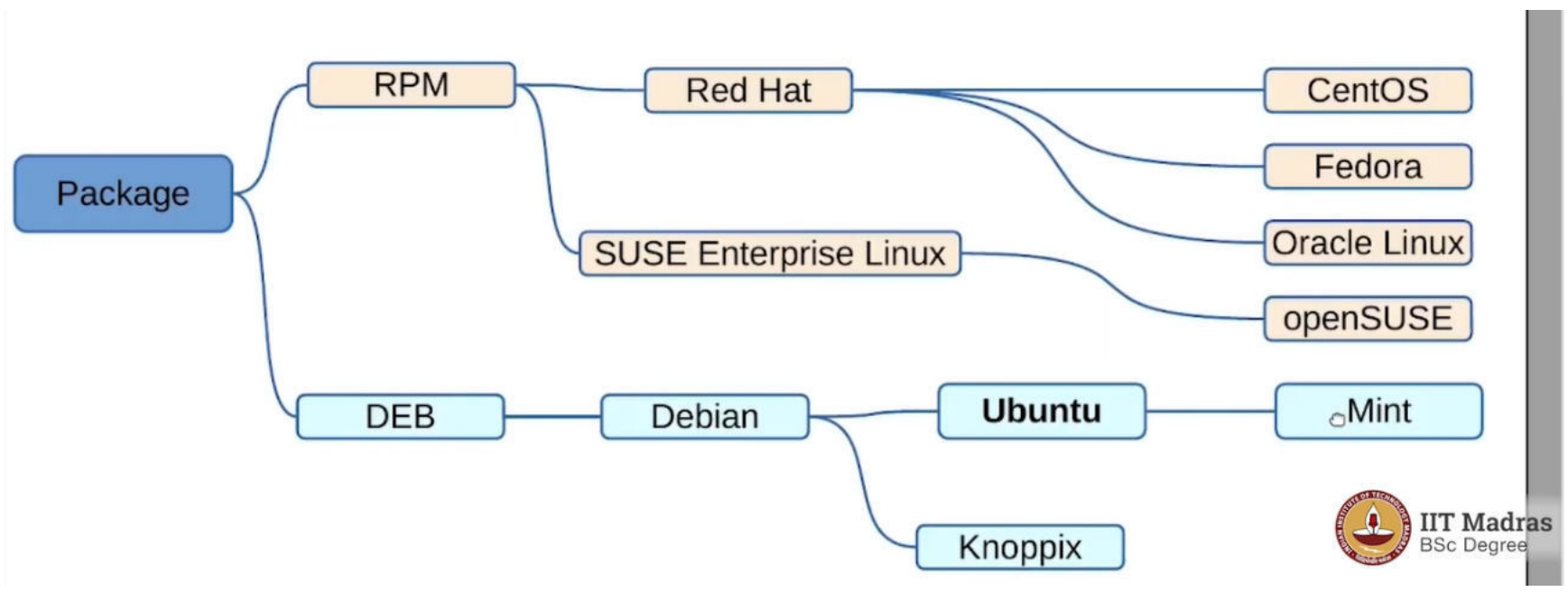
Software Management - pt. 1

Type	📒 Lecture
Date	@January 16, 2022
Lecture #	1
Lecture URL	https://youtu.be/hG-bxCmfArc
Notion URL	https://21f1003586.notion.site/Shell-variables-3c228fce1aef41719f77bc4b8e786ff1
Week #	4

Need for a package manager

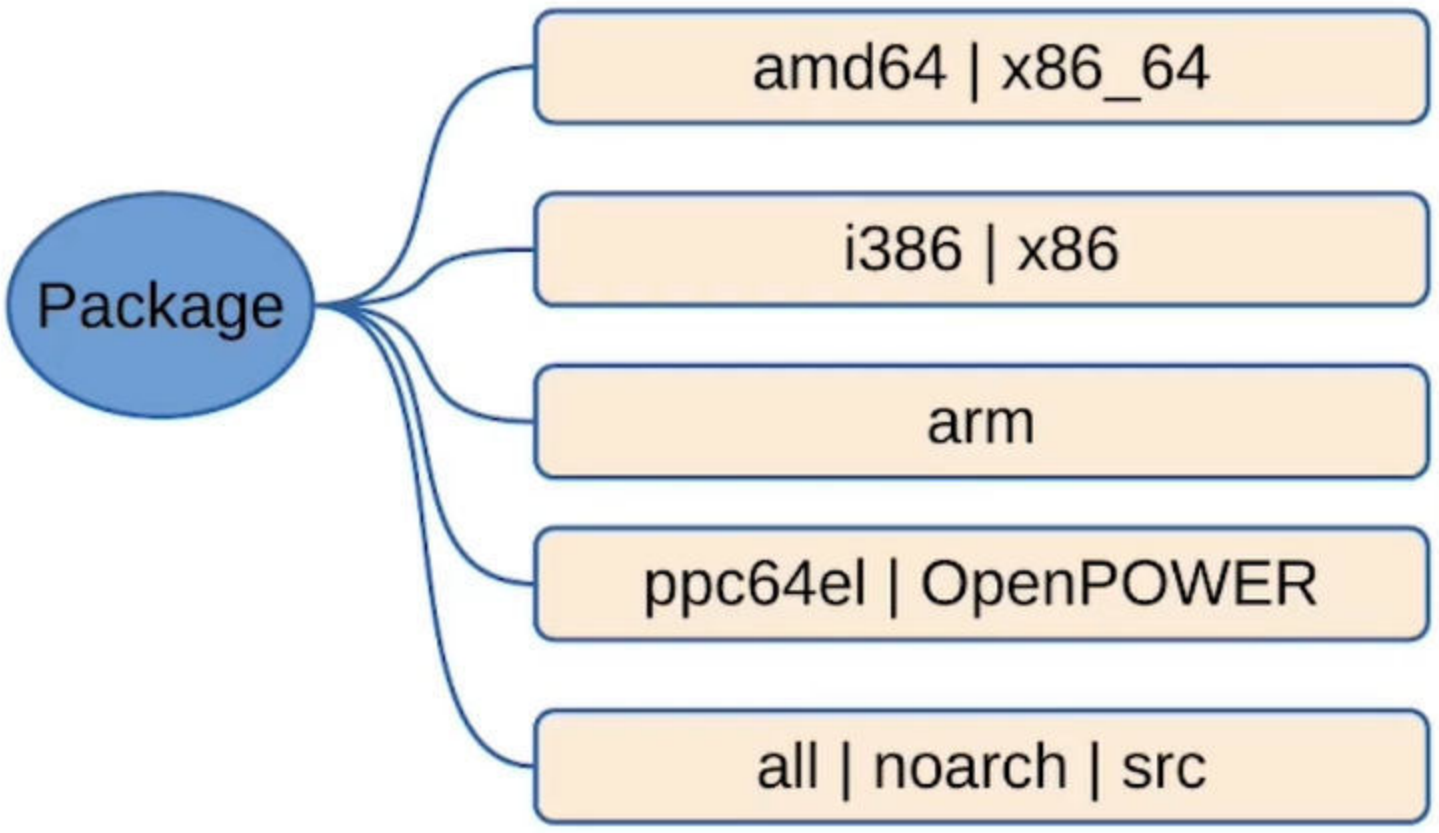
- Tools for installing, updating, removing, managing software
- Install new/updated software across network
- Package — File look up, both ways
- Database of packages on the system including versions
- Dependency checking
- Signature verification tools
- Tools for building packages

Package Types



```
~ lsb_release -a
LSB Version:    n/a
Distributor ID: ManjaroLinux
Description:    Manjaro Linux
Release:        21.2.1
Codename:       Qonos
```

Architecture

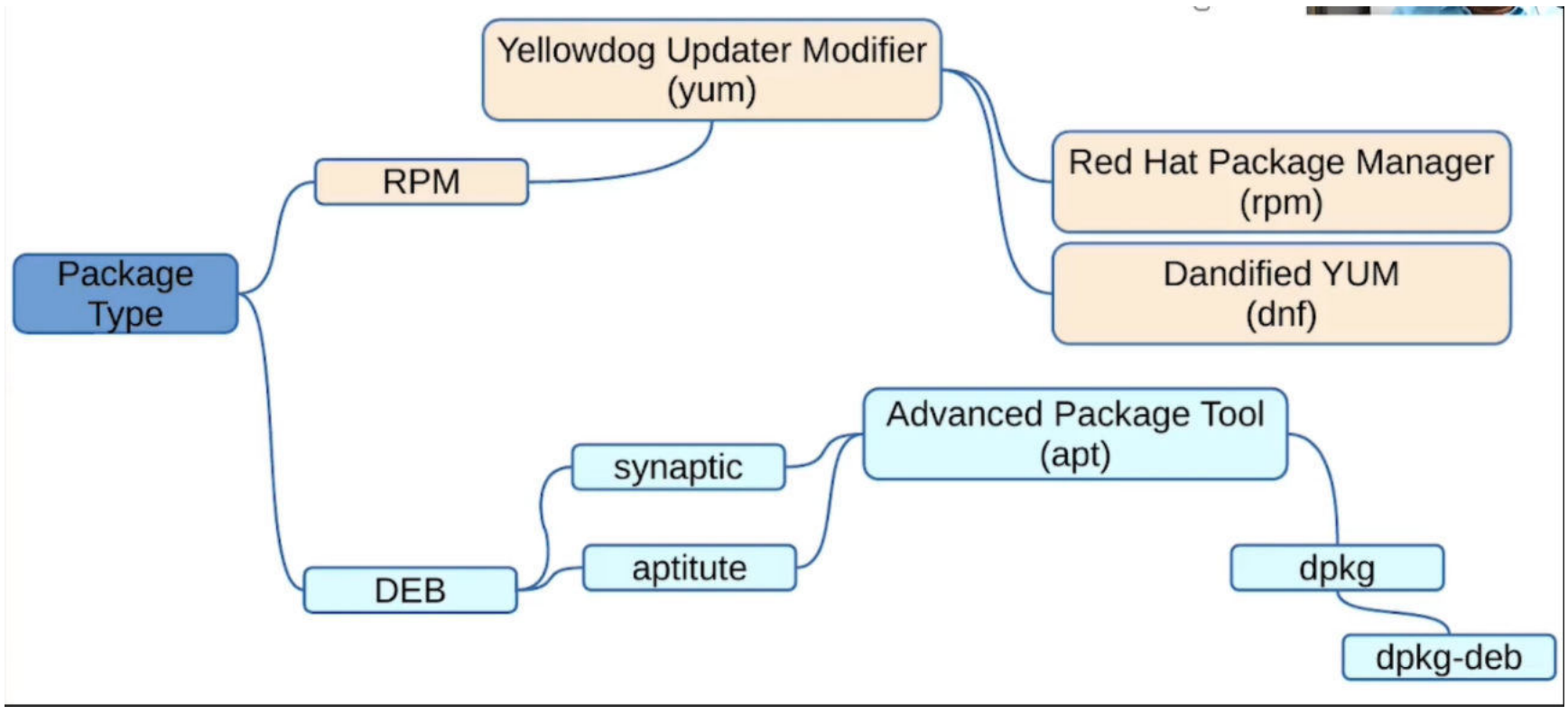


To know your system architecture

~~*Can't spell architecture without arch btw*~~

```
~ uname -a

Linux Zen 5.15.12-1-MANJARO #1 SMP PREEMPT Wed Dec 29 18:08:07 UTC 2021 x86_64 GNU/Linux
```

Tools

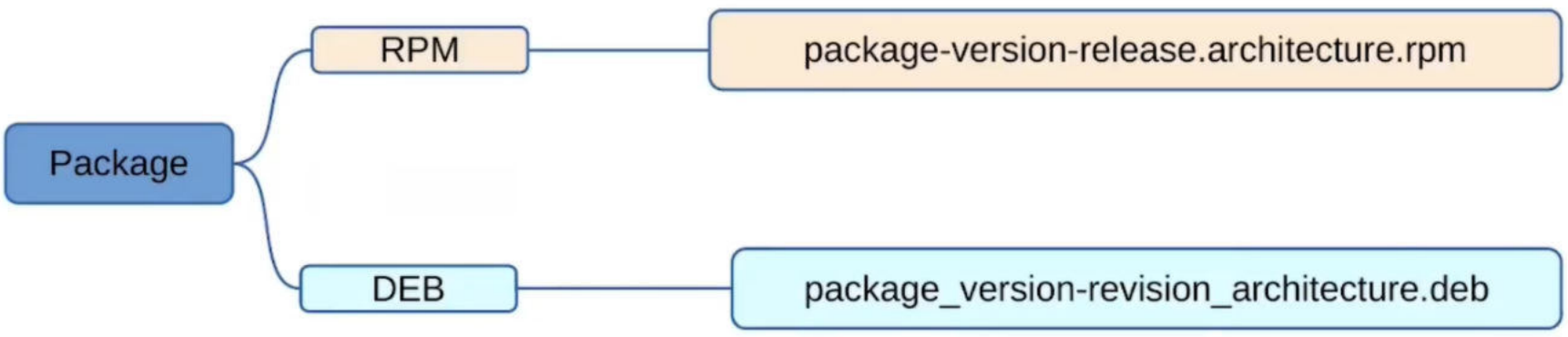


Package Management in Ubuntu using `apt`

Inquiring package db

- Search packages for a keyword
  - `apt-cache search keyword`
- List all packages
  - `apt-cache pkgnames`
  - Try `apt-cache pkgnames <starting-char>` to search to packages with the few starting chars
- Display package records of a package
  - `apt-cache show -a package`

Package names



Package priorities

- **required** → essential to the proper functioning of a system
- **important** → provides functionality that enables the system to run well
- **standard** → included in a standard system installation
- **optional** → can omit if you do not have enough storage
- **extra** → could conflict with packages with higher priority, has specialized requirements, install only if needed

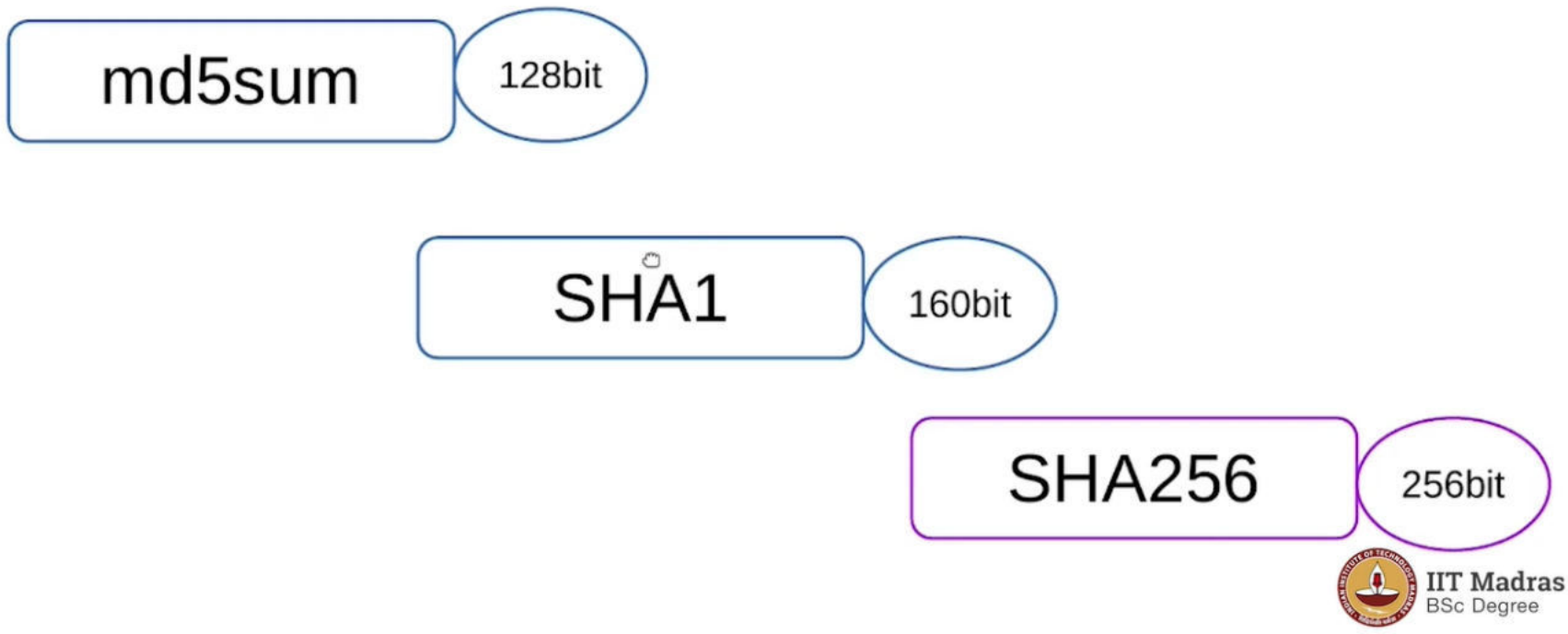
Package sections

https://packages.ubuntu.com/focal

Administration Utilities, Mono/CLI, Communication Programs, Databases, Debug packages, Development, Documentation, Editors, Electronics, Embedded software, Fonts, Games, GNOME, GNU R, GNUstep, Graphics, Haskell, Web Servers, Interpreters, Java, KDE, Kernels, Library development, Libraries, Lisp, Language packs, Mail, Mathematics, Miscellaneous, Network, Newsgroups, OCaml, Perl, PHP, Python, Ruby, Science, Shells, Sound, TeX, Text Processing, Translations, Utilities, Version Control Systems, Video, Web Software, X Window System software, Xfce, Zope/Plone Framework

Checksums

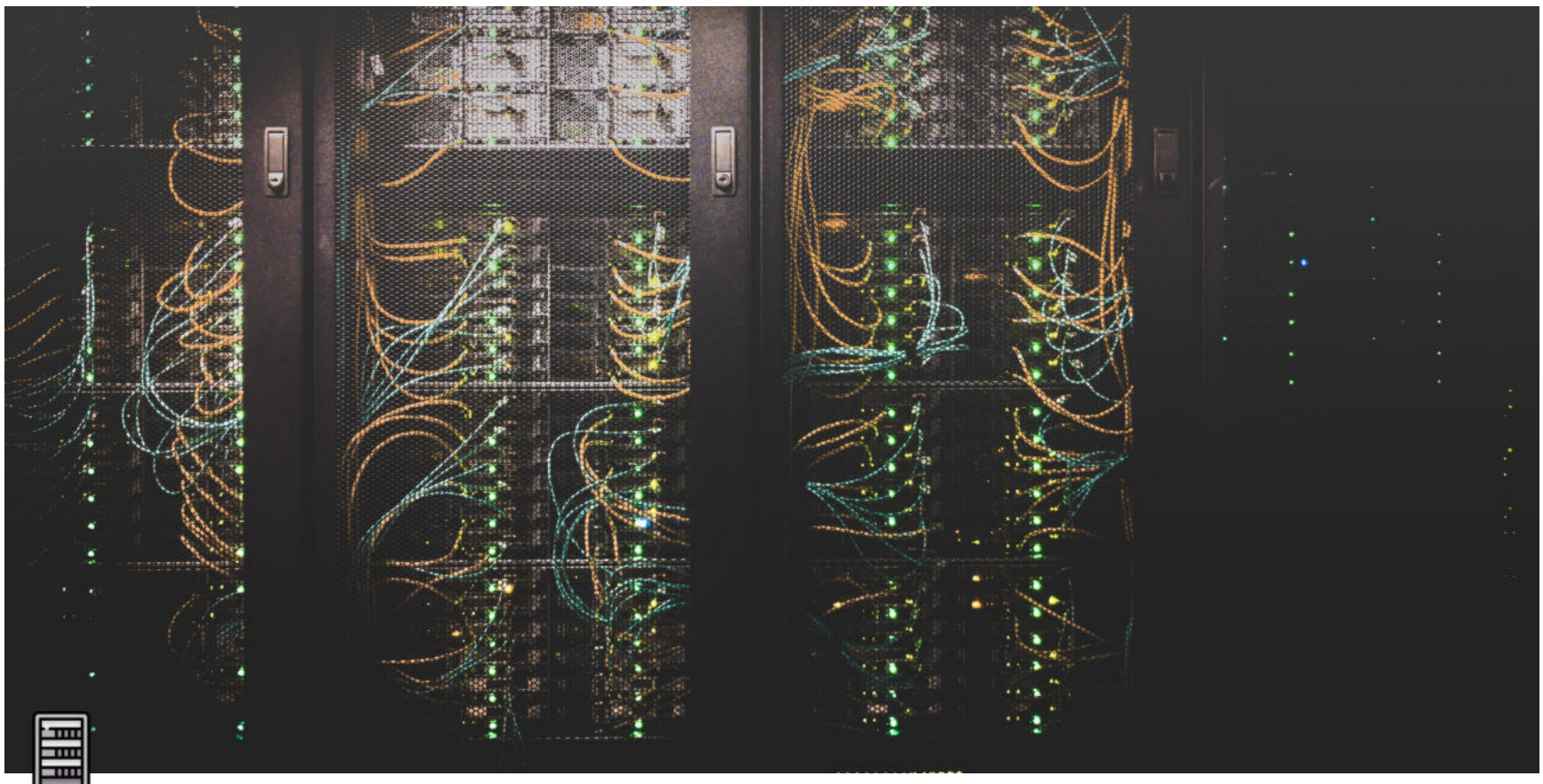
A way to verify that the packages that we are installing are legit



`md5sum <filename>` to get the MD5 checksum of a file, use it to compare it to other files

`sha1sum <filename>` to get the SHA1 checksum of a file, use it to compare it to other files

`sha256sum <filename>` to get the SHA256 checksum of a file, use it to compare it to other files



Software Management - pt. 2

Type	📒 Lecture
Date	@January 16, 2022
Lecture #	2
Lecture URL	https://youtu.be/ctn_s7SDTV0
Notion URL	https://21f1003586.notion.site/Software-Management-b0cd46d7790f479aacf4378fbe9611cf
Week #	4

Only sudoers can install/upgrade/remove packages → `/etc/sudoers`

We can view the `sudoers` file by typing `sudo cat /etc/sudoers`

This will ask for the user password and depending on the `su` status of the user, it will either display the file or show an error like `user is not in the sudoers file. This incident will be reported.`

How can a superuser find out the failed `sudo` attempt by non-su?

Go to `/var/log`

Look for `auth.log` file, these events are reported in this file

Sources for packages

Location: `/etc/apt`

File → `sources.list` → Contains repo URLs for Ubuntu pkgs

Folder → `sources.list.d` → Contains repo URLs for 3rd party pkgs

To get updates

```
sudo apt-get update
```

To upgrade the pkgs (if an upgrade is available)

```
sudo apt-get upgrade
```

You can pass the `-y` flag to not get bugged by the Y/N prompt

To remove older, usually obsolete pkgs

```
sudo apt autoremove
```

To remove a specific pkg

```
sudo apt-get remove <pkg-name>
```

Installing/Updating

- Install a package
  - `apt-get install <pkg-name>`
- Reinstall a package
  - `apt-get reinstall <pkg-name>`

Removing/Cleaning up

- Remove packages that were automatically installed to satisfy a dependency and not needed
  - `apt-get autoremove`
- Clean local repository of retrieved package files
  - `apt-get clean`
- Purge package files from the system
  - `apt-get purge package`

Package management in Ubuntu using `dpkg`

Text info regarding packages found at `/var/lib/dpkg`

Files: `arch, available, status`

Folder: `info`

Using `dpkg`

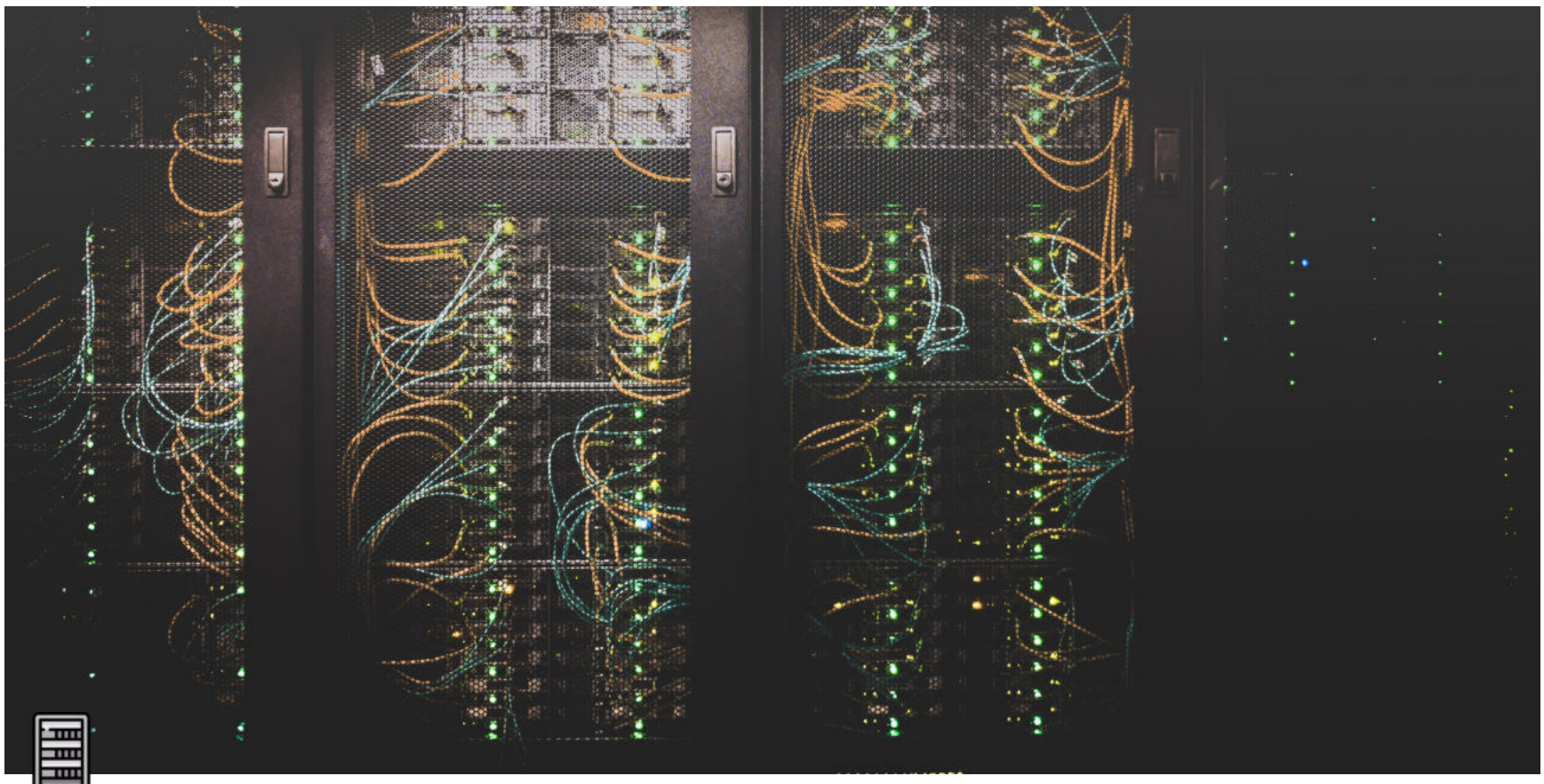
- List all the packages whose names match the pattern
  - `dpkg -l pattern`
- List installed files that came from packages
  - `dpkg -L package`
- Report the status of packages
  - `dpkg -s package`
- Search installed packages for a file
  - `dpkg -S pattern`
- A tool to query the `dpkg` database
  - `dpkg-query`
  - Example usage
    - `dpkg-query -W -f='${Section} ${binary:Package}\n'`

Installing a `deb` package

`dpkg -i filename.deb`

By default, use package management pointing to a reliable repository

Uninstalling packages using `dpkg` is ***NOT*** recommended



Pattern Matching - pt. 1

Type	📒 Lecture
Date	@January 17, 2022
Lecture #	3
Lecture URL	https://youtu.be/1y85iTaqq8Y
Notion URL	https://21f1003586.notion.site/Pattern-Matching-pt-1-8cb1aa5c0e6c42b2a9b517b040006284
Week #	4

Pattern matching

`regex` & `grep`

Regex

- regex is a pattern template to filter text
- BRE: POSIX Basic Regular Expression engine
- ERE: POSIX Extended Regular Expression engine

Why learn regex?

- Languages: Java, Perl, Python, Ruby, ...
  - To process input from the user
  - To perform string operations
- Tools: `grep`, `sed`, `awk`, ...
- Applications: MySQL, PostgreSQL, ...

Usage

- `grep ‘pattern’ filename`
- `command | grep ‘pattern’`
- Default engine: BRE
- Switch to use ERE:
  - `egrep ‘pattern’ filename`
  - `grep -E ‘pattern’ filename`

Special characters (BRE & ERE)

.	Any single character except null or newline
*	Zero or more of the preceding character / expression
[ ]	Any of the enclosed characters; hyphen (-) indicates character range
^	Anchor for beginning of line or negation of enclosed characters
$	Anchor for end of line
\	Escape special characters

Special characters (BRE)

\{n,m\}	Range of occurances of preceding pattern at least n and utmost m times
\(\)	Grouping of regular expressions

Special characters (ERE)

{n,m}	Range of occurances of preceding pattern at least n and utmost m times
( )	Grouping of regular expressions
+	One or more of preceding character / expression
?	Zero or one of preceding character / expression
|	Logical OR over the patterns

Character classes

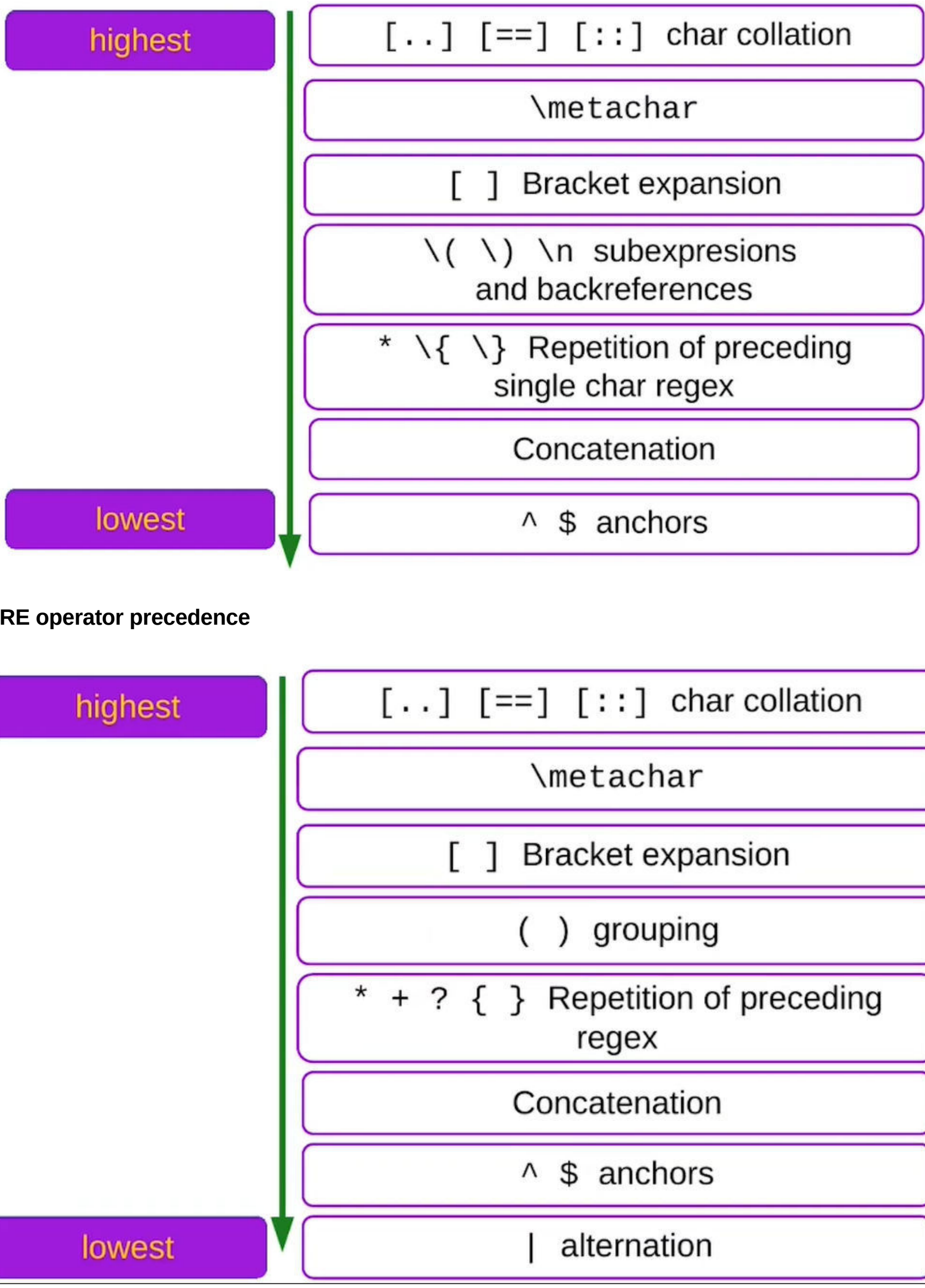
`[[:print:]]`	Printable	`[[:blank:]]`	Space / Tab
`[[:alnum:]]`	Alphanumeric	`[[:space:]]`	Whitespace
`[[:alpha:]]`	Alphabetic	`[[:punct:]]`	Punctuation
`[[:lower:]]`	Lower case	`[[:xdigit:]]`	Hexadecimal
`[[:upper:]]`	Upper case	`[[:graph:]]`	Non-space
`[[:digit:]]`	Decimal digits	`[[:cntrl:]]`	Control characters

To make it slightly easy to match pattern

Backreferences

- `\1` through `\9`
- `\n` matches whatever was matched by the `n`th earlier paranthesized sub-expression
- A line with two occurences of *“hello”* will be matched using: `\(hello\).*\1`

BRE operator precedence



ERE operator precedence

Uses of `grep`

```
~/Documents/week4 cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4 grep Raman names.txt
ED22B902 Raman Singh
~/Documents/week4 grep 'Raman' names.txt
ED22B902 Raman Singh
~/Documents/week4 grep 'Anu' names.txt
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
~/Documents/week4 grep 'Sa' names.txt
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
~/Documents/week4 grep 'ai' names.txt
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
~/Documents/week4 cat names.txt | grep 'ai'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
~/Documents/week4
```

Usage of `.` in the `grep` command

`.` is like a wildcard for a single character

```
~/Documents/week4 cat names.txt | grep 'S.n'
ED22B902 Raman Singh
PH22B907 Vel Sankaran
~/Documents/week4 cat names.txt | grep '.am'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
```

`$` is used to denote an anchor

Like a pattern at the end of the line

```
~/Documents/week4 cat names.txt | grep '.am$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
```

Well what if your name contains a `.`

Then escape it using the `\` character

```
~/Documents/week4 cat names.txt | grep '\.'
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
~/Documents/week4
```

If we want the `.` to be necessarily after a character

```
~/Documents/week4 cat names.txt | grep '.\.'
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
~/Documents/week4
```

Match string using anchors at the beginning

ask `grep` to ignore the case by passing in the `-i` flag

```
~/Documents/week4 cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4 cat names.txt | grep '^M'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
~/Documents/week4 cat names.txt | grep '^E'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
~/Documents/week4 cat names.txt | grep '^e'
~/Documents/week4 cat names.txt | grep -i '^e'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
~/Documents/week4
```

Match a pattern at the end of the line, a word boundary one might say

`\b` looks for a word boundary, so that pattern could also occur at the end of a word in the middle of a line

`$` looks for line boundary only, so the pattern occurs at the end of the line

```
~/Documents/week4 cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
~/Documents/week4 cat names.txt | grep 'am\b'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
~/Documents/week4 cat names.txt | grep 'am$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
~/Documents/week4
```

Usage of square brackets `[]`

Here, the first character in the pattern is followed by either of the 2 characters given in `[]`

In the `grep 'S.*[mn]'` command, it matches from the start of the line

We add `\b` to mark a word boundary just to match it within a word

Had to switch to `bash` to show the formatting

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[ME]E'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'E[ED]'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'M[EM]'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep 'S.*[mn]'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\bS.*[mn]'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
```

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[aeiou]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[aeiou][aeiou]'
ME22B903 Umair Ahmad
EE22B905 Anu K. Jain
[kashif@Zen week4]$ cat names.txt | grep 'B90[1-4]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | grep 'B90[5-7]'
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'B90[1-9]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```

The last command in the following screenshot does negation

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '[M-Z][aeiou]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'B90[^5-7]'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```

Number of times a character should occur

In the curly braces, we provide the # of times the preceding character should be matched

We can pass one number or a multiple numbers separated by comma for their matching

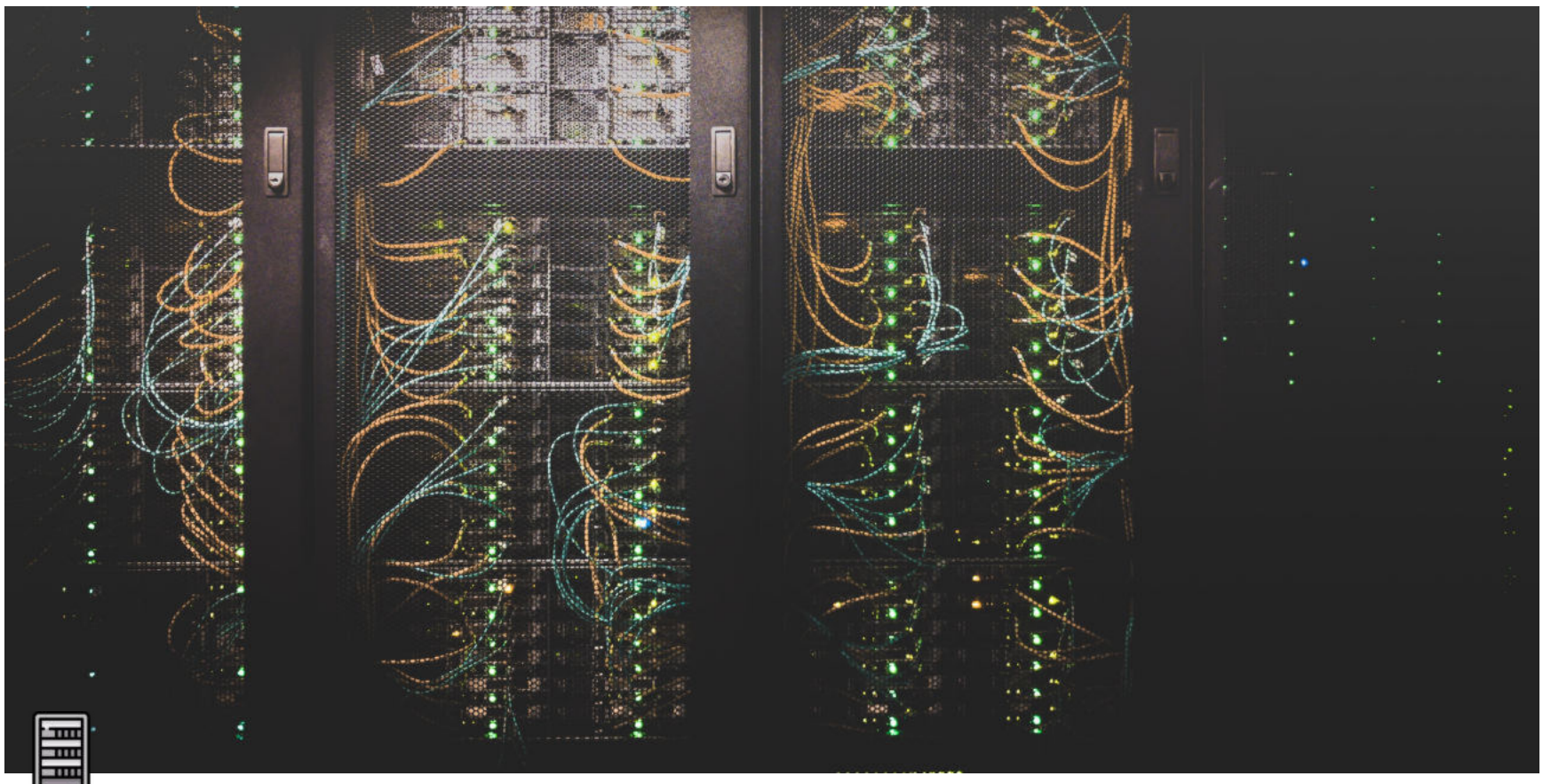
```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep 'M\{2\}'
MM22B901 Mary Manickam
[kashif@Zen week4]$ cat names.txt | grep 'M\{1,2\}'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(ma\)'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | grep '\(ma\).*\1'
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep '\(.a\).*\1'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | grep '\(a.\).*\1'
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{3\}'
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{2\}'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | grep '\(a.\)\{2,3\}'
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M+'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$ cat names.txt | egrep '^M+'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | egrep '^M*'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M*a'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep 'M.*a'
MM22B901 Mary Manickam
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
[kashif@Zen week4]$
```

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(ma)+'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | egrep '(ma)*'
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```

```
[kashif@Zen week4]$ cat names.txt
MM22B901 Mary Manickam
ED22B902 Raman Singh
ME22B903 Umair Ahmad
CS22B904 Charles M. Sagayam
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(ED|ME)'
ED22B902 Raman Singh
ME22B903 Umair Ahmad
[kashif@Zen week4]$ cat names.txt | egrep '(Anu|Raman)'
ED22B902 Raman Singh
EE22B905 Anu K. Jain
NA22B906 Anupama Sridhar
[kashif@Zen week4]$ cat names.txt | egrep '(am|an)'
MM22B901 Mary Manickam
ED22B902 Raman Singh
CS22B904 Charles M. Sagayam
NA22B906 Anupama Sridhar
PH22B907 Vel Sankaran
[kashif@Zen week4]$ cat names.txt | egrep '(am|an)$'
MM22B901 Mary Manickam
CS22B904 Charles M. Sagayam
PH22B907 Vel Sankaran
[kashif@Zen week4]$
```



Pattern Matching - pt. 2

Type	📒 Lecture
Date	@January 17, 2022
Lecture #	4
Lecture URL	https://youtu.be/XQUJPRc-7zA
Notion URL	https://21f1003586.notion.site/Pattern-Matching-pt-2-18da9c08be534558ad86e28d07cba0bd
Week #	4

Match package names that are 4 characters long

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' .{4}$'
```

Match package names that are 3 characters long and start with the letter `g`

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' g.{3}$'
```

Match package names that are between 1 to 5 characters long and start with the letter `g`

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' g.{1,5}$'
```

Match package names that are from the `math` category

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep '^math'
```

make sure to use the `^` (hat) character in the front of the regex pattern to match the `math` category, otherwise it will match package category and the names

Match package names that from KDE

```
dpkg-query -W -f='${Section} ${binary:Package}\n' | egrep ' kd.*$'
```

To skip empty lines from a file

```
cat filename.txt | egrep -v '^$'
```

- Pick any 12 digit or more number from a text file
  - `egrep '[[:digit:]]{12}' filename.txt`
- Pick any 6 digit or more number from a text file
  - `egrep '[[:digit:]]{6}' filename.txt`

But, there is one problem, if there is any number that is more than 12 digits or more than 6 digits respectively, it will pick that up too

- Pick an exactly 6 digit number from a text file
  - Add a word boundary `\b`

    - `egrep '\b[[:digit:]]{6}\b' filename.txt`
- Pick a roll number (of the type MM22B001) from a text file
    - `egrep '\b[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]][[:digit:]]{3}\b' filename.txt`
- Pick a URL from a text file (like github.com or https://www.iitm.ac.in)
    - `egrep '\b[[:alnum:]]+\.[[:alnum:]]+\b' filename.txt`

`cut`

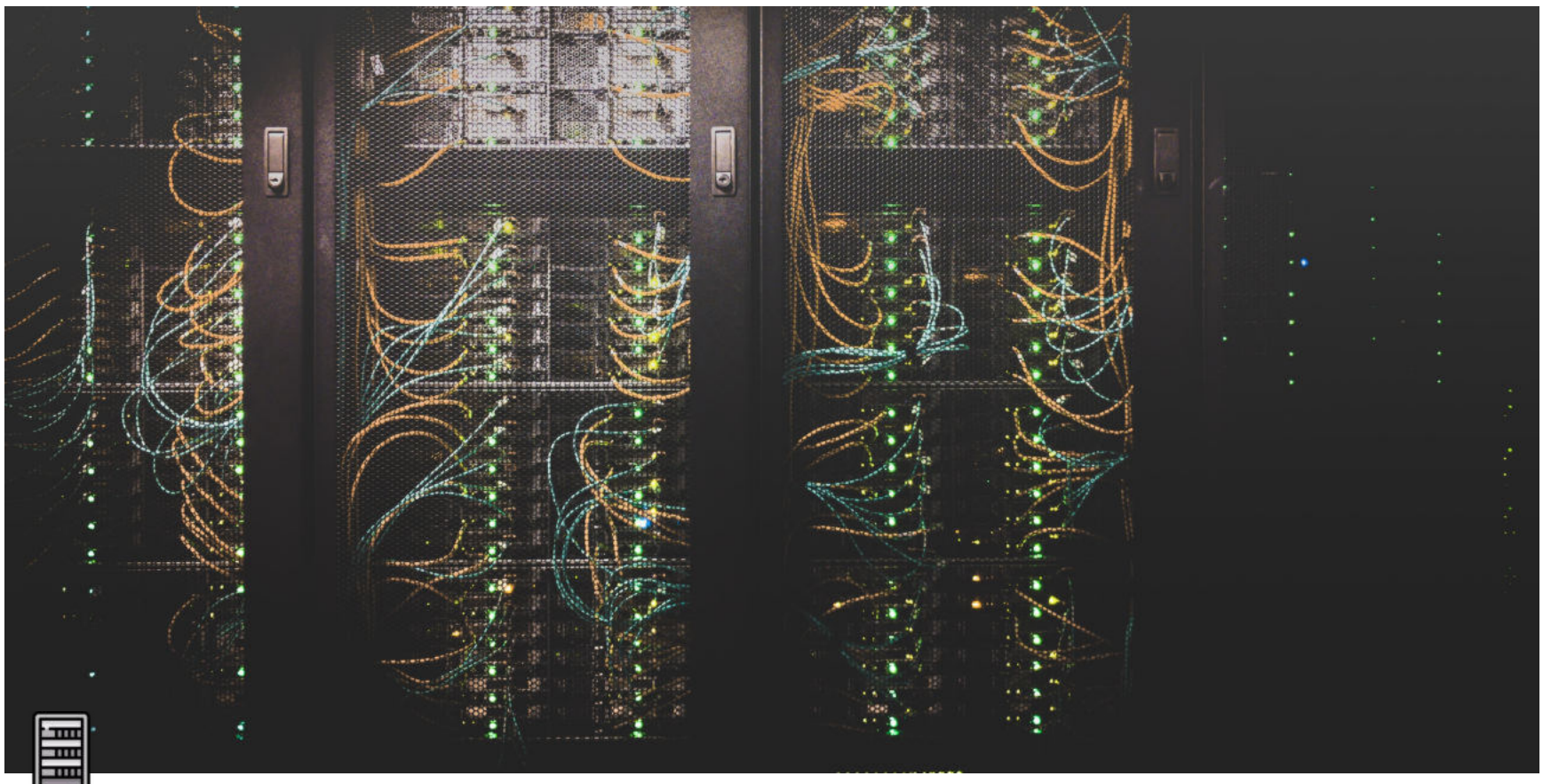
A command used to cut lines from files

does horizontal trimming

A sample file `fields.txt`

```
1234;hello world,line-1
234567;welcome cmdline,line-2
3456;parse text,line-3
```

- Cut first 4 characters from the beginning of the lines
    - `cut -c 1-4 fields.txt`
- Cut the next 4 characters from the previous
    - `cut -c 5-8 fields.txt`
- We can skip the beginning or the ending of the substring parameter, *it works like python*
    - Cut 4 chars from the beginning
        - `cut -c -4 fields.txt`
    - Cut from 8th char to the end
        - `cut -c 8- fields.txt`
- Use space as the delimiter and print the first field
    - `cat fields.txt | cut -d " " -f 1`
- Similarly, print the second field
    - `cat fields.txt | cut -d " " -f 2`
- If we want both fields
    - `cat fields.txt | cut -d " " -f 1-2`
- Delimit at a semi-colon `;` and get the first field
    - `cat fields.txt | cut -d ";" -f 1`
- Similarly, get the 2nd field
    - `cat fields.txt | cut -d ";" -f 2`
- We can pipe multiple commands
    - To get the part of the line between `;` and `,`
        - `cat fields.txt | cut -d ";" -f 2 | cut -d "," -f 1`
    - To do the same thing using `grep` (*similar thing, not exactly the same*)
        - `cat fields.txt | egrep ';.*,'`
- To get the part `welcome cmdline` from the file `fields.txt`
    - `cat fields.txt | cut -d ";" -f 2 | cut -d "," -f 1 | head -n 2 | tail -n 1`

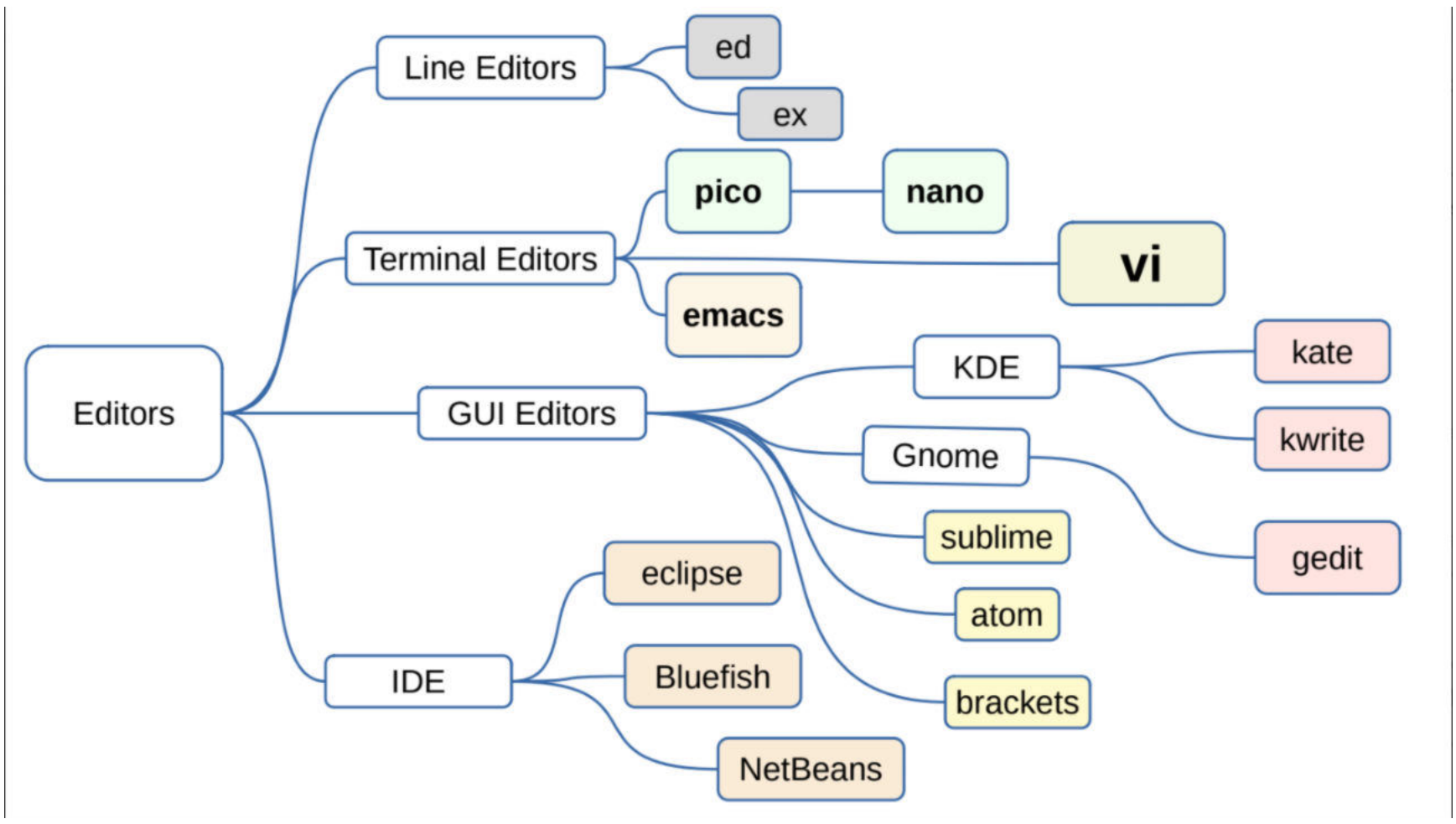


Command Line Editors - pt. 1

Type	Lecture
Date	@February 2, 2022
Lecture #	1
Lecture URL	https://youtu.be/NIIZ1cgrO7g
Notion URL	https://21f1003586.notion.site/Command-Line-Editors-pt-1-d1e3cf98aed64ed78a8e621b73e43636
Week #	5

Command line editors

Working with text files in the terminal



Features

- Scrolling, view modes, current position in the file

- Navigation (char, word, line, pattern)
- Insert, Replace, Delete
- Cut-Copy-Paste
- Search-Replace
- Language-aware syntax highlighting
- Key-maps, init scripts, macros
- Plugins

`ed`

Show the Prompt	P
Command Format	`[addr[,addr]]cmd[params]`
commands for location	*2* . $ % + - , ; /RE/
commands for editing	f p a c d i j s m u
execute a shell *command*	*!command*
edit a file	e *filename*
read file contents into buffer	r *filename*
read *command* output into buffer	r *!command*
write buffer to filename	w *filename*
quit	q

Command Format → `[starting-address[,ending-address]command[command-parameters]]`

To locate the cursor on any particular line of the file ...

We can use the line number itself

- press **2** → in the 2nd line of the text file
- press **. (dot)** → referring to the current line the cursor is
- press **$** → refers to the last line
- press **%** → refers to all the lines, i.e. any action we are doing applies to all the lines
- press **+** → line after the cursor
- press **- (minus sign)** → line before the cursor
- press **, (comma)** → represent the entire buffer, i.e. the whole file
- press **; (semicolon)** → refers the end of the text file from the current position
- `/RE/` → To match a specific Regular Expression in the file

A way to invoke the `ed` editor

```
ed file.txt
```

and it shows the number of bytes in the file, instead of the contents of the file ... *ayo wut*

```
~/Documents/week5 ed test.txt
117
P  This gives a prompt
*1  Move the cursor to line #1
line-1 hello world
*$  Move the cursor to end of the buffer
line-4 end of file
*,p Show all the lines in the file
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*2,3p  Show lines from 2 to 3
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
*/hello/  Find the line with the pattern “hello”
line-1 hello world
*/oldest/ Find the line with the pattern “oldest”
line-3 ed is perhaps the oldest editor out there
*1 Move to line #1
line-1 hello world
*+ Move to the next line
line-2 welcome to line editor
*- Move to the previous line
line-1 hello world
*3 Move to line #3
line-3 ed is perhaps the oldest editor out there
*;p Get all the lines from current line to the end of file
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*%p Display all the lines of the file
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*. Get the current line
line-4 end of file
```

Run a bash command, read the bash command’s output and write it to the file

```
~/Documents/week5 ed test.txt
117
P
*!date
Tuesday 01 February 2022 09:03:06 PM IST
!
*r !date
41
*w
158
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
Tuesday 01 February 2022 09:03:11 PM IST
*
```

Delete the last line of the file and write the changes to the file

```
~/Documents/week5 ed test.txt
158
,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
Tuesday 01 February 2022 09:03:11 PM IST
$
Tuesday 01 February 2022 09:03:11 PM IST
.d
P
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*w
117
*q
```

Append a line to the file (press . then enter to exit)

```
~/Documents/week5 ed test.txt
117
P
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*1
line-1 hello world
*a
appended this line after line-1
.
*,p
line-1 hello world
appended this line after line-1
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*
```

Search and Replace

```
*2
appended this line after the first line
*s/appended/Appended
Appended this line after the first line
*,p
line-1 hello world
Appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
```

File name

```
*f
test.txt
```

Print the current line

```
*p
line-4 end of file
```

Append to the current line

```
*a
This line is appended at the end of the file
.
```

Join line 5 and 6 (The command is `5,6j`)

```
*5
line-4 end of file
*5,6j
*p
line-4 end of fileThis line is appended at the end of the file
*.
line-4 end of fileThis line is appended at the end of the file
```

Move the current line after the given line # (*We are moving the current line below line 1 here*)

```
*m1
*,p
line-1 hello world
line-4 end of fileThis line is appended at the end of the file
Appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
```

Undo the previous operation

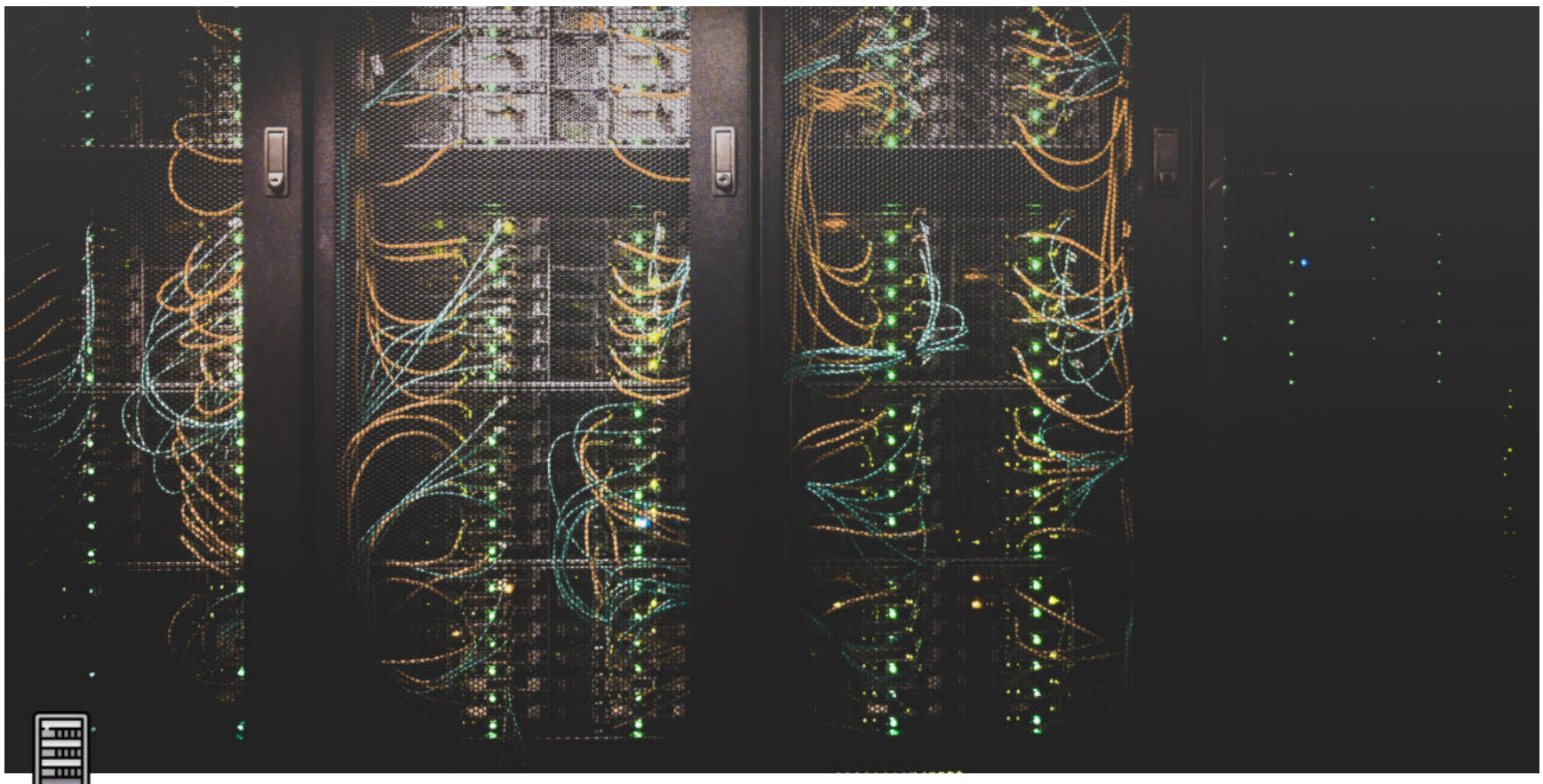
```
*u
*,p
line-1 hello world
Appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of fileThis line is appended at the end of the file
```

Add a prefix to all the lines and then modify the prefix on some lines

```
~/Documents/week5 ed test.txt
201
P
*,p
line-1 hello world
Appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of fileThis line is appended at the end of the file
*%s/\(.*\)/PREFIX \1/
*,p
PREFIX line-1 hello world
PREFIX Appended this line after the first line
PREFIX line-2 welcome to line editor
PREFIX line-3 ed is perhaps the oldest editor out there
PREFIX line-4 end of fileThis line is appended at the end of the file
*3,5s/PREFIX/prefix/
*,p
PREFIX line-1 hello world
PREFIX Appended this line after the first line
prefix line-2 welcome to line editor
prefix line-3 ed is perhaps the oldest editor out there
prefix line-4 end of fileThis line is appended at the end of the file
```

`ed/ex` commands

f	show name of file being edited
p	print the current line
a	append at the current line
c	change the line
d	delete the current line
i	insert line at the current position
j	join lines
s	search for regex pattern
m	move current line to position
u	undo latest change



Command Line Editors - pt. 2

Type	📒 Lecture
Date	@February 2, 2022
Lecture #	2
Lecture URL	https://youtu.be/HLhza4vZTsI
Notion URL	https://21f1003586.notion.site/Command-Line-Editors-pt-2-258a6a2baeaa4ec59dee72011a8382b6
Week #	5

`readlink`

Prints the resolved symbolic links or canonical file names, *but what does that mean?*

If we have file which is a symbolic link to file which in turn is another symbolic link to a file and so on ...

The `readlink` command will display the actual file the initial symbolic link is referring to

Example usage

```
readlink -f /usr/bin/pico
```

Output

```
/usr/bin/nano
```

`.bashrc`

It is a config file that is read by the bash shell everytime it opens

`nano`

`nano` is a text editor, example syntax is `nano filename`

It also does syntax highlighting

File handling

Ctrl+S	Save current file
Ctrl+O	Offer to write file ("Save as")
Ctrl+R	Insert a file into current one
Ctrl+X	Close buffer, exit from nano

Editing

Ctrl+K	Cut current line into cutbuffer
Alt+6	Copy current line into cutbuffer
Ctrl+U	Paste contents of cutbuffer
Alt+T	Cut until end of buffer
Ctrl+]	Complete current word
Alt+3	Comment/uncomment line/region
Alt+U	Undo last action
Alt+E	Redo last undone action

Search and replace

Ctrl+Q	Start backward search
Ctrl+W	Start forward search
Alt+Q	Find next occurrence backward
Alt+W	Find next occurrence forward
Alt+R	Start a replacing session

Deletion

Ctrl+H	Delete character before cursor
Ctrl+D	Delete character under cursor
Alt+Bsp	Delete word to the left
Ctrl+Del	Delete word to the right
Alt+Del	Delete current line

Operations

Ctrl+T	Execute some command
Ctrl+J	Justify paragraph or region
Alt+J	Justify entire buffer
Alt+B	Run a syntax check
Alt+F	Run a formatter/fixer/arranger
Alt+:	Start/stop recording of macro
Alt+;	Replay macro

Moving around

Ctrl+B	One character backward
Ctrl+F	One character forward
Ctrl+←	One word backward
Ctrl+→	One word forward
Ctrl+A	To start of line
Ctrl+E	To end of line
Ctrl+P	One line up
Ctrl+N	One line down
Ctrl+↑	To previous block
Ctrl+↓	To next block
Ctrl+Y	One page up
Ctrl+V	One page down
Alt+\	To top of buffer
Alt+/	To end of buffer

Special movement

Alt+G	Go to specified line
Alt+]	Go to complementary bracket
Alt+↑	Scroll viewport up
Alt+↓	Scroll viewport down
Alt+<	Switch to preceding buffer
Alt+>	Switch to succeeding buffer

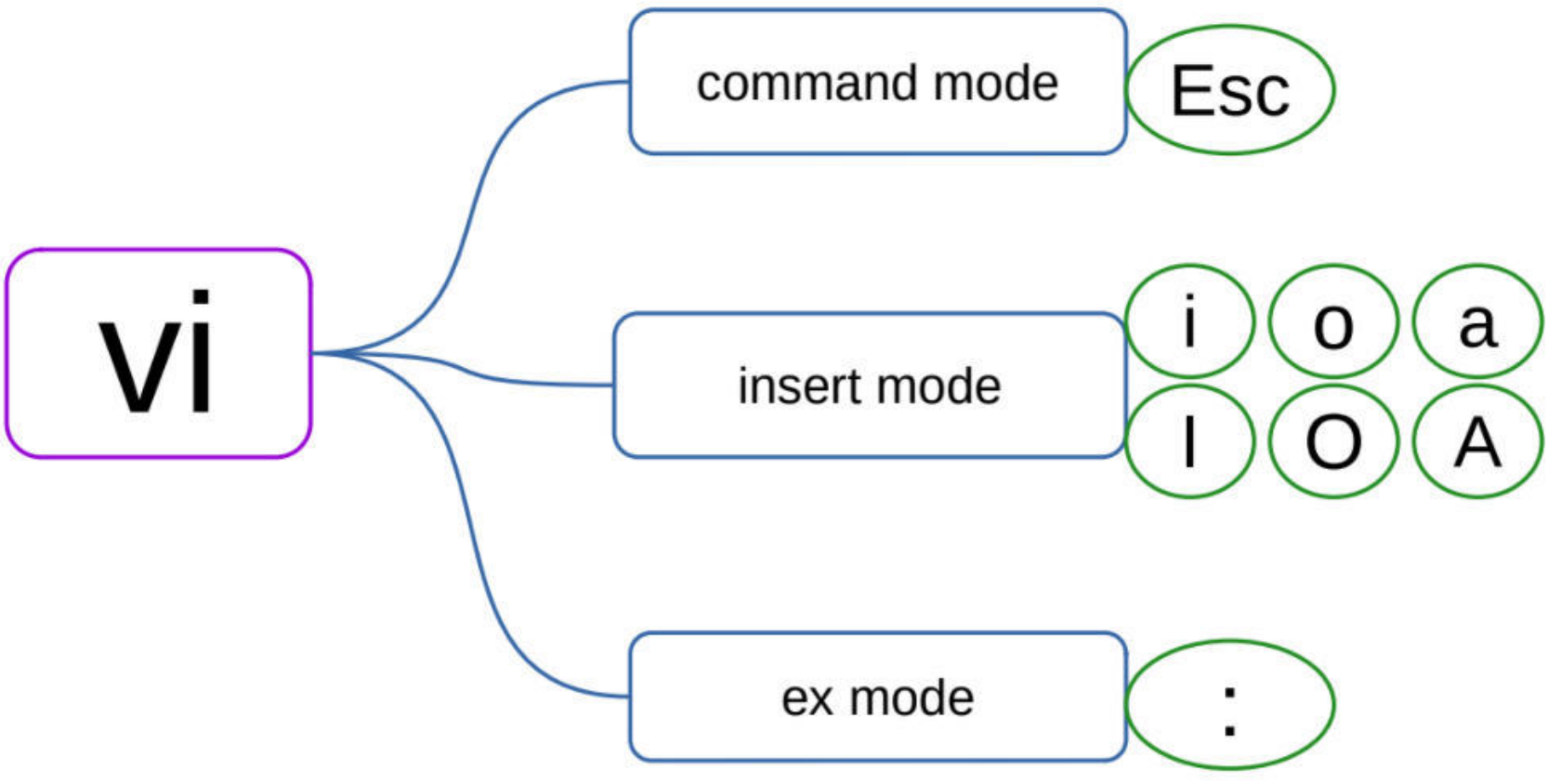
Information

Ctrl+C	Report cursor position
Alt+D	Report line/word/character count
Ctrl+G	Display help text

Various	
Alt+A	Turn the mark on/off
Tab	Indent marked region
Shift+Tab	Unindent marked region
Alt+V	Enter next keystroke verbatim
Alt+N	Turn line numbers on/off
Alt+P	Turn visible whitespace on/off
Alt+X	Hide or unhide the help lines
Ctrl+L	Refresh the screen

Source: https://www.nano-editor.org/dist/latest/cheatsheet.html

Modes in `vi` editor



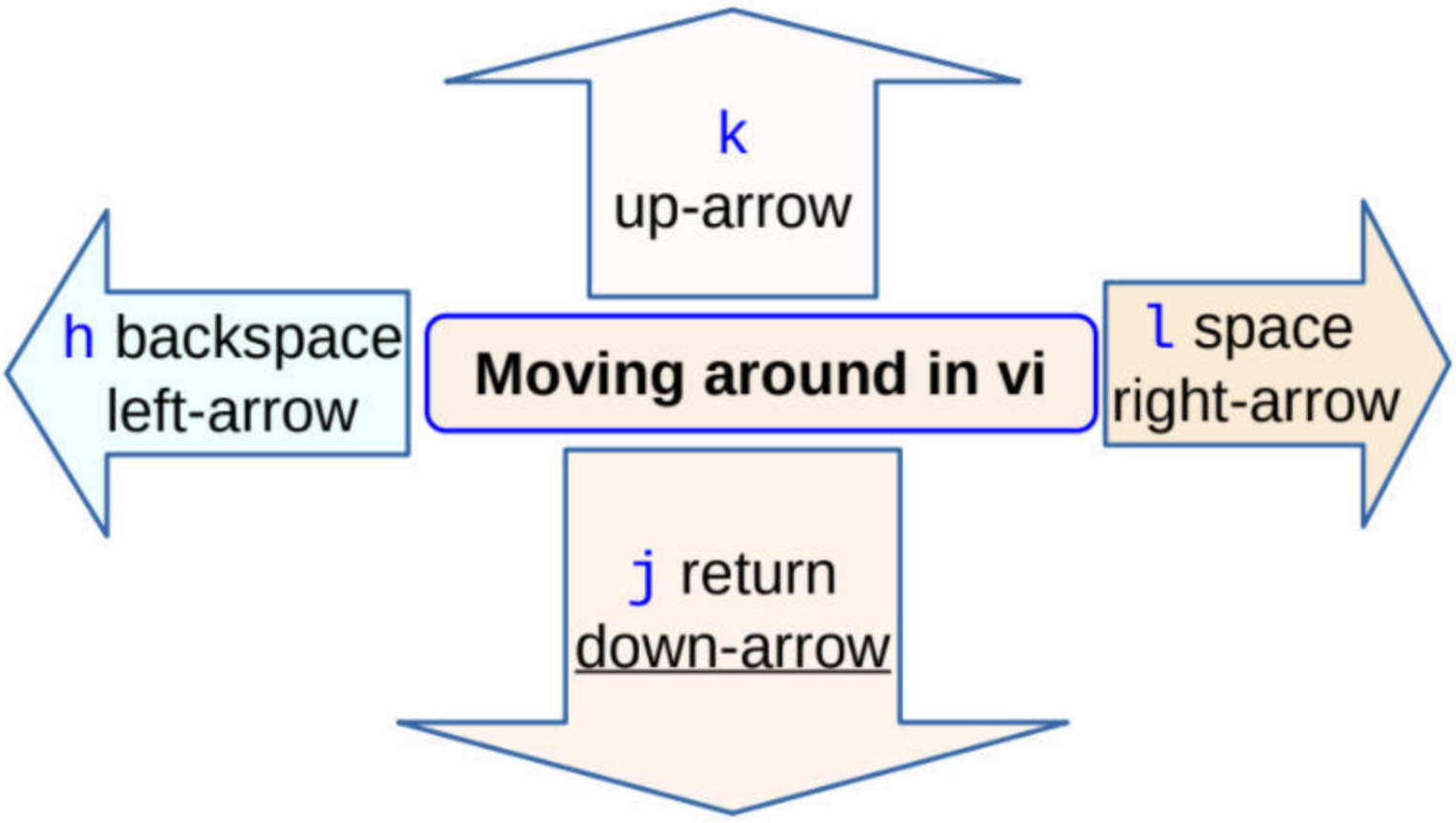
`i` will insert the characters from the current position of the cursor

`o` will insert a new line

`a` will append the text

`vi` help

- **These work only in the Command mode**
  - Press `Esc` to enter this mode
- To exit out of `vi`
  - `:w` → write out
  - `:x` → write out and quit
  - `:wq` → write out and quit
  - `:q` → quit (if write out is over)
  - `:q!` → ignore changes and quit
  - *If nothing else works, cry and smash your keyboard, and then rage quit*



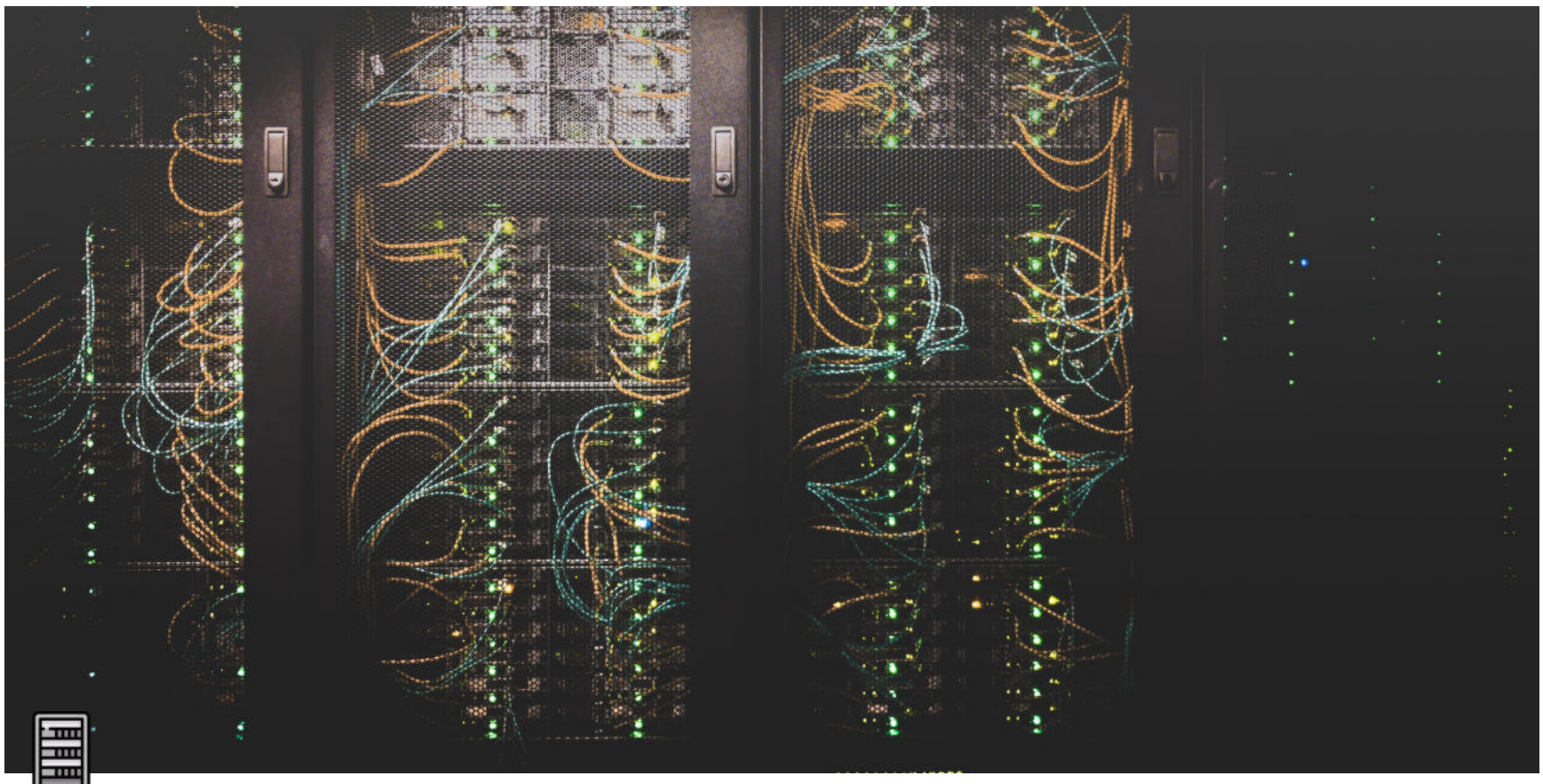
`vi` command mode

- Screen manipulation
  - `Ctrl + F` → Scroll forward one screen
  - `Ctrl + B` → Scroll backward one screen
  - `Ctrl + D` → Scroll down half screen
  - `Ctrl + U` → Scroll up half screen
  - `Ctrl + L` → Redraw screen
  - `Ctrl + R` → Redraw screen removing deleted stuff
- Moving around
  - `0` → Start of the current line
  - `$` → End of the current line
  - `w` → Beginning of the next word
  - `b` → Beginning of the preceding word
  - `:0` → First line in the file
  - `1G` → First line in the file
  - `:n` → n-th line in the file
  - `nG` → n-th line in the file
  - `:$` → Last line in the file
  - `G` → Last line in the file
- Changing text
  - `r` → Replace a single character under the cursor
  - `R` → Replace characters from the cursor till `Esc`
  - `cw` → Change word under the cursor, from the current character till `Esc`
  - `cNw` → Change **N** words, from the current character till `Esc`
  - `C` (CAPITAL C) → Change characters in the current line till `Esc`
  - `cc` (small c) → Change the line till `Esc`
  - `Ncc` → Change the next **N** lines, starting from the current till `Esc`
- Deleting text

    - `x` → Delete a single character under the cursor
    - `Nx` → Delete **N** characters from the cursor
    - `dw` → Delete one word, from the character under the cursor
    - `dNw` → Delete **N** words, from the character under the cursor
    - `D` → Delete the rest of the line, from the character under the cursor
    - `dd` → Delete the current line
    - `Ndd` → Delete the next N lines, starting from the current one
- Copy and Paste text
    - `yy` (small y) → Copy the current line to the buffer
    - `Nyy` → Copy the next N lines, including the current, into the buffer
    - `p` → Paste buffer into the file after the current line
    - `u` → Undo the previous action
- Searching text
    - `/string` → Search forward for the given `string`
    - `?string` → Search backward for the given `string`
    - `n` → Move the cursor to the next occurance of the `string`
    - `N` → Move the cursor to the previous occurrance of the `string`

`:se nu` → Set line numbers

`:se nonu` → Unset the line numbers



Command Line Editors - pt. 3

Type	📒 Lecture
Date	@February 2, 2022
Lecture #	3
Lecture URL	https://youtu.be/w25zlMXshHw
Notion URL	https://21f1003586.notion.site/Command-Line-Editors-pt-3-5452a0b89e32403184ce432b0bfd3b5a
Week #	5

To copy a file using secure copy `scp`

Syntax → `scp <username>@<IP_ADDRESS>:<path/to/file/on/that/machine> <where/to/save>`

Example → `scp gphani@10.17.0.167:Documents/code3d.tar .`

To untar (extract) a `.tar` file

Syntax → `tar -xvf filename.tar`

`emacs`

C-x means `Ctrl + x`

M-x means `Alt + x`

- Moving around
  - `C-p` → Move up by one line
  - `C-b` → Move left by one character
  - `C-f` → Move right by one character
  - `C-n` → Move down by one line
  - `C-a` → Go to the beginning of the current line
  - `C-e` → Go to the end of the current line
  - `C-v` → Move forward one screen
  - `M-<` → Move to the first line of the file
  - `M-b` → Move left to the previous word
  - `M-f` → Move right to the next word
  - `M->` → Move to the last line of the file

- `M-a` → Move to the beginning of the current sentence
- `M-e` → Move to the end of the current sentence
- `M-v` → Move back one screen

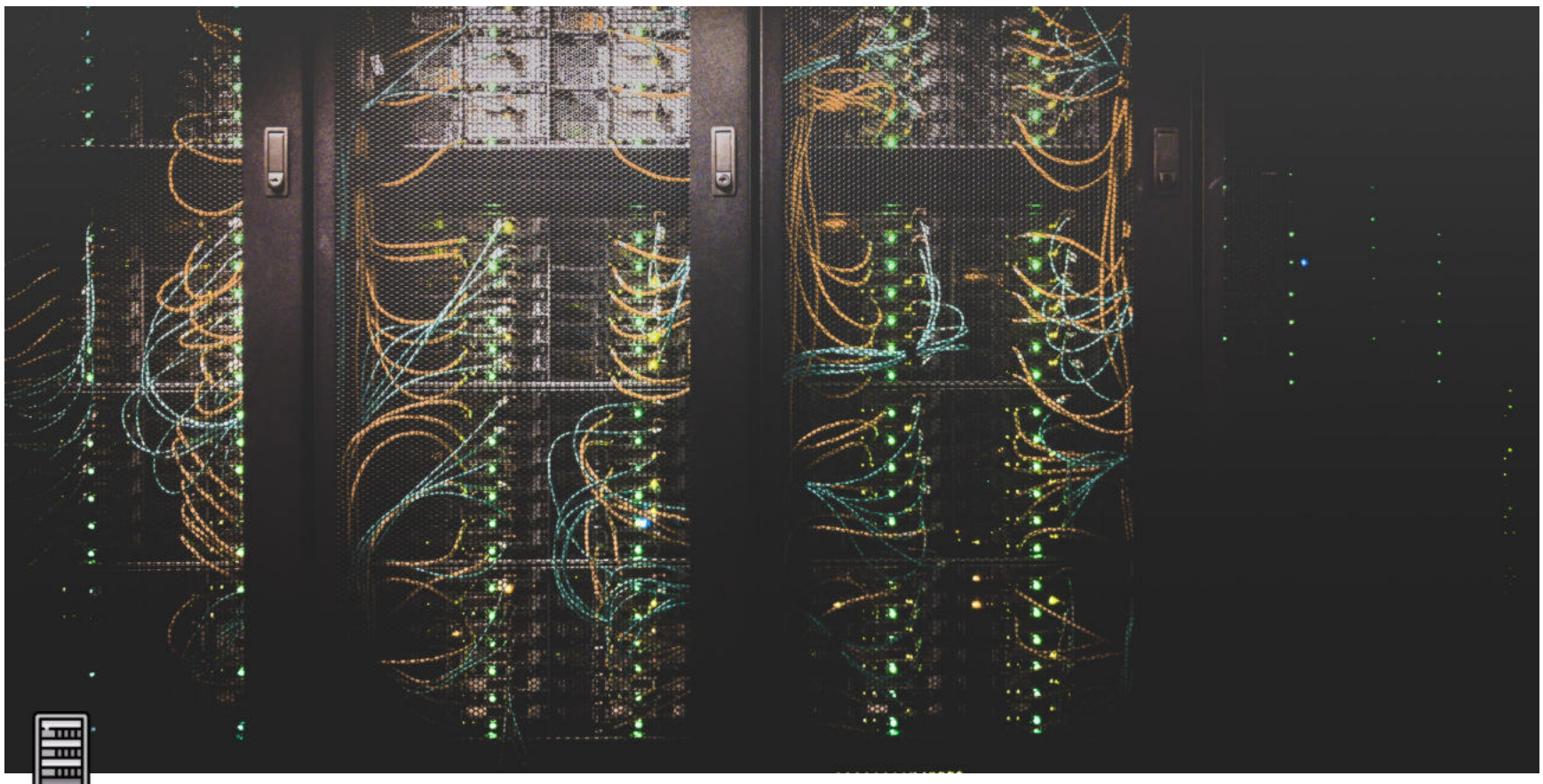
Source: https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

`emacs` commands

- Exiting `emacs`
  - `C-x C-s` → Save buffer to the file
  - `C-z` → Exit emacs but keep it running
  - `C-x C-c` → Exit emacs and stop it
- Searching a text
  - `C-s` → Search forward
  - `C-r` → Search backward
  - `M-x` → Replace string
- Copy and Paste
  - `M-backspace` → Cut the word before the cursor
  - `M-d` → Cut the word after the cursor
  - `C-k` → Cut from the cursor to the end of the line
  - `M-k` → Cut from the cursor to the end of the sentence
  - `C-y` → Paste the content at the cursor

Syntax for emacs → `emacs -nw filename`

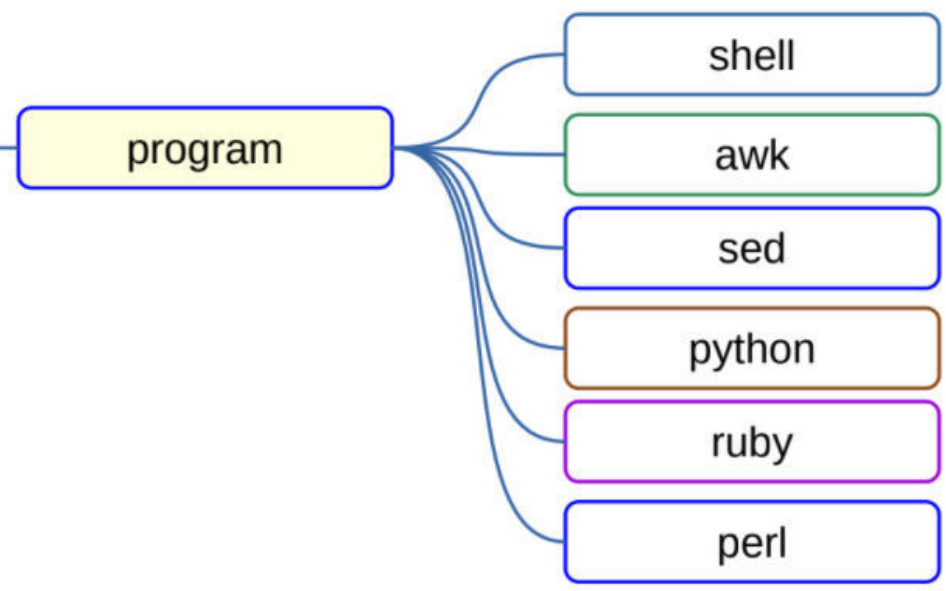
Make sure to pass the `-nw` flag to make it open in terminal mode instead of the GUI

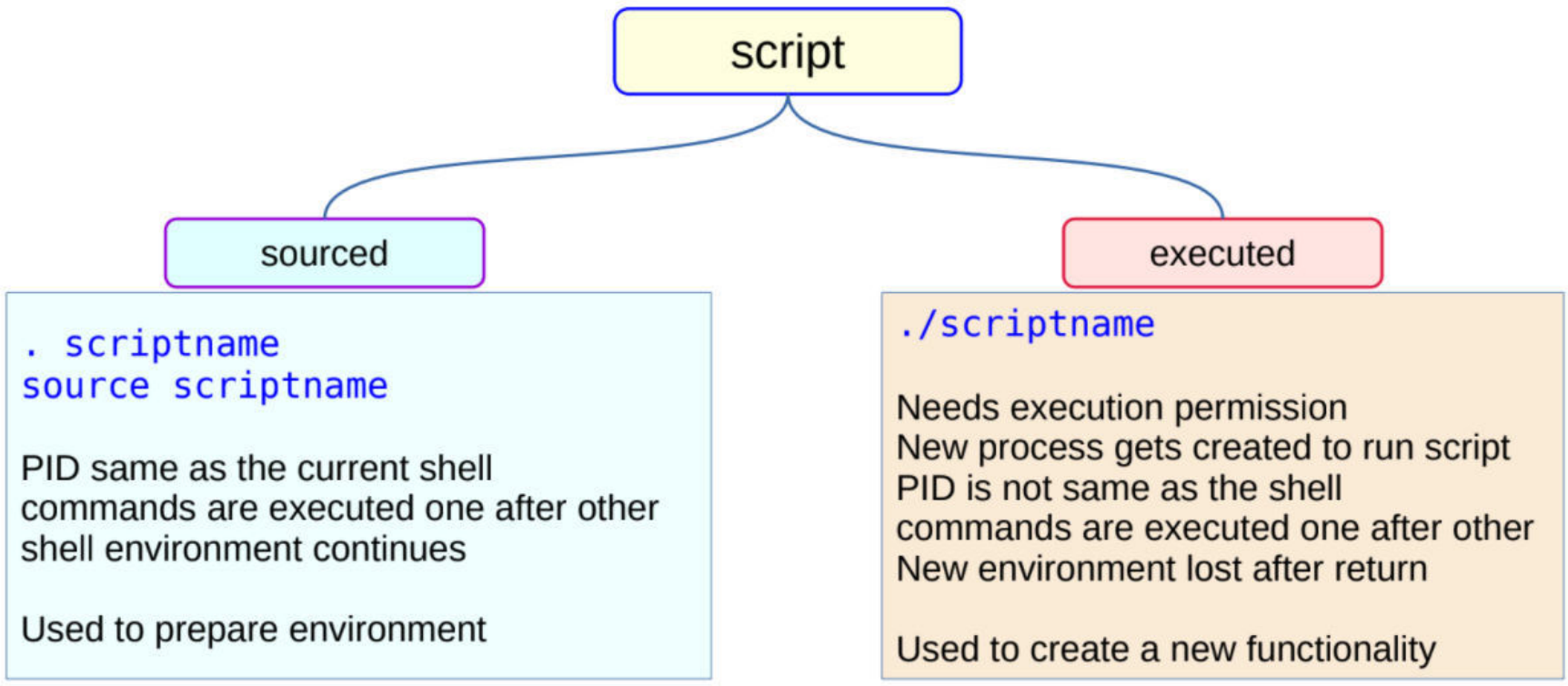


Overview of shell scripts

Type	📒 Lecture
Date	@February 2, 2022
Lecture #	4
Lecture URL	https://youtu.be/YAddHeVpG7I
Notion URL	https://21f1003586.notion.site/Overview-of-shell-scripts-2d85d6d2542f4122b49db652e3913de8
Week #	5

```
#! interpreter
# comments
commands
loops
variables
case statements
functions
```



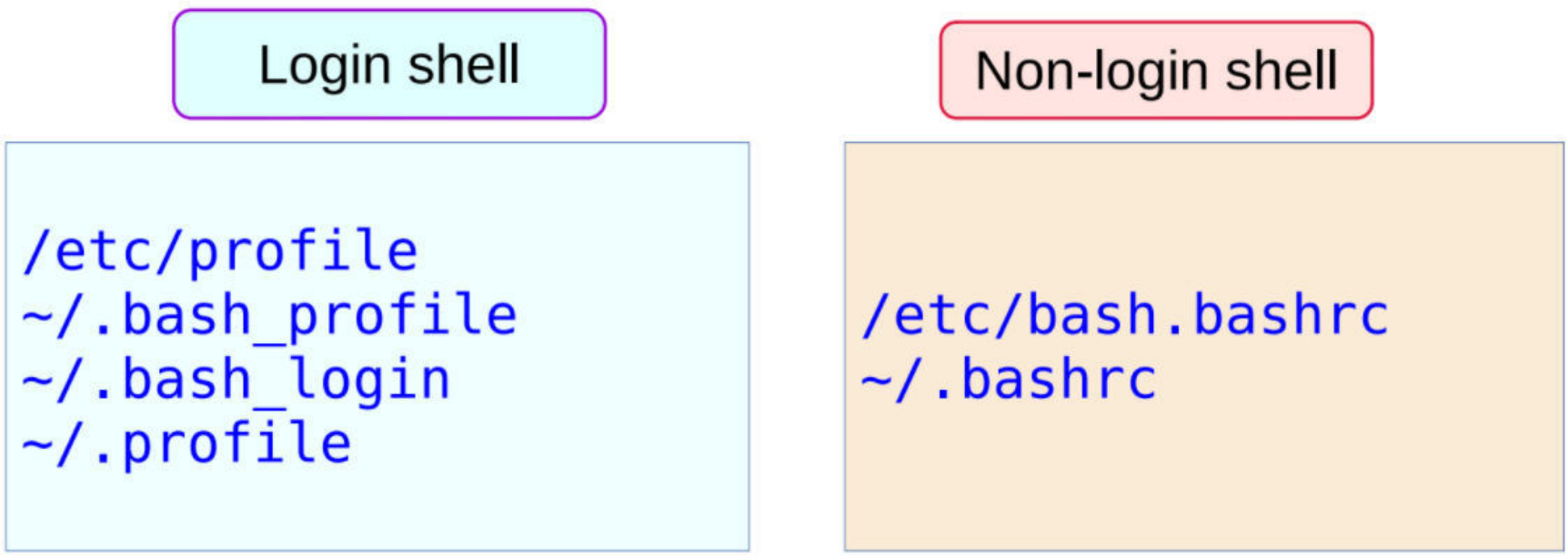


Script location

- Use absolute path or relative path while executing the script
- Keep the script in the folder listed in `$PATH`
- Watch out for the sequence of directories in `$PATH`

bash environment

- When you are already logged in to a system and using the GNOME environment to open a terminal, then the shell we are opening is not asking for a login and is called a **Non-login shell**
- When we ssh into a remote machine, then the shell we are logging into would have checked for the login credentials and then opened up the command line environment for us, therefore that shell is called a **Login shell**



Output from the shell scripts

- `echo`
  - Terminates with a newline if `-n` flag is not given
  - `echo My home is $HOME`
- `printf`
  - Supports format specifiers like in C
  - `printf “My home is %s\n” $HOME`

Input to shell scripts

- `read var`
  - String read from command line is stored in `$var`

Shell script arguments

`./myscript.sh -l arg2 -v arg4`

- `$0` → name of the shell program
- `$#` → number of arguments passed
- `$1` or `${1}` → first argument
- `${11}` → eleventh argument
- `$*` or `$@` → all arguments at once
- `"$*"` → all arguments as a single string
- `"$@"` → all arguments as separate strings

Command substitution

```
var=`command`
var=$(command)
```

`command` is executed and the output is substituted

Here, the variable `var` will be assigned with that output

`for do` loop

```
for var in list
do
  commands
done
```

`commands` are executed once for each item in the `list`

space is the field delimiters

set IFS if required

`case` statement

```
case var in
pattern1)
  commands
  ;;
pattern2)
  commands
  ;;
esac
```

`commands` are executed

each pattern matched

for var in the options

`if` loop

```
if condition
then
  commands
fi
```

```
if condition; then
  commands
fi
```

`commands` are executed only if `condition` returns true

`conditions` in `if`

- `test -e file` or any `text expression`

    - True, if the file exists in the current directory of the shell script
    - else False
- `[ -e file ]` or any `[ expression ]`
    - Same as the above one, just a different syntax
- `[[ expression ]]` like `[[ $ver == 5.* ]]`
    - If we intend to use regex or other complex operations
- `(( expression ))` like `(( $v ** 2 > 10 ))`
    - To perform complex arithmetic operations for the test conditions
- Or just use a `command` like `wc -l file`
    - If the command returns True, i.e. upon successful execution, this means the condition is satisfied
    - else False
- `pipeline` like `who | grep "joy" > /dev/null`
    - A set of commands which can be combined with other commands and redirection of output and combination of logical expressions using the `&&` or `||`
    - These can be put as the condition

For negation, use `!` before the condition

test numeric comparisons

- `$n1 -eq $n2`
    - Check if n1 is equal to n2
- `$n1 -ge $n2`
    - Check if n1 is greater than or equal to n2
- `$n1 -gt $n2`
    - Check if n1 is greater than n2
- `$n1 -le $n2`
    - Check if n1 is less than or equal to n2
- `$n1 -lt $n2`
    - Check if n1 is less than n2
- `$n1 -ne $n2`
    - Check if n1 is not equal to n2

test string comparisons

- `$str1 = $str2`
    - Check if str1 is the same as str2
- `$str1 != $str2`
    - Check if str1 is not the same as str2
- `$str1 < $str2`
    - Check if str1 is less than str2
    - Compares based on lexicographical (alphabetical) order
- `$str1 > $str2`
    - Check if str1 is greater than str2
    - Compares based on lexicographical (alphabetical) order
- `-n $str1`
    - Check if str1 has length greater than zero

- `-z $str1`
  - Check if str1 has the length zero

Unary file comparisons

- `-e file`
  - Check if the file exists
- `-d file`
  - Check if the file exists and is a directory
- `-f file`
  - Check if the file exists and is a file
- `-r file`
  - Check if the file exists and is readable
- `-s file`
  - Check if the file exists and is not empty
- `-w file`
  - Check if the file exists and is writable
- `-x file`
  - Check if the file exists and is executable
- `-O file`
  - Check if the file exists and is owned by the current user
- `-G file`
  - Check if the file exists and the default group is the same as that of the current user

Binary file comparisons

- `file1 -nt file2`
  - Check if file1 is newer than file2
- `file1 -ot file2`
  - Check if file1 is older than file2

`while do` loop

```
while condition
do
  commands
done
```

`commands` are executed only if the `condition` returns `true`

`until do` loop

```
until condition
do
  commands
done
```

`commands` are executed only if the `condition` returns `false`

functions

definition

```
myfunc()
{
```

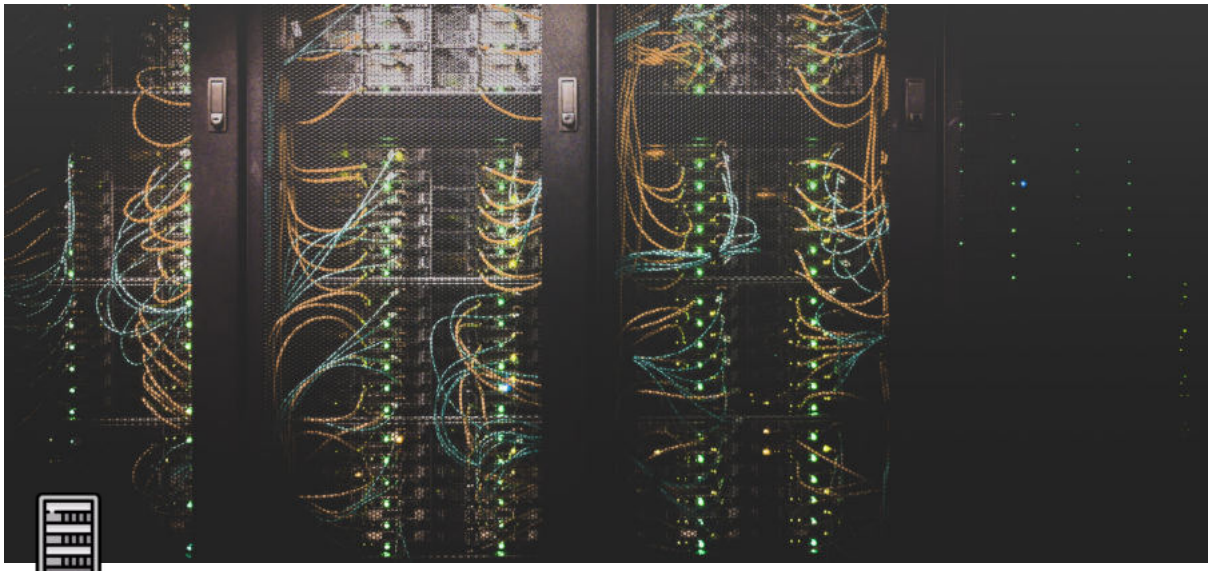
```
  commands
}
```

call

```
myfunc
```

`commands` are executed each time `myfunc` is called

Definitions must be before the calls



Bash scripts - pt. 2A

Type	📒 Lecture
Date	@February 9, 2022
Lecture #	1
Lecture URL	https://youtu.be/8YmNovNo6NE
Notion URL	https://21f1003586.notion.site/Bash-scripts-pt-2A-f7361a999fa344499488a73d6db381b1
Week #	6

Debugging

```
set -x
./myscript.sh
```

Prints the command before executing it

Place the `set -x` inside the script

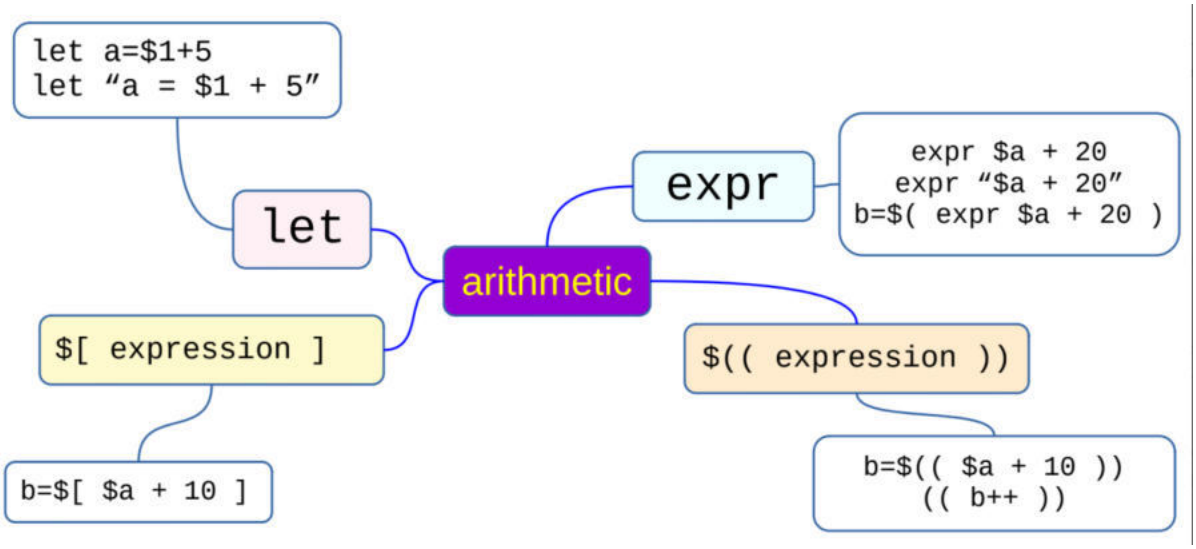
`bash -x ./myscript.sh`

Combining conditions

```
[ $a -gt 3 ] && [ $a -lt 7 ]
```

```
[ $a -le 3 ] || [ $a -ge 7 ]
```

Shell arithmetic



`expr` command operators

a + b	Return arithmetic sum of a and b
a - b	Return arithmetic difference of a and b
a * b	Return arithmetic product of a and b
a / b	Return arithmetic quotient of a divided by b
a % b	Return arithmetic remainder of a divided by b
a > b	Return 1 if a greater than b; else return 0
a >= b	Return 1 if a greater than or equal to b; else return 0
a < b	Return 1 if a less than b; else return 0
a <= b	Return 1 if a less than or equal to b; else return 0
a = b	Return 1 if a equals b; else return 0

`a | b`	Return a if neither argument is null or 0; else return b
`a & b`	Return a if neither argument is null or 0; else return 0
`a != b`	Return 1 if a is not equal to b; else return 0
`str : reg`	Return the position upto anchored pattern match with BRE str
`match str reg`	Return the pattern match if reg matches pattern in str
`substr str n m`	Return the substring m chars in length starting at position n
`index str chars`	Return position in str where any one of chars is found; else return 0
`length str`	Return numeric length of string str
`+ token`	Interpret token as string even if its a keyword
`(exprn)`	Return the value of expression exprn

```
#!/bin/bash

if [ $# -lt 2 ]; then
  echo "Use 2 natural numbers as arguments";
  exit 1;
fi;

regex='^[0-9]+$'
if ! [[ $1 =~ $regex ]]; then
    echo "$1 is not a natural number";
    exit 1;
fi;

if ! [[ $2 =~ $regex ]]; then
    echo "$2 is not a natural number";
    exit 1;
fi;

let a=$1*$2;
echo "Product a is $a";
(( a++ ));
echo "Product a incremented is $a";

let b=$1**$2;
echo "Power is $b";

c=$[ $1 + $2 + 10 ];
echo "sum + 10 is $c";

d=$(expr $1 + $2 + 20);
echo "sum + 20 is $d";

f=$(( $1 * $2 * 2 ));
echo "Product times 2 is $f";
```

```
#!/bin/bash
# The following line is for debugging
# set -x;

# Code starts here
a=256;
b=4;
c=3;

ans=$( expr $a + $b );
echo $ans;

ans=$( expr $a - $b );
echo $ans;

ans=$( expr $a \* $b );
echo $ans;

ans=$( expr $a / $b );
echo $ans;

ans=$( expr $a % $c );
echo $ans;

ans=$( expr $a \> $b );
echo $ans;

ans=$( expr $a \>= $b );
echo $ans;

ans=$( expr $a \< $b );
echo $ans;

ans=$( expr $a \<= $b );
echo $ans;

ans=$( expr $a = $b );
echo $ans;

ans=$( expr $a != $b );
echo $ans;

ans=$( expr $a \| $b );
echo $ans;

ans=$( expr $a \& $b );
echo $ans;

str="octavio version as in Jan 2022 is 6.4.0";
reg="[oO]ctav[aeiou]*";
ans=$( expr "$str" : $reg );
echo $ans;

ans=$( expr substr "$str" 1 6 );
echo $ans;
```

```
ans=$( expr index "$str" "vw" );
echo $ans;

ans=$( expr length "$str" );
echo $ans;
```

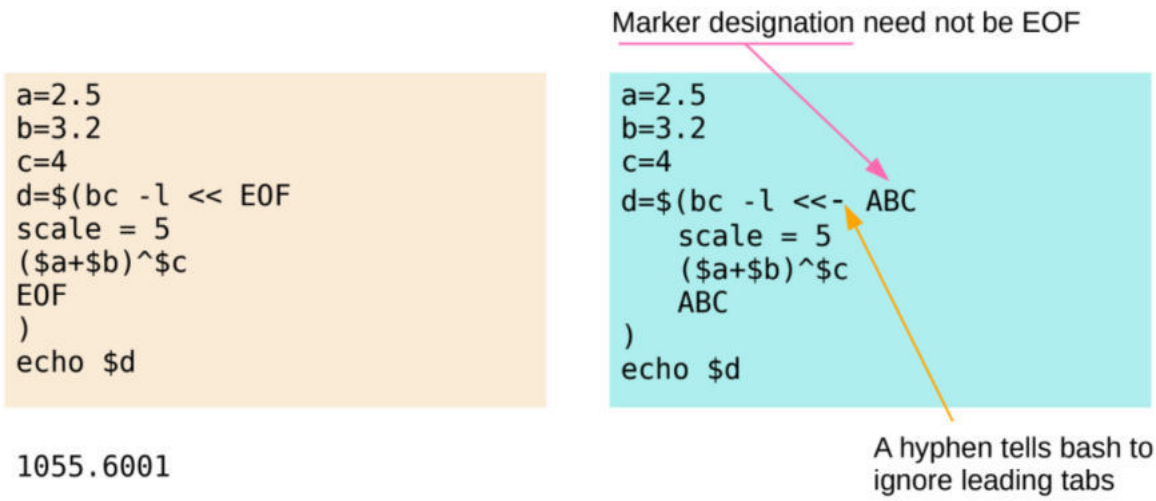
`heredoc` feature

When writing shell scripts you may be in a situation where you need to pass a multiline block of text or code to an interactive command, such as `tee`, `cat`, or `sftp`

In `bash` and other shells like `zsh`, a Here document (`heredoc`) is a type of redirection that allows you to pass multiple lines of input to a command.

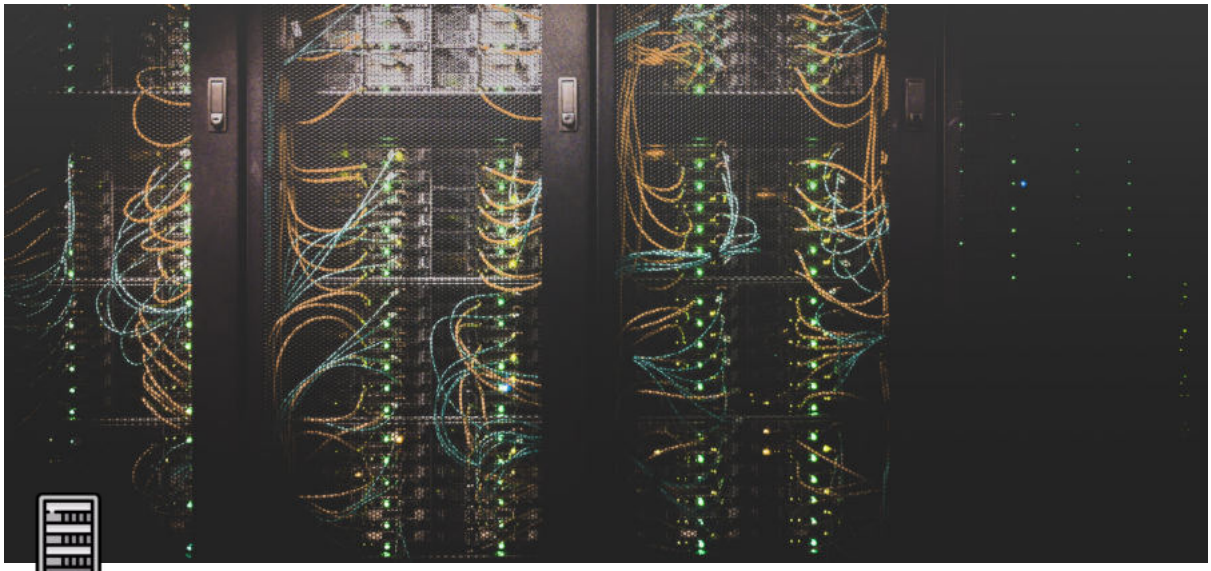
This is what a general syntax looks like

```
[COMMAND] <<[-] 'DELIMITER'
  HERE-DOCUMENT
DELIMITER
```



Notice no-indent (left) vs indentation (right) in the above example

```
#!/bin/bash

# set -x;
echo "path is set as $PATH";
i=0;
IFS=:;

for n in $PATH; do
    echo "$i $n";
    (( i++ ));
done;
```



Bash scripts - pt. 2B

Type	📒 Lecture
Date	@February 9, 2022
Lecture #	2
Lecture URL	https://youtu.be/n56JIC15DBo
Notion URL	https://21f1003586.notion.site/Bash-scripts-pt-2B-caebca1e09d24acfba760344452db02d
Week #	6

`if-elif-else-fi` loop

```
if condition1
then
    commandset1
else
    commandset2
fi
```

```
if condition1
then
    commandset1
elif condition2
then
    commandset2
elif condition3
then
    commandset3
else
    commandset4
fi
```

```
#!/bin/bash

if [ $# -gt 2 ]; then
    echo "More than 2 arguments";
elif [ $# -gt 1 ]; then
    echo "More than 1 argument";
elif [ $# -gt 0 ]; then
    echo "Not enough arguments";
else
    echo "Arguments required";
fi;
```

`case` statement options

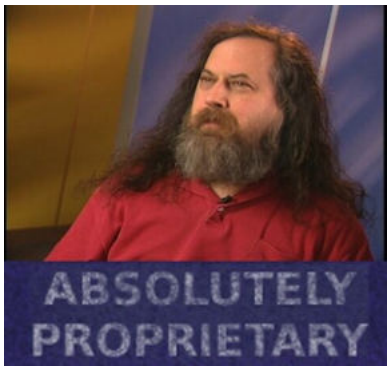
```
case $var in
    op1)
        commandset1;;
    op2 | op3)
        commandset2;;
    op4 | op5 | op6)
        commandset3;;
    *)
        commandset4;;
esac
```

commandset4 is the default for values of `$var` not matching what are listed

```
#!/bin/bash

echo "What is your favourite image processor?";
read pname;
```

```
case $pname in
    [gG]imp | inkscape)
        echo "Good choice";
        ;;
    [aA]dobe*)
        echo "Absolutely proprietary and costs a lot";
        ;;
    imagej)
        echo "Measuring things on the image?";
        ;;
    *)
        echo "$pname is a new find for me";
        ;;
esac;
```

When you enter “adobe”



C-style `for` loop → one variable

```
begin=1
finish=10
for (( a = $begin; a < $finish; a++ ))
do
   echo $a
done
```

```
#!/bin/bash

begin=1;
finish=10;
```

```
for (( a = $begin; a < $finish; a++ )); do
    b=$(( a**2 ));
    echo $b;
done;
```

C-style `for` loop → two variables

```
begin1=1
begin2=10
finish=10
for (( a=$begin1, b=$begin2; a < $finish; a++, b-- ))
do
   echo $a $b
done
```

NOTE: only one condition to close out the for loop

```
#!/bin/bash

begin1=1;
begin2=20;
finish=10;

for (( a = $begin1, b = $begin2; a < $finish; a++, b-- )); do
    c=$(( a**2 ));
    d=$(( b**2 ));
    echo $c $d;
done;
```

Processing output of a loop

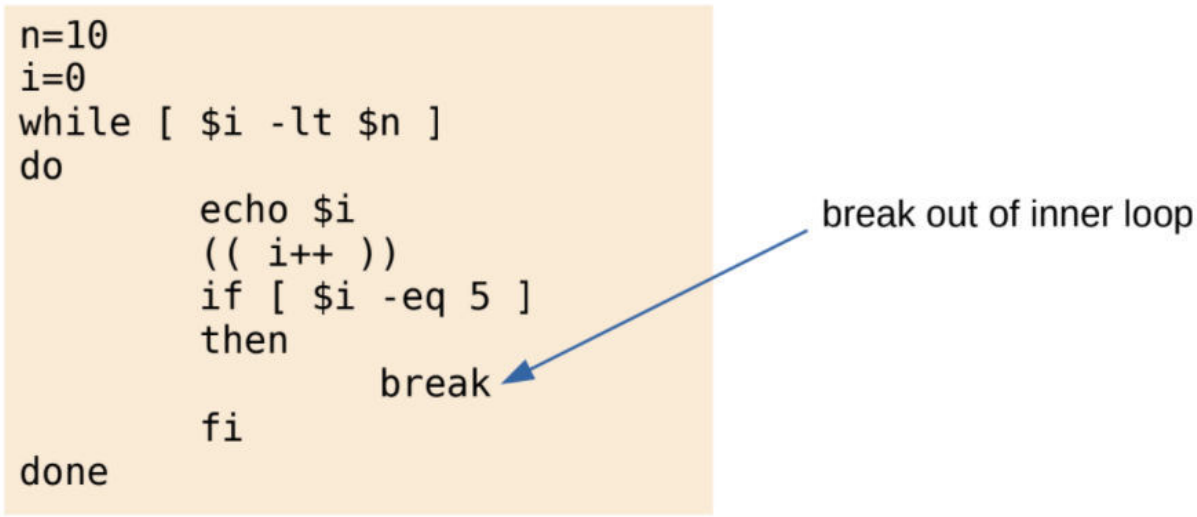
```
filename=tmp.$$
begin=1
finish=10
for (( a = $begin; a < $finish; a++ ))
do
   echo $a
done > $filename
```

NOTE: Output of the loop is re-directed to the tmp file

```
#!/bin/bash

filename=largefile.txt;
if [ -e $filename ]; then
    echo "file $filename exists";
    exit 1;
fi;

i=1;
while [ $i -lt 10 ]; do
    echo "$i $[$i+1]";
    (( i++ ));
done > $filename;

echo "file $filename written";
ls -l $filename;
```

`break`

To break out of a loop



```
n=10
i=0
while [ $i -lt $n ]
do
        echo $i
        j=0
        while [ $j -le $i ]
        do
                printf "$j "
                (( j++ ))
                if [ $j -eq 7 ]
                then
                        break 2
                fi
        done
        (( i++ ))
done
```

break out of outer loop

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6
```

`break 2` refers to the outer loop as seen from the nesting

`continue`

Opposite of `break`

```
n=9
i=0
while [ $i -lt $n ]
do
        printf "\n loop $i:"
        j=0
        (( i++ ))
        while [ $j -le $i ]
        do
                (( j++ ))
                if [ $j -gt 3 ] && [ $j -lt 6 ]
                then
                        continue
                fi
                printf "$j "
        done
done
```

```
loop 0:1 2
loop 1:1 2 3
loop 2:1 2 3
loop 3:1 2 3
loop 4:1 2 3 6
loop 5:1 2 3 6 7
loop 6:1 2 3 6 7 8
loop 7:1 2 3 6 7 8 9
loop 8:1 2 3 6 7 8 9 10
```

Continue will skip rest of the commands in the loop and goes to next iteration

`shift`

```
i=1
while [ -n "$1" ]
do
        echo argument $i is $1
        shift
        (( i++ ))
done
```

shift will shift the command line arguments by one to the left.

It keeps on shifting (read: delete/destroy) the arguments to the left

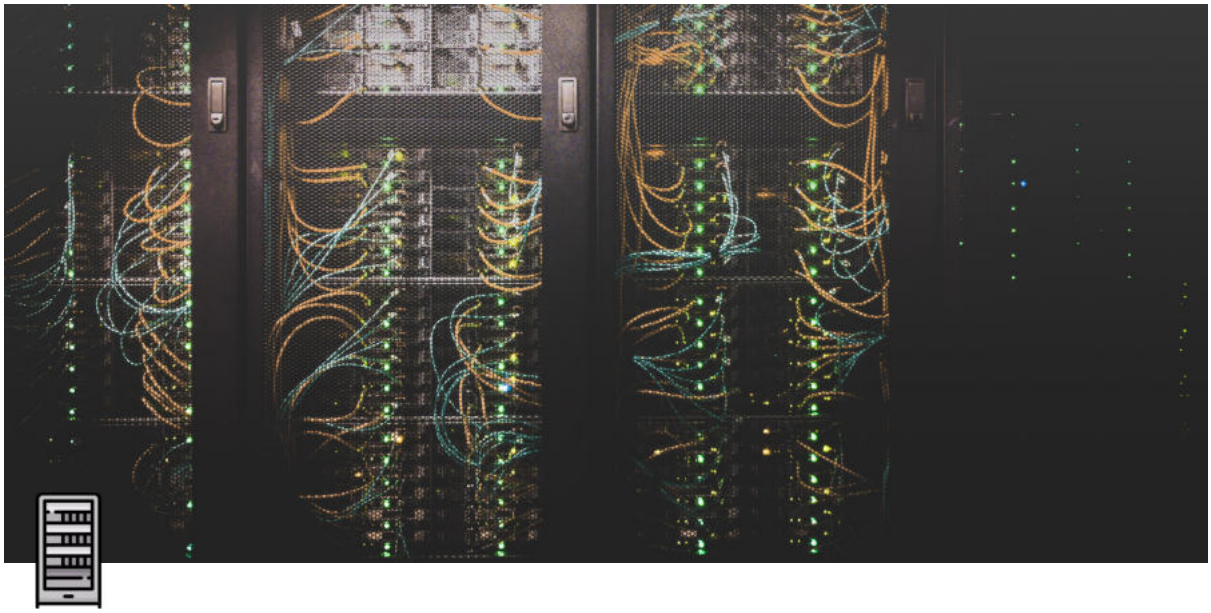
```
#!/bin/bash

echo "Number of args: ";
echo $#;
i=1;

while [ -n "$1" ]; do
    echo "arg $i is $1";
    shift;
    (( i++ ));
done;

echo "Number of args now: ";
echo $#;
```

exec

`exec ./my-executable --my-options --my-args`

- To replace shell with a new program or to change i/o settings
- If new program is launched successfully, it will not return control back to the shell
- If new program fails to launch, the shell continues

```
#!/bin/bash
echo "PID of shell running this command: $$";
echo  "Leaving bash and opening xterm if available";
exec xterm;
echo "Looks like xterm is not available or failed to start";
```



Bash scripts - pt. 2C

Type	📒 Lecture
Date	@February 9, 2022
Lecture #	3
Lecture URL	https://youtu.be/kjMhi4gI0po
Notion URL	https://21f1003586.notion.site/Bash-scripts-pt-2C-22f3a0910ba04a088ce5281aab4462e4
Week #	6

`eval`

`eval my-arg`

- Executes argument as a shell command
- Combines arguments into a single string
- Returns control to the shell with exit status

```
#!/bin/bash
cmd="date";
```

```
fmt="+%d-%B-%Y";
eval $cmd $fmt;
```

Functions in bash

```
#!/bin/bash

usage() {
  echo "usage $1 str1 str2";
}

swap() {
  echo "$2 $1";
}

if [ $# -lt 2 ]; then
  usage $0;
  exit 1;
fi;

swap $1 $2;
```

`getopts`

```
while getopts "ab:c:" options;
do
    case "${options}" in
        b)
            barg=${OPTARG}
            echo accepted: -b $barg
            ;;
        c)
            carg=${OPTARG}
            echo accepted: -c $carg
            ;;
        a)
            echo accepted: -a
            ;;
        *)
            echo Usage: -a -b barg -c carg
            ;;
    esac
done
```

This script can be invoked with only three options: a, b, c. The options b and c will take arguments.

```
#!/bin/bash
while getopts "ab:c:" options; do
  case "${options}" in
    b)
      barg=${OPTARG};
      echo "accepted: -b $barg";
      ;;
```

```
    c)
      carg=${OPTARG};
      echo "accepted: -c $carg";
      ;;
    a)
      echo "accepted: -b";
      ;;
    *)
      echo "Usage: -a -b barg -c carg";
      ;;
  esac;
done;
```

Usage

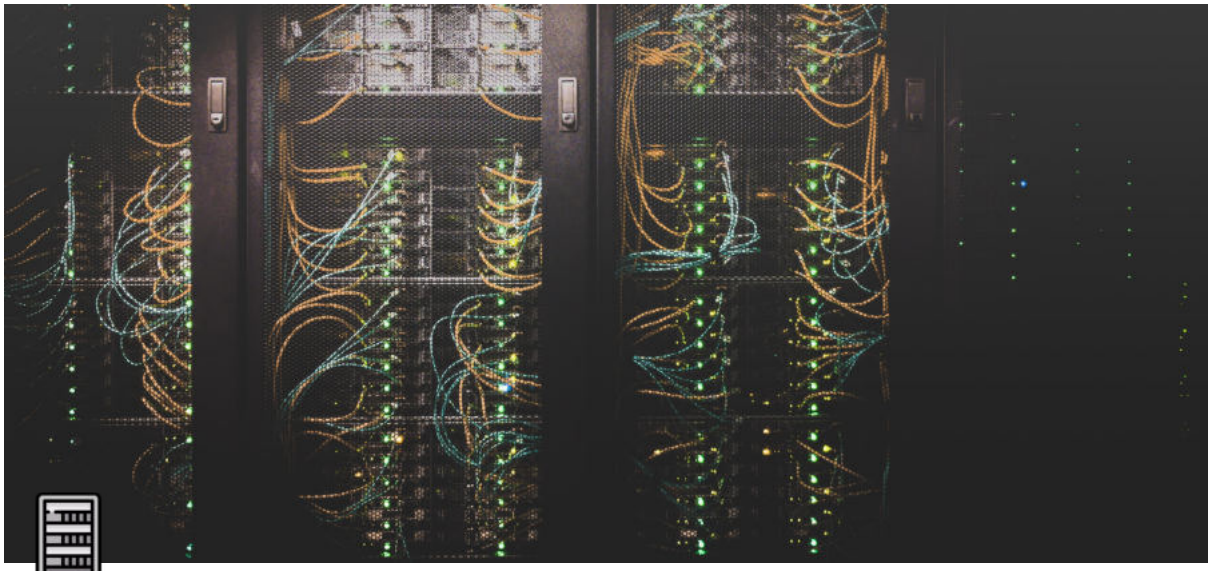
```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec1
$ bash getopts.sh -a -b bargument -c cargument
accepted: -b
accepted: -b bargument
accepted: -c cargument

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec1
$ bash getopts.sh -a -b bargument -c cargument -d -e
accepted: -b
accepted: -b bargument
accepted: -c cargument
getopts.sh: illegal option -- d
Usage: -a -b barg -c carg
getopts.sh: illegal option -- e
Usage: -a -b barg -c carg
```

`select` loop

```
echo select a middle one
select i in {1..10}
do
        case $i in
                1 | 2 | 3)
                        echo you picked a small one;;
                8 | 9 | 10)
                        echo you picked a big one;;
                4 | 5 | 6 | 7)
                        echo you picked the right one
                        break;;
        esac
done
echo selection completed with $i
```

Text menu !

```
#!/bin/bash

echo "select a middle one";
select i in {1..10}; do
    case $i in
        1 | 2 | 3)
            echo "you picked a small one";
            ;;
        8 | 9 | 10)
            echo "you picked a big one";
            ;;
        4 | 5 | 6 | 7)
            echo "you picked the right one";
            break;
            ;;
    esac;
done;

echo "selection completed with $i";
```



awk programming - pt. 1

Type	📒 Lecture
Date	@February 9, 2022
Lecture #	4
Lecture URL	https://youtu.be/k3q-9ntZI4s
Notion URL	https://21f1003586.notion.site/awk-programming-pt-1-63f859576b6c4836a7cc2ed96ac4c99b
Week #	6

`awk`

A language for processing fields and records

Introduction

- It is a programming language
- `awk` is an abbreviation of the 3 people who developed it
  - **A**ho
  - **W**einberger

    - **K**ernighan
- It is a part of POSIX, IEEE 1003.1-2008
- Variants of `awk`
    - `nawk`
    - `gawk`
    - `mawk`
    - `...`
- `gawk` contains features that extend POSIX

Execution model

- Input stream is a set of records
    - Eg. using `"\n"` as a record separator, lines are recorded
- Each record is a sequence of fields
    - Eg. using `" "` as a field separator, words are fields
- Splitting of records to fields is done automatically
- Each codeblock executes on one record at a time, as matched by the pattern of that block

usage

- Single line at the command line

  `cat /etc/passwd | awk -F":" '{print $1}'`
- Script interpreted by awk

  `./myscript.awk /etc/passwd`

  `myscript.awk` 👇

```
#!/usr/bin/gawk -f
BEGIN {
  FS=":"
}
{
  print $1
}
```

Example #1

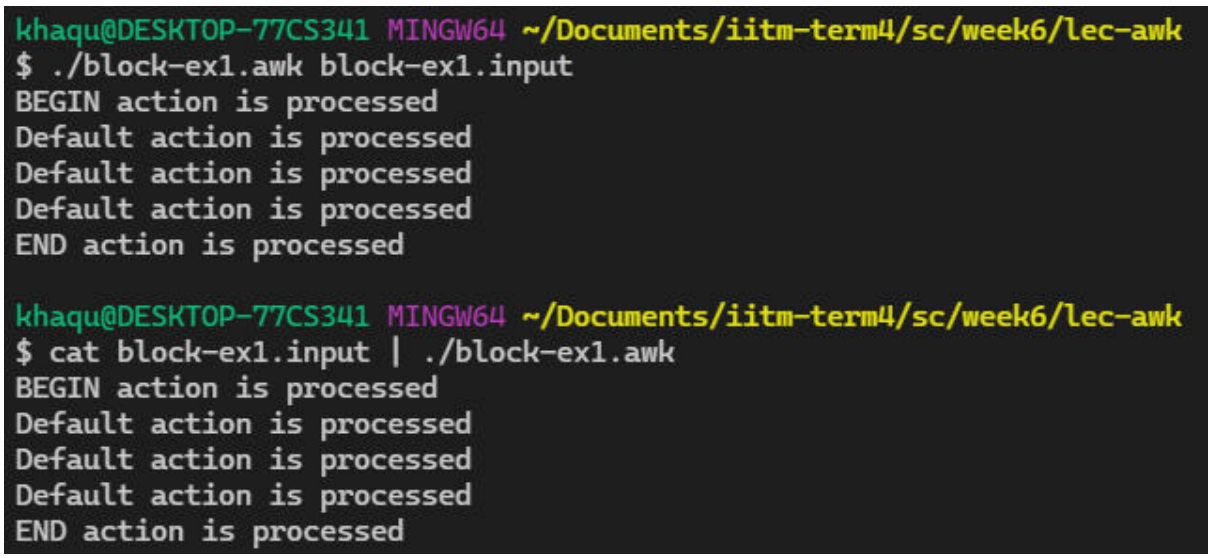
Block example

It does one print statement to illustrate how many times each codeblock is being processed

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
{
    print "Default action is processed";
}
END {
    print "END action is processed";
}
```

```
line-1 word1a word1b word1b
line-2 word2a word2b word2b
line-3 word3a word3b word3c
```

Output

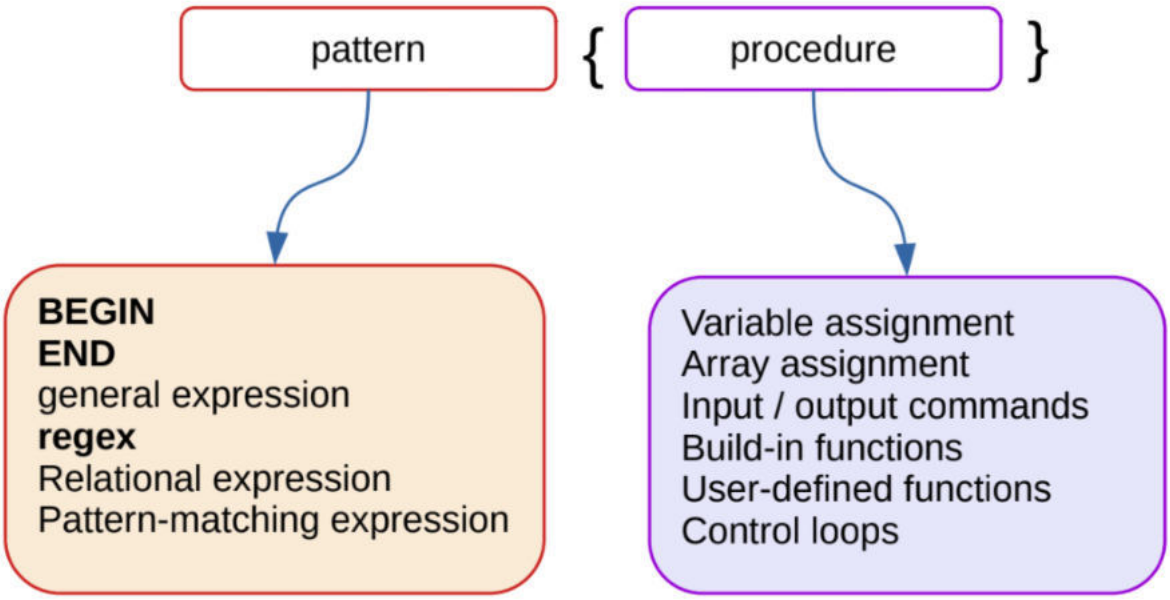


Built-in variables

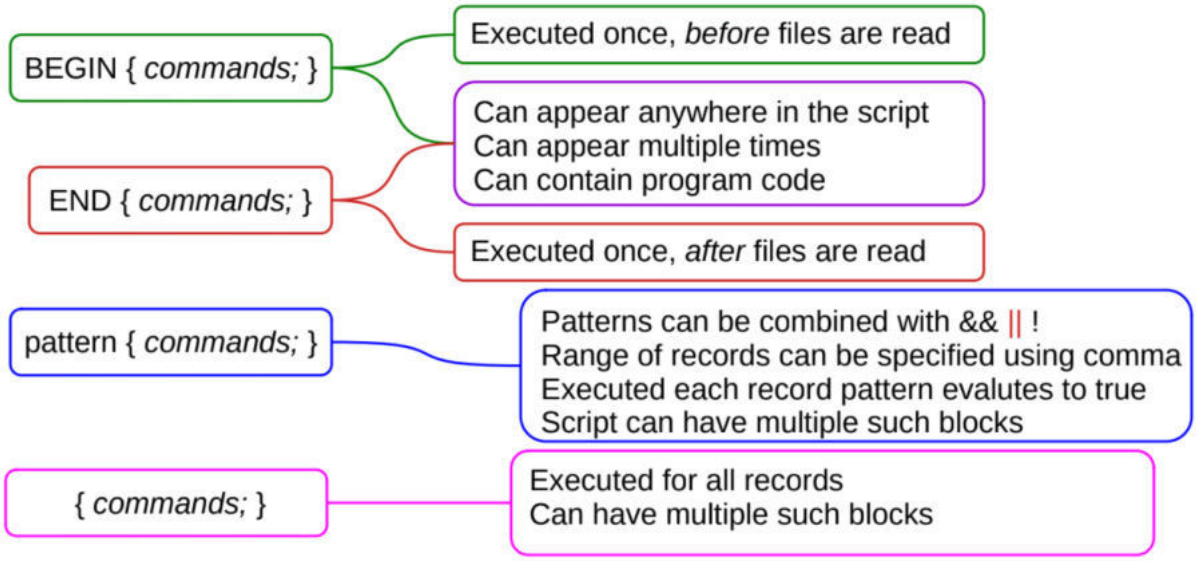
ARGC	Number of arguments supplied on the command line (except those that came with -f & -v options)
ARGV	Array of command line arguments supplied; indexed from 0 to ARGC-1
ENVIRON	Associative array of environment variables
FILENAME	Current filename being processed
FNR	Number of the current record, relative to the current file
FS	Field separator, can use regex
NF	Number of fields in the current record
NR	Number of the current record
OFMT	Output format for numbers

OFS	Output fields separator
ORS	Output record separator
RS	Record separator
RLENGTH	Length of string matched by match() function
RSTART	First position in the string matched by match() function
SUBSEP	Separator character for array subscripts
$0	Entire input record
$n	nth field in the current record

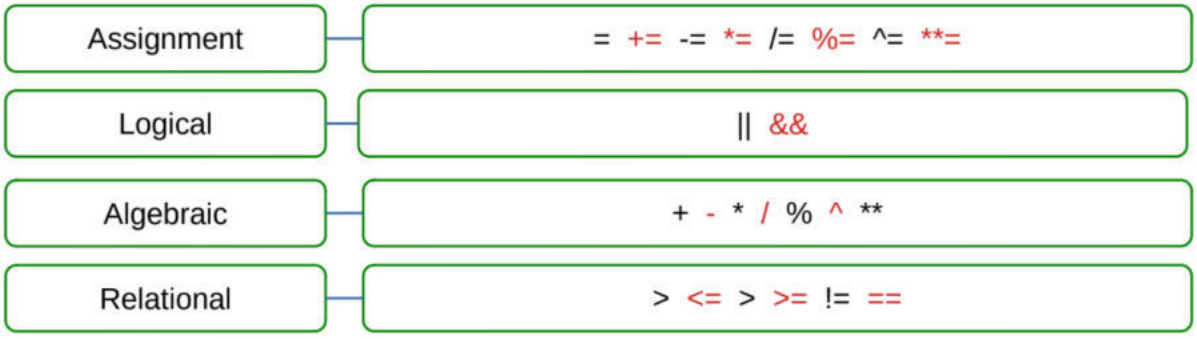
`awk` scripts



execution



operators



`expr ? a : b`	Conditional expression
`a in array`	Array membership
`a ~ /regex/`	Regular expression match
`a !~ /regex/`	Negation of regular expression match
`++`	Increment, both prefix and postfix
`--`	decrement, both prefix and postfix
`$`	Field reference
	Blank is for concatenation

Functions and Commands

Arithmetic	`atan2 cos exp int log rand sin sqrt srand`
String	`asort asorti gsub index length match split sprintf strtonum sub substr tolower toupper`
Control Flow	`break continue do while exit for if else return`
Input / Output	`close fflush getline next nextline print printf`
Programming	`extension delete function system`
bit-wise	`and compl lshift or rshift xor`

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
{
    print "record: " $0;
    print "processing record number: " FNR;
    print "number of fields in the current record: " NF;
```

```
}
END {
    print "END action is processed";
}
```

```
line-1 word1a word1b word1b
line-2 word2a word2b
line-3 word3a
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex2.awk block-ex2.input
BEGIN action is processed
record: line-1 word1a word1b word1b
processing record number: 1
number of fields in the current record: 4
record: line-2 word2a word2b
processing record number: 2
number of fields in the current record: 3
record: line-3 word3a
processing record number: 3
number of fields in the current record: 2
END action is processed
```

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
/[[:alpha:]]/ {
    print "alpha record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
/[[:alnum:]]/ {
    print "alnum record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
/[[:digit:]]/ {
    print "digit record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
END {
    print "END action is processed";
}
```

```
hello world welcome to awk
mm22b901 name place
600036 mm23b902 name
04422574770 231
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex3.awk block-ex3.input
BEGIN action is processed
alpha record: 1:hello world welcome to awk
number of fields in the current record: 5
alnum record: 1:hello world welcome to awk
number of fields in the current record: 5
alpha record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 2:mm22b901 name place
number of fields in the current record: 3
digit record: 2:mm22b901 name place
number of fields in the current record: 3
alpha record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 3:600036 mm23b902 name
number of fields in the current record: 3
digit record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 4:044222574770 231
number of fields in the current record: 2
digit record: 4:044222574770 231
number of fields in the current record: 2
END action is processed
```

To match only the first field, instead of the entire record

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
$1 ~ /[[:alpha:]]/ {
    print "alpha record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
$1 ~ /[[:alnum:]]/ {
    print "alnum record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
$1 ~ /[[:digit:]]/ {
    print "digit record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
END {
    print "END action is processed";
}
```

Input is the same as the previous

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex4.awk block-ex3.input
BEGIN action is processed
alpha record: 1:hello world welcome to awk
number of fields in the current record: 5
alnum record: 1:hello world welcome to awk
number of fields in the current record: 5
alpha record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 2:mm22b901 name place
number of fields in the current record: 3
digit record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 3:600036 mm23b902 name
number of fields in the current record: 3
digit record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 4:044222574770 231
number of fields in the current record: 2
digit record: 4:044222574770 231
number of fields in the current record: 2
END action is processed
```

In the following example, we don't do any pattern matching or comparison

Instead, we look at the number of fields in any particular (or arbitrary) record

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
    FS="[ .;:-]";
}
NF > 2 {
    print "acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
NF <= 2 {
    print "not acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
END {
    print "END action is processed";
}
```

Input

```
hello:world:welcome to awk
mm22b901;name;place
600036-mm23b902-name
04422574770 231 12345
gphani@iitm.ac.in
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex5.awk block-ex5.input
BEGIN action is processed
acceptable record:1:hello:world:welcome to awk
number of fields in the current record:5
acceptable record:2:mm22b901;name;place
number of fields in the current record:3
acceptable record:3:600036-mm23b902-name
number of fields in the current record:3
acceptable record:4:04422574770 231 12345
number of fields in the current record:3
acceptable record:5:gphani@iitm.ac.in
number of fields in the current record:3
END action is processed
```

We can modify the FS (field separator) in the above script to not allow `" "` (space) and `"."` (dot) as the field separator

The result will change

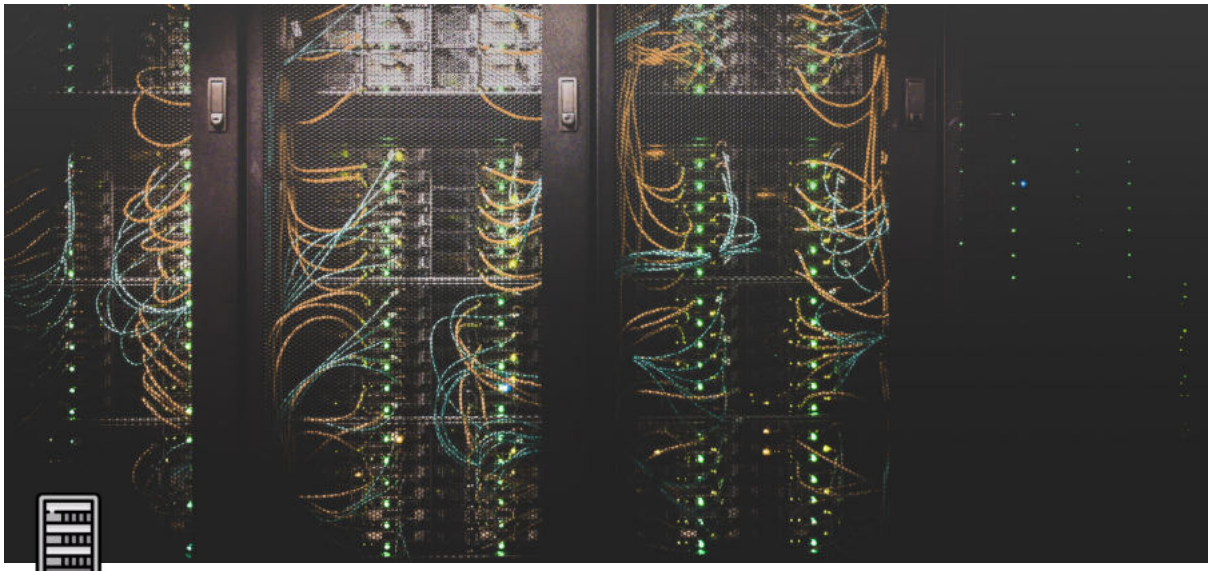
Modified code (Notice line #4)

```
BEGIN {
    print "BEGIN action is processed";
    FS="[;:-]";
}
NF > 2 {
    print "acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
NF <= 2 {
    print "not acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
END {
    print "END action is processed";
}
```

Input is the same as previous unmodified one

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex5.awk block-ex5.input
BEGIN action is processed
acceptable record:1:hello:world:welcome to awk
number of fields in the current record:3
acceptable record:2:mm22b901;name;place
number of fields in the current record:3
acceptable record:3:600036-mm23b902-name
number of fields in the current record:3
not acceptable record:4:044422574770 231 12345
number of fields in the current record:1
not acceptable record:5:gphani@iitm.ac.in
number of fields in the current record:1
END action is processed
```



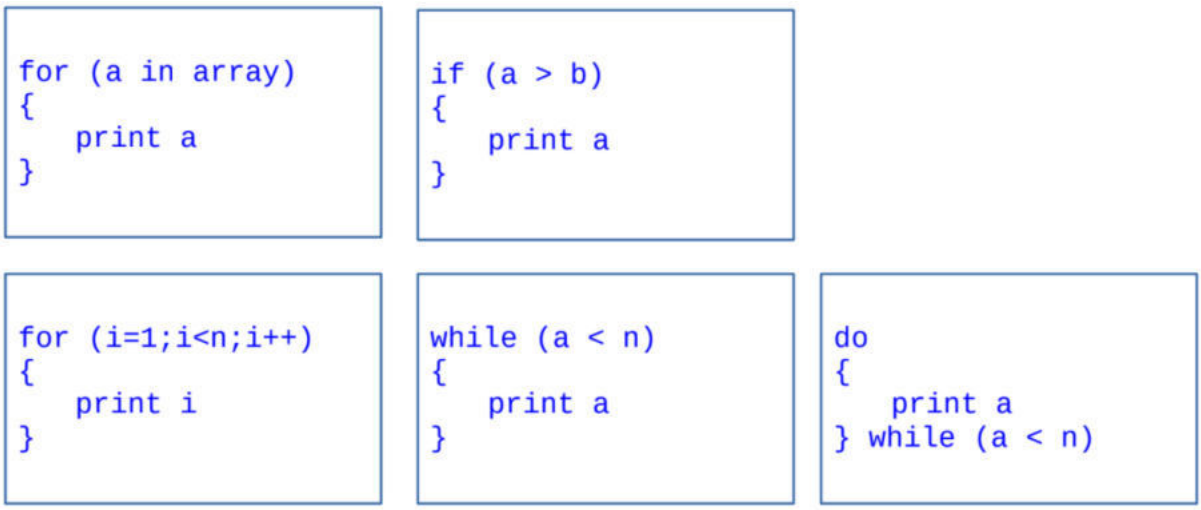
awk programming - pt. 2

Type	📒 Lecture
Date	@February 9, 2022
Lecture #	5
Lecture URL	https://youtu.be/g9_ij64u8eQ
Notion URL	https://21f1003586.notion.site/awk-programming-pt-2-b59debe0a4b54507a20eadf4bc0e1042
Week #	6

Arrays

- Associative arrays
- Sparse storage
- Index need not be integer
- `arr[index]=value`
- `for (var in arr)`
- `delete arr[index]`

Loops



`awk` program to calculate the total fee, avg fee and the roll numbers (and their fee) of those who paid less than a threshold from a garbled data file

`block-ex6.awk`

```
#!/usr/bin/gawk -f
BEGIN {
    print "Report of fee paid:";
    totfee=0;
    FS=" ";
}
{
    # print $0;
    if ($1 ~ /[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]][[:digit:]]{3}+/) {
        roll=$1;
        fee=$2;
        rf[roll]=fee;
        totfee += fee;
        print roll " paid " fee;
    }
}
END {
    cutoff=21500;
    print "List of students who paid less than " cutoff;
    ns=0;
    for (r in rf)
    {
        ns++;
        if(rf[r] < cutoff) print "check ", r, " paid only " rf[r];
    }
    avg = totfee/ns;
    print "Total fee collected:" totfee;
    print "Average fee collected:" avg;
}
```

Input

`block-ex6.input`

```
This file contains comments and numbers
Lines containing roll number and the fee paid need to be picked up
mm22b901  21000
Sum of the total fee paid needs to be printed at the end
Also a list of students, who paid how much
mm22b902 21000
Input text may contain some commends in between
mm22b906      21800
mm22b903 21500 paid through DD
mm22b905   22000
updated as on Jan 26, 2022 by Phani
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex6.awk block-ex6.input
Report of fee paid:
mm22b901 paid 21000
mm22b902 paid 21000
mm22b906 paid 21800
mm22b903 paid 21500
mm22b905 paid 22000
List of students who paid less than 21500
check  mm22b901  paid only 21000
check  mm22b902  paid only 21000
Total fee collected:107300
Average fee collected:21460
```

functions

```
cat infile |awk -f mylib -f myscript.awk
```

mylib

```
function myfunc1()
{
    printf "%s\n", $1
}

function myfunc2(a)
{
    return a*rand()
}
```

myscript.awk

```
BEGIN
{
    a=1
}
{
    myfunc1()
    b = myfunc2(a)
    print b
}
```

`func-lib.awk`

```
function myfunc1() {
    printf "%f\n", 2*$1;
}
function myfunc2(a) {
    # Adding zero to 'a' here just to store the value in 'b' as an integer
    b=a+0;
    return sin(b);
}
```

`func-example.awk`

```
BEGIN {
    FS=":";
    print "---------- BEGIN ----------";
    c=atan2(1,1);
    print "c=" c;
    print "----------------------------";
}
{
    print $0;
    myfunc1();
    a=$1;
    b=myfunc2(a);
    print "b=" b;
}
END {
    print "---------- ADD ----------";
    d=myfunc2(c);
    print "d=" d;
    print "-------------------------";
}
```

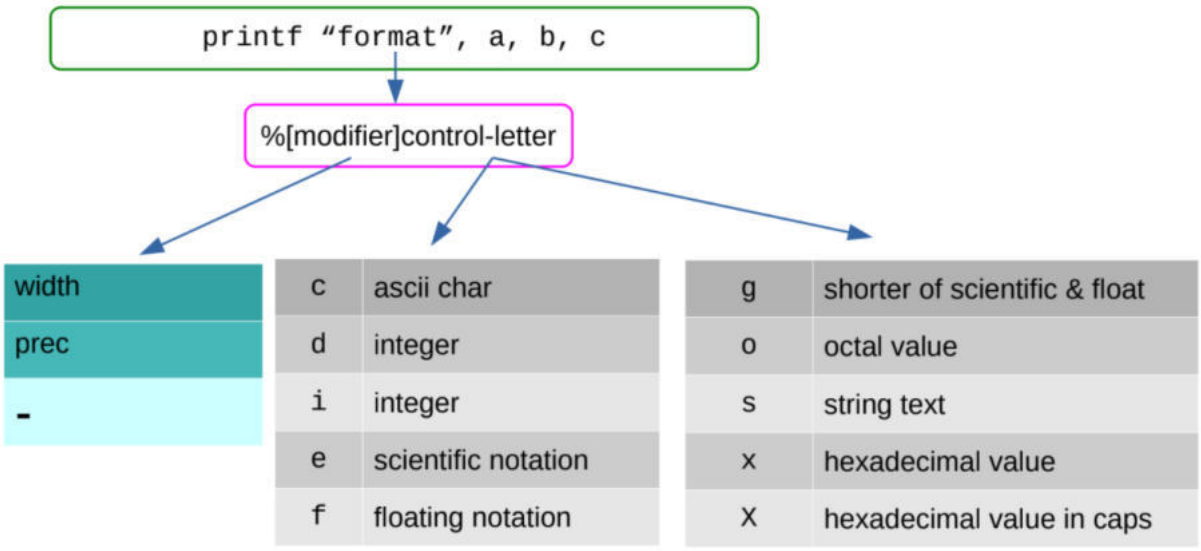
Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo "12.5" | gawk -f func-lib.awk -f func-example.awk
---------- BEGIN ----------
c=0.785398
-------------------------
12.5
25.000000
b=-0.0663219
---------- ADD ----------
d=0.707107
-----------------------
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo "" | gawk -f func-lib.awk -f func-example.awk
---------- BEGIN ----------
c=0.785398
-------------------------

0.000000
b=0
---------- ADD ----------
d=0.707107
-----------------------
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ touch xaa; cat xaa | gawk -f func-lib.awk -f func-example.awk
---------- BEGIN ----------
c=0.785398
-------------------------
---------- ADD ----------
d=0.707107
-----------------------
```

Pretty printing



`bash` + `awk`

- Including awk inside shell script
- heredoc feature
- Use with other shell scripts on command line using pipe

Examples

The following code creates 3 columns of 10 rows filled with random numbers

We also need to pass an input string for this code

But as we do not have a body for the action block, empty string suffices

`rsheet-create.awk`

```
#!/usr/bin/gawk -f
BEGIN {
    nl = 10;
    nc = 3;
    for(j=0; j<nl; j++) {
        for(i=0; i<nc; i++) {
            printf("%f ", rand());
        }
        printf("\n");
    }
}
{}
END {
    print nl " lines with " nc " columns of random numbers created";
}
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo " " | ./rsheet-create.awk
0.924046 0.593909 0.306394
0.578941 0.740133 0.786926
0.436370 0.332195 0.778880
0.100887 0.785084 0.835159
0.761209 0.496077 0.426298
0.945798 0.821802 0.709269
0.157828 0.119752 0.909685
0.868084 0.449256 0.705432
0.399686 0.645049 0.696163
0.300211 0.591664 0.956569
10 lines with 3 columns of random numbers created
```

Now, we can modify the above code to generate 2 columns of 2 million rows filled with random numbers

We will pass the empty file `xaa` we created earlier, and also use the command `time` to calculate the time taken to execute

```
#!/usr/bin/gawk -f
BEGIN {
    nl = 2000000;
    nc = 2;
    for(j=0; j<nl; j++) {
        for(i=0; i<nc; i++) {
            printf("%f ", rand());
        }
        printf("\n");
    }
}
{}
END {
    # print nl " lines with " nc " columns of random numbers created";
}
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time ./rsheet-create.awk xaa > rsheet-data.txt

real    0m17.544s
user    0m17.500s
sys     0m0.078s
```

Welp, it took a long time on my 🥔 laptop

Now, we should process these numbers and create 2 more rows after that which would contain the product and the sum of these numbers

`rsheet-process.awk`

```
#!/usr/bin/gawk -f
BEGIN {
    OFS=" ";
    FS=" ";
}
{
    prod = $1*$2;
    sum = $1+$2;
    printf("%f %f %f %f\n", $1, $2, prod, sum);
}
END {}
```

Input is the `rsheet-data.txt` file generated earlier

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time ./rsheet-process.awk rsheet-data.txt > rsheet-pdata.txt

real    0m53.066s
user    0m52.671s
sys     0m0.077s
```

An awk program to know the IP addresses of all the clients that connected to our web server (Apache) in the last 5 days

Server log file [76.6MB]: https://firebasestorage.googleapis.com/v0/b/fb-sandbox-25.appspot.com/o/access-full.log?alt=media&token=b1e22c89-5f85-4e12-8087-dfe50fa0e49d

To get the IP addresses

Type the following code on the bash terminal

```
awk 'BEGIN {FS=" "}{print $1}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{print $1}' access-head.log
103.47.219.249
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
```

```
awk 'BEGIN {FS=" "}{print $1}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{print $1}' access-tail.log
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
```

Here, `access-head.log` and `access-tail.log` are the top 10 and bottom 10 lines of the file `access-full.log` respectively

Similarly, to get the date

Type the following code on the bash terminal

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-head.log
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
```

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-tail.log
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
```

Now, combine the above 2 scripts (one from the IP address one, one from the date one) to get the IP address and the date on which they connected to the server

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-head.log
27/Jan/2022 103.47.219.249
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
```

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-tail.log
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
```

Now, run it on the entire log file and `time` it too. Also save it to a file named `date-ip.txt`

```
time awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-full.log > date-ip.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-full.log > date-ip.txt

real    0m0.688s
user    0m0.625s
sys     0m0.031s
```

Statistics time 😐

To calculate the number of connections per day from the log file

To get the datestring for the last n days

```
date --date="<n> days ago" +%d/%m/%Y
```

NOTE: Replace the `<n>` with the actual number

Example

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="5 days ago" +%d/%m/%Y
05/02/2022

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="2 days ago" +%d/%m/%Y
08/02/2022

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="3 days ago" +%d/%m/%Y
07/02/2022
```

Code

`apache-log-ex1.awk`

```
#!/usr/bin/gawk -f
BEGIN {
  ndays=15;
  dformat="+%d/%b/%Y";
  for(i=0; i<ndays; i++) {
    cmdstr=sprintf("date --date=\"%d days ago\" %s", i, dformat);
    cmdstr | getline mydate;
    dates[i]=mydate;
  }
  dstring = "";
  for (i in dates) {
    dstring = dstring " " dates[i];
  }
  print "date string=" dstring;
}
{
  ldate=substr($4,2,11);
  w = match(dstring,ldate);
  if(w != 0) {
    # print ldate " " $1 " " $7;
    print ldate " " $1;
    ipcount[$1]++;
  }
}
END {
  print "------------ IP STATS ------------";
  for (j in ipcount) {
    print j " "ipcount[j];
```

```
  }
}
```

Input

`access-head.log` or `access-tail.log` or `access-full.log`

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./apache-log-ex1.awk access-head.log
date string= 10/Feb/2022 09/Feb/2022 08/Feb/2022 07/Feb/2022 06/Feb/2022
8/Jan/2022 27/Jan/2022
27/Jan/2022 103.47.219.249
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
------------ IP STATS ------------
103.47.219.249 1
54.209.123.136 9
```

DNS lookup utility

Command: `dig`

Usage: `dig -x <ip-address>`

Example

```
kashif@zen:~$ dig -x 54.209.123.136

; <<>> DiG 9.16.1-Ubuntu <<>> -x 54.209.123.136
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46107
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;136.123.209.54.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
136.123.209.54.in-addr.arpa. 300 IN     PTR     ec2-54-209-123-136.compute-1.amazonaws.com.

;; Query time: 88 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Fri Feb 11 09:08:39 UTC 2022
;; MSG SIZE  rcvd: 112
```

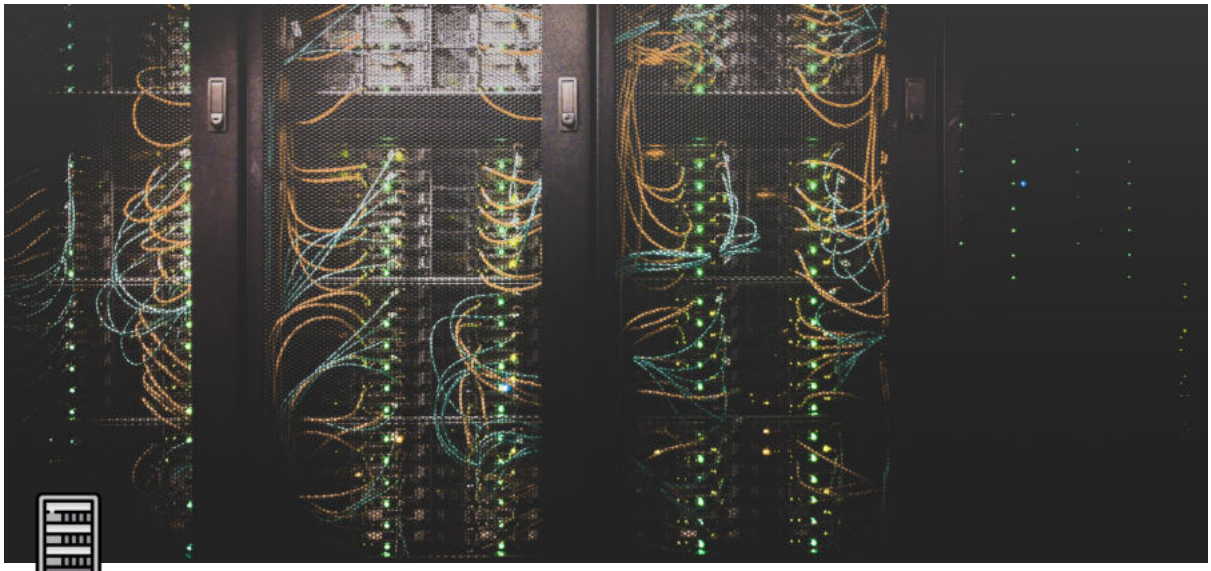
To get a slimmer output: `dig +noall +answer -x 34.234.167.93`

Example

```
kashif@zen:~$ dig +noall +answer -x 34.234.167.93
93.167.234.34.in-addr.arpa. 300 IN      PTR     ec2-34-234-167-93.compute-1.amazonaws.com.
```

Previous `awk` code, modified

```
#!/usr/bin/gawk -f
BEGIN {
  ndays=25;
  dformat="+%d/%b/%Y";
  for(i=0; i<ndays; i++) {
    cmdstr=sprintf("date --date=\"%d days ago\" %s", i, dformat);
    cmdstr | getline mydate;
    dates[i]=mydate;
  }
  dstring = "";
  for (i in dates) {
    dstring = dstring " " dates[i];
  }
  print "date string=" dstring;
}
{
  ldate=substr($4,2,11);
  w = match(dstring,ldate);
  if(w != 0) {
    # print ldate " " $1 " " $7;
    # print ldate " " $1;
    ipcount[$1]++;
  }
}
END {
  print "------------ IP STATS ------------";
  for (j in ipcount) {
    print j " "ipcount[j];
        cmdstr = sprintf("dig +noall +answer -x %s", j);
        cmdstr | getline ipinfo;
        print ipinfo;
  }
}
```



Stream editor sed

Type	📒 Lecture
Date	@February 20, 2022
Lecture #	1
Lecture URL	https://youtu.be/uCJjZ3cbaAw
Notion URL	https://21f1003586.notion.site/Stream-editor-sed-4d1c90712d98497a9bd56a4dd192e134
Week #	7

`sed`

It is a language for processing text streams

Introduction

- It is a programming language
- `sed` is an abbreviation of **s**tream **ed**itor
- It is a part of POSIX
- `sed` precedes `awk`

Execution Model

- Input stream is a set of lines
- Each line is a sequence of characters
- Two data buffers are maintained → active ***pattern*** space and auxiliary ***hold*** space
- For each line of input, an execution cycle is performed loading the line into the pattern space
- During each cycle, all the statements in the script are executed in the sequence for matching address pattern for actions specified with the options provided

Usage

- Single line at the command line

  `sed -e 's/hello/world/g' input.txt`

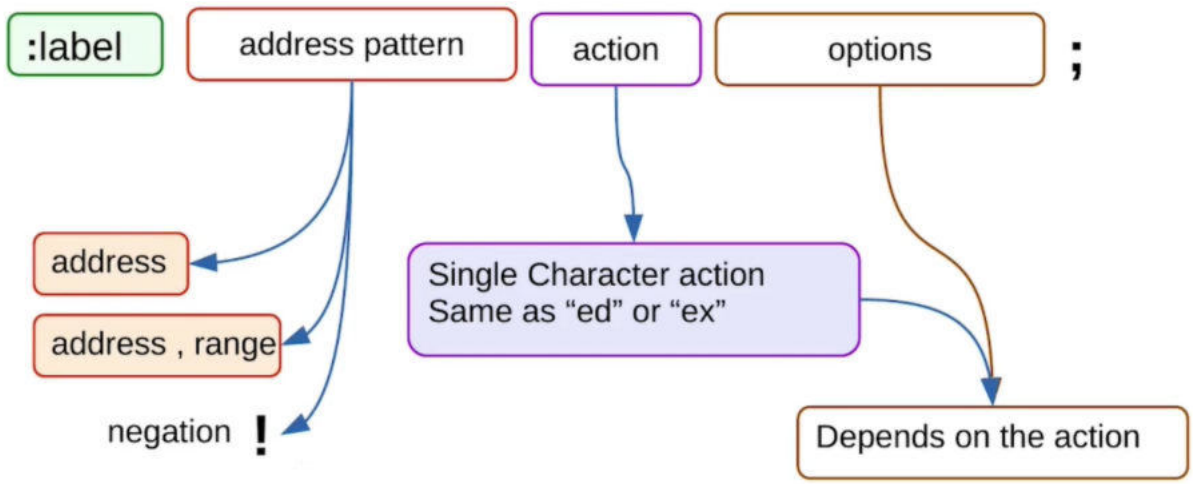
- Script interpreted by sed

  `sed -f ./myscript.sed input.txt`

  Where `myscript.sed` could be ...

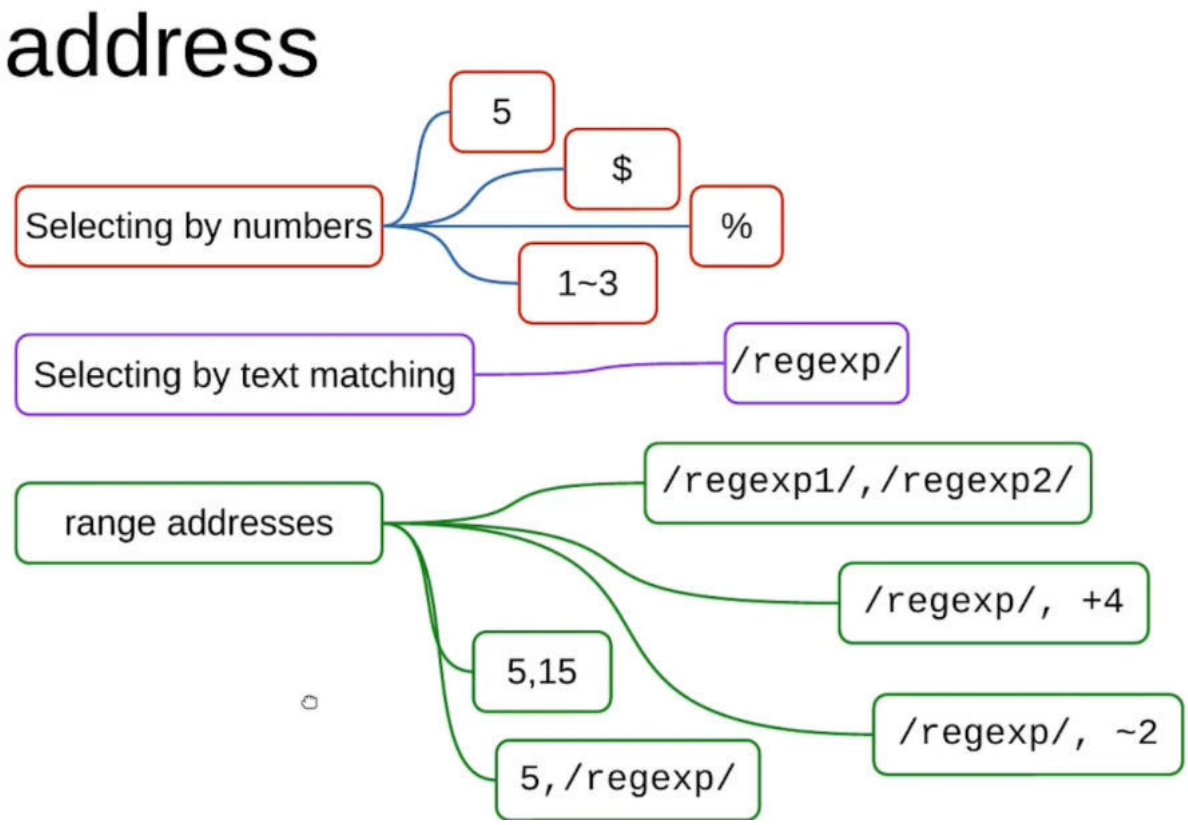
```
#!/usr/bin/sed -f
2,8s/hello/world/g
```

`sed` statements



Grouping commands

`{ cmd; cmd; }`



- `5` → select the 5th line
- `$` → select the last line
- `%` → select all the lines
- `1~3` → starting from the 1st line, select every 3rd line
- `/regex/` → select the line by matching the regular expression
- `/regex1/, /regex2/` → from the line of the 1st occurrence of `regex1` to the line of the 1st occurrence of `regex2`
- `/regex/, +4` → action has to be taken on the next 4 lines, starting from the first match of `regex`
- `/regex/, ~2` → the line where the 1st regular expression is matched, the next line which is a multiple of 2 will be matched
- `5, 15` → execute from the 5th line to 15th line
- `5, /regex/` → from the 5th line till the first occurrence of `regex`

actions

p	Print the pattern space
d	Delete the pattern space
s	Substitute using regex match `s/pattern/replacement/g`
=	Print current input line number, \n
#	comment
i	Insert above current line
a	Append below current line
c	Change current line

programming

b *label*	Branch unconditionally to *label*
:label	Specify location of *label* for branch command
N	Add a new line to the pattern space and append next line of input into it.
q	Exit sed without processing any more commands or input lines
t *label*	Branch to *label* only if there was a successful substitution was made
T *label*	Branch to *label* only if there was **no** successful substitution was made
w *filename*	Write pattern space to *filename*
x	Exchange the contents of hold and pattern spaces

`bash` + `sed`

- Including sed inside shell script
- heredoc feature

- Use with other shell scripts on the command line using pipe

---

Examples

`sample.txt`

```
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Default action is to print

`sed -e “” sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e "" sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

The above command takes an optional `-n` flag to negate whatever command we give to `sed`

Suppress automatic printing of pattern space

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e "" sample.txt
```

Print every line with it’s line number

`sed -e “=” sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e "=" sample.txt
1
L1 Linux is a free, opensource, secure and reliable os
2
L2 text editing: vim or emacs in place of notepad
3
L3 text processing: latex, libreoffice write in place of microsoft word
4
L4 spreadsheet: libreoffice calc in place of microsoft excel
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
9
L9 symbolic math: sympy, sagemath in place of maple, mathematica
10
L10 array math: octave, numpy scilab in place of matlab
11
L11 literature: jabref, calibre in place of mendeley
12
L12 video editing: kdenlive in place of camtasia
13
L13 screen recording: obs in place of camtasia
14
L14 fetch files: wget, curl
15
L15 security: ufw, netstat, nmap
```

Print the 5th line, and don’t print anything else by default

`sed -n -e “5p” sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e "5p" sample.txt
L5 slides: libreoffice impress in place of microsoft powerpoint
```

If we use `sed -e “5p” sample.txt`, line #5 will be printed twice

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e "5p" sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

We can also replace the double quotes with single quotes

We also use single quotes when we don’t want the shell to interpret certain special characters

---

Print every line except line #5

`sed -n -e ‘5!p’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '5!p' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

To print the last line

`sed -n -e ‘$p’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '$p' sample.txt
L15 security: ufw, netstat, nmap
```

To print all lines except the last line

```
sed -n -e '$!p' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '$!p' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
```

Print line #5 to line #8

```
sed -n -e '5,8p' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '5,8p' sample.txt
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
```

Now, combine two commands

```
sed -n -e '=; 5,8p' sample.txt
```

Print the line numbers, and print the lines from 5 to 8

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '=; 5,8p' sample.txt
1
2
3
4
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
9
10
11
12
13
14
15
```

If we don’t want those blank line numbers to show up, we write the following command

`sed -n -e ‘5,8{=;p}’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '5,8{=;p}' sample.txt
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
```

Print every odd line (or starting from line #1, print every 2nd line)

`sed -n -e ‘1~2p’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '1~2p' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L3 text processing: latex, libreoffice write in place of microsoft word
L5 slides: libreoffice impress in place of microsoft powerpoint
L7 browsing: mozilla firefox in place of microsoft edge
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L11 literature: jabref, calibre in place of mendeley
L13 screen recording: obs in place of camtasia
L15 security: ufw, netstat, nmap
```

Similarly, print every 3rd line starting from line #1

`sed -n -e '1~3p' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '1~3p' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L4 spreadsheet: libreoffice calc in place of microsoft excel
L7 browsing: mozilla firefox in place of microsoft edge
L10 array math: octave, numpy scilab in place of matlab
L13 screen recording: obs in place of camtasia
```

Negate the above command

`sed -n -e '1~3!p' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '1~3!p' sample.txt
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Print the lines that contain the following regex pattern (here, in this case, it is a phrase)

`sed -n -e '/microsoft/p' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '/microsoft/p' sample.txt
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '/in place of/p' sample.txt
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '/in place of/!p' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Print n number of lines after the matching regex pattern

`sed -n -e '/adobe/,+2p' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -n -e '/adobe/,+2p' sample.txt
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
```

Delete a line (line #5 in this case) and print the remaining lines

`sed -e '5d' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '5d' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Delete a range of lines (and print the remaining lines, *duh*)

`sed -e ‘5,8d’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '5,8d' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Delete the first line

`sed -e ‘1d’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1d' sample.txt
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Delete the last line

`sed -e '$d' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '$d' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
```

Delete the lines with the following regex pattern

`sed -e '/microsoft/d' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '/microsoft/d' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Search and replace the first regex with the second regex pattern on all lines

`sed -e ‘s/microsoft/MICROSOFT/g’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e 's/microsoft/MICROSOFT/g' sample.txt
L1 Linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of MICROSOFT word
L4 spreadsheet: libreoffice calc in place of MICROSOFT excel
L5 slides: libreoffice impress in place of MICROSOFT powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of MICROSOFT edge
L8 email: mozilla thunderbird in place of MICROSOFT outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Search and replace the first regex with the second one on line #1

`sed -e ‘1s/linux/LINUX/g’ sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1s/linux/LINUX/g' sample.txt
L1 LINUX is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

```
sed -e '1,$s/in place of/in lieu of/g' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1,$s/in place of/in lieu of/g' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in lieu of notepad
L3 text processing: latex, libreoffice write in lieu of microsoft word
L4 spreadsheet: libreoffice calc in lieu of microsoft excel
L5 slides: libreoffice impress in lieu of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in lieu of adobe photoshop
L7 browsing: mozilla firefox in lieu of microsoft edge
L8 email: mozilla thunderbird in lieu of microsoft outlook
L9 symbolic math: sympy, sagemath in lieu of maple, mathematica
L10 array math: octave, numpy scilab in lieu of matlab
L11 literature: jabref, calibre in lieu of mendeley
L12 video editing: kdenlive in lieu of camtasia
L13 screen recording: obs in lieu of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Print the lines without the first few characters at the beginning of each line

(Here, we are removing it from line #3 to line #6)

```
sed -E -e '3,6s/^L[[:digit:]]+ //g' sample.txt
```

Takes in an optional flag `-E` to let `sed` know that we're using the extended regex engine

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e '3,6s/^L[[:digit:]]+ //g' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

To apply the above command to all the lines

`sed -E -e 's/^L[[:digit:]]+ //g' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e 's/^L[[:digit:]]+ //g' sample.txt
linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkscape, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy scilab in place of matlab
literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
screen recording: obs in place of camtasia
fetch files: wget, curl
security: ufw, netstat, nmap
```

Remove line numbers from line #3 to the line till it encounters a regex pattern

`sed -E -e '3,/symbolic/s/^L[[:digit:]]+ //g' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e '3,/symbolic/s/^L[[:digit:]]+ //g' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkscape, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Remove the line numbers starting from line #1 and do it every 3rd line

```
sed -E -e '1~3s/^L[[:digit:]]+ //g' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e '1~3s/^L[[:digit:]]+ //g' sample.txt
linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Similarly, do the opposite of remove line numbers starting from line #1 and doing it every 2nd line

```
sed -E -e '1~2!s/^L[[:digit:]]+ //g' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e '1~2!s/^L[[:digit:]]+ //g' sample.txt
L1 linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Remove the line numbers starting from a line having a regex pattern to a line having another regex pattern

`sed -E -e '/text/,/video/s/^L[[:digit:]]+ //g' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -e '/text/,/video/s/^L[[:digit:]]+ //g' sample.txt
L1 linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
text processing: latex, libreoffice write in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkscape, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy scilab in place of matlab
literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Insert a line at line #1

Syntax within single quotes: `<position>i <text-to-insert>`

`sed -e '1i ---------- header ----------' sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1i ---------- header ----------' sample.txt
---------- header ----------
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Similarly, to insert header and append footer in a single line ...

Syntax to append a footer: `$a <text-to-append>`

```
sed -e '1i ---------- header ----------' -e '$a ---------- footer ----------' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1i ---------- header ----------' -e '$a ---------- footer ----------' sample.txt
---------- header ----------
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
---------- footer ----------
```

So, it seems like that `i` inserts before the mentioned line and `a` appends after the line

Insert a line before line #4 and append a line after line #8 in the same syntax

```
sed -e '4i -------- header --------' -e '8a -------- footer --------' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '4i -------- header --------' -e '8a -------- footer --------' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
-------- header --------
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
-------- footer --------
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

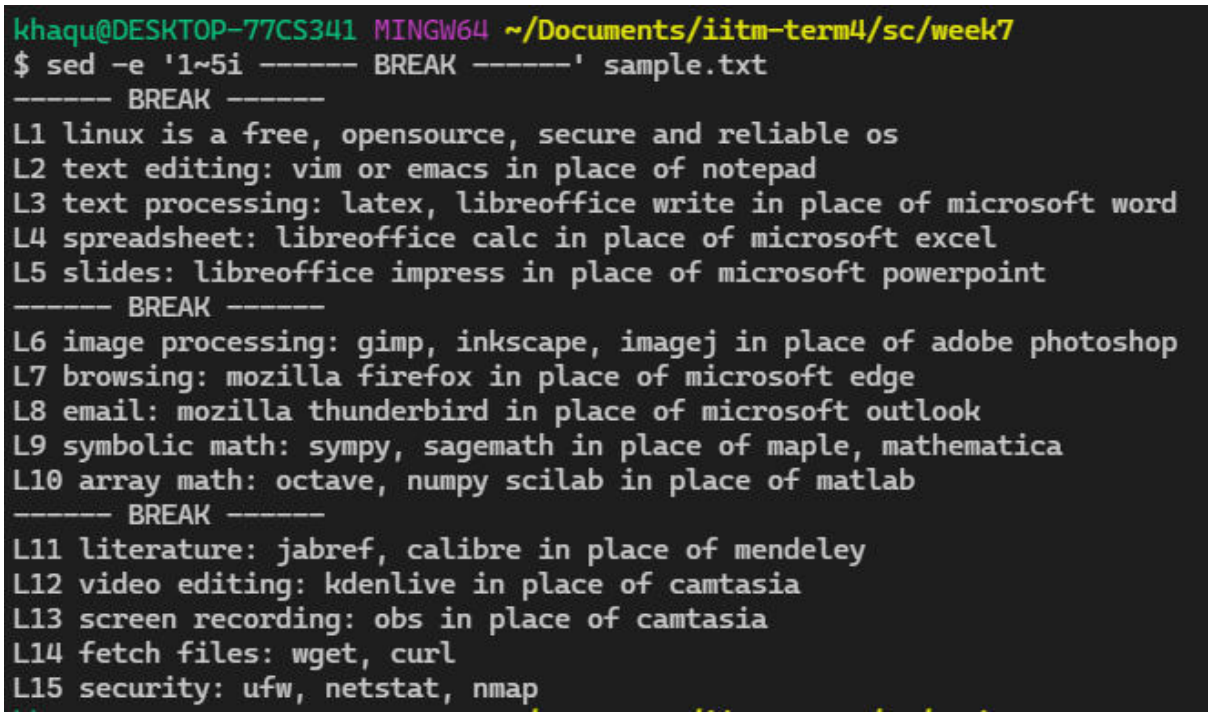
Similarly, we can do this insert and append using regex patterns instead of line numbers

```
sed -e '/microsoft/i ---- WATCHOUT ----' -e '/in place of/a ---- ALTERNATIVE ----' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '/microsoft/i ---- WATCHOUT ----' -e '/in place of/a ---- ALTERNATIVE ----' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
---- ALTERNATIVE ----
---- WATCHOUT ----
L3 text processing: latex, libreoffice write in place of microsoft word
---- ALTERNATIVE ----
---- WATCHOUT ----
L4 spreadsheet: libreoffice calc in place of microsoft excel
---- ALTERNATIVE ----
---- WATCHOUT ----
L5 slides: libreoffice impress in place of microsoft powerpoint
---- ALTERNATIVE ----
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
---- ALTERNATIVE ----
---- WATCHOUT ----
L7 browsing: mozilla firefox in place of microsoft edge
---- ALTERNATIVE ----
---- WATCHOUT ----
L8 email: mozilla thunderbird in place of microsoft outlook
---- ALTERNATIVE ----
L9 symbolic math: sympy, sagemath in place of maple, mathematica
---- ALTERNATIVE ----
L10 array math: octave, numpy scilab in place of matlab
---- ALTERNATIVE ----
L11 literature: jabref, calibre in place of mendeley
---- ALTERNATIVE ----
L12 video editing: kdenlive in place of camtasia
---- ALTERNATIVE ----
L13 screen recording: obs in place of camtasia
---- ALTERNATIVE ----
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

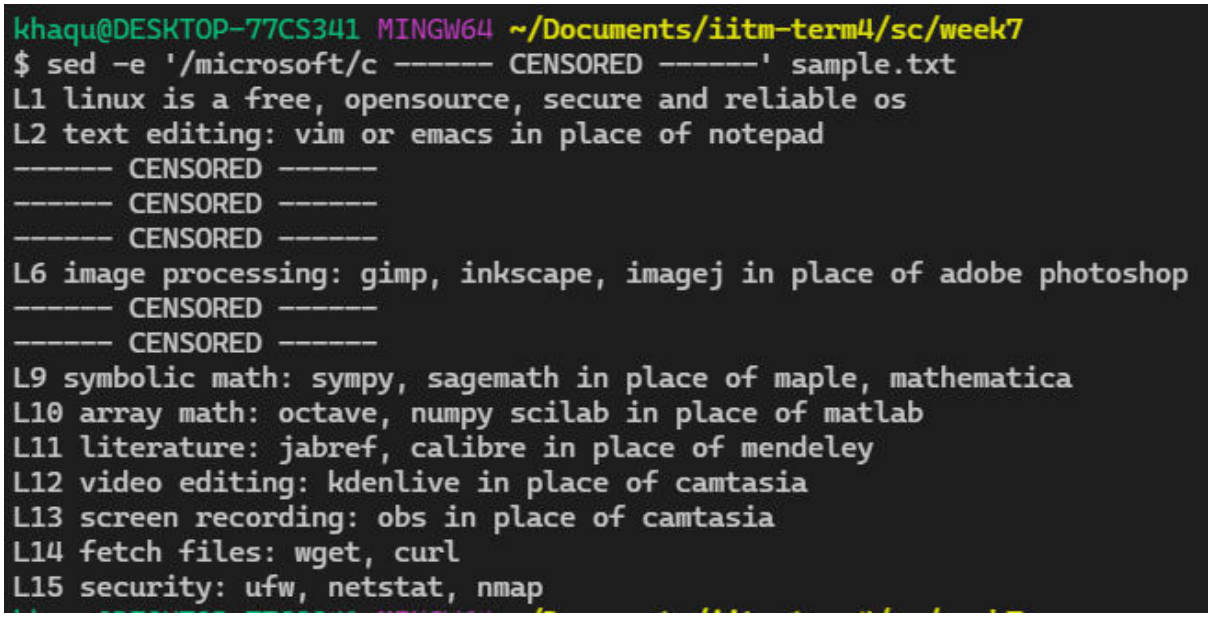
Insert a line of text starting at line #1 and every 5 lines

```
sed -e '1~5i ------ BREAK ------' sample.txt
```



Replace any line with a matching regex pattern with another line

```
sed -e '/microsoft/c ------ CENSORED ------' sample.txt
```



Replace line #1 and every 3rd line after that with another line

```
sed -e '1~3c ------ CENSORED ------' sample.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -e '1~3c ------ CENSORED ------' sample.txt
------ CENSORED ------
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice write in place of microsoft word
------ CENSORED ------
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place of adobe photoshop
------ CENSORED ------
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
------ CENSORED ------
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
------ CENSORED ------
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

`sed` commands from a file

`hf.sed`

```
#!/usr/bin/sed -f
1i ------ HEADER ------
$a ------ FOOTER ------
1,5s/in place of/in lieu of/g
6i --- SIMPLER STUFF FROM HERE ONWARDS ---
6,$s/in place of.*//g
```

To run the above sed file

`sed -f hf.sed sample.txt`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -f hf.sed sample.txt
------ HEADER ------
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in lieu of notepad
L3 text processing: latex, libreoffice write in lieu of microsoft word
L4 spreadsheet: libreoffice calc in lieu of microsoft excel
L5 slides: libreoffice impress in lieu of microsoft powerpoint
--- SIMPLER STUFF FROM HERE ONWARDS ---
L6 image processing: gimp, inkscape, imagej
L7 browsing: mozilla firefox
L8 email: mozilla thunderbird
L9 symbolic math: sympy, sagemath
L10 array math: octave, numpy scilab
L11 literature: jabref, calibre
L12 video editing: kdenlive
L13 screen recording: obs
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
------ FOOTER ------
```

To clean up the input file we used previously in `awk`

`block-ex6.input`

```
This file contains comments and numbers
Lines containing roll number and the fee paid need to be picked up
mm22b901  21000
Sum of the total fee paid needs to be printed at the end
Also a list of students, who paid how much
mm22b902 21000
Input text may contain some commends in between
mm22b906      21800
mm22b903 21500 paid through DD
mm22b905   22000
updated as on Jan 26, 2022 by Phani
```

`sed` script

`clean.sed`

```
/[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]][[:digit:]]+/!d
s/[ ]+/ /g
s/ ([[:digit:]]+).*/ \1/g
```

After we run the `sed` script on `block-ex6.input`

`sed -E -f clean.sed block-ex6.input`

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -E -f clean.sed block-ex6.input
mm22b901 21000
mm22b902 21000
mm22b906 21800
mm22b903 21500
mm22b905 22000
```

To join lines using `sed`

Joining lines requires `sed` to read one more line into the buffer

And this is achieved in a limited number of commands

`sed` commands can be constructed as loops and can perform complicated actions

Input file: `sample-split.txt`

```
L1 linux is a \ free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer \
in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place\
of microsoft \
powerpoint
L6 image processing: gimp, inkscape, imagej in place of\
adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple,\
mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

Join the lines that are ending in a backslash

`join.sed`

```
#!/usr/bin/sed -f
:x /\\$/N
/\\/s/\\\n//g
/\\$/bx
```

This is a loop (*ayo wut*)

Every line of the above `sed` file will be executed for every single that is read into the buffer from the input file

colon `:` indicates a label, and it is ignored while execution, it is used for branching

First line → Whenever there is a `\` followed by the end of the line, it will read the next line in the buffer, it's basically like joining the line following the one whic is being processed

Second line → On the lines which have `\` if there's a `\n` character after the `\` it will be replaced with null, basically to remove the `\` character

Third line → Those lines which contain `\` at the end of the line, we branch to the first line and repeat the steps again

When do we not branch and go on to the next cycle? It is when the line doesn't contain `\` at the end of the line, which means that a recursive action is performed

This recursive loop can go infinite if not designed properly

To run the above `sed` file on `sample-split.txt`

```
sed -f join.sed sample-split.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week7
$ sed -f join.sed sample-split.txt
L1 linux is a \ free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in placeof microsoft powerpoint
L6 image processing: gimp, inkscape, imagej in place ofadobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple,mathematica
L10 array math: octave, numpy scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
```

There is an optional `--debug` flag that can be passed with the `sed` command

and it will display a plethora of information

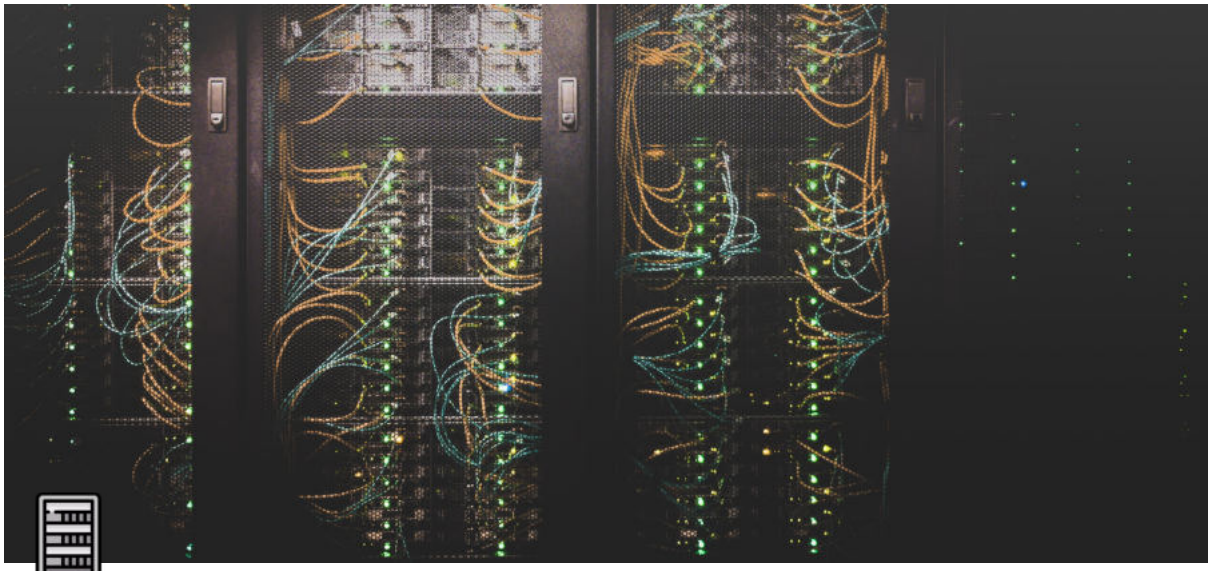
Command: `sed --debug -f join.sed sample-split.txt`

Output:

```
SED PROGRAM:
  :x
  /\\\$/ N
  /\\\\/ s/\\\\\n//g
  /\\\$/ b x
INPUT:   'sample-split.txt' line 1
PATTERN: L1 linux is a \\ free, opensource, secure and reliable os
COMMAND: :x
COMMAND: /\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
PATTERN: L1 linux is a \\ free, opensource, secure and reliable os
COMMAND: /\\\$/ b x
END-OF-CYCLE:
L1 linux is a \ free, opensource, secure and reliable os
INPUT:   'sample-split.txt' line 2
PATTERN: L2 text editing: vim or emacs in place of notepad
COMMAND: :x
COMMAND: /\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\$/ b x
END-OF-CYCLE:
L2 text editing: vim or emacs in place of notepad
INPUT:   'sample-split.txt' line 3
PATTERN: L3 text processing: latex, libreoffice writer \\
COMMAND: :x
COMMAND: /\\\$/ N
PATTERN: L3 text processing: latex, libreoffice writer \\\nin place of microsoft word
COMMAND: /\\\\/ s/\\\\\n//g
MATCHED REGEX REGISTERS
  regex[0] = 46-48 '\
'
PATTERN: L3 text processing: latex, libreoffice writer in place of microsoft word
COMMAND: /\\\$/ b x
END-OF-CYCLE:
L3 text processing: latex, libreoffice writer in place of microsoft word
INPUT:   'sample-split.txt' line 5
PATTERN: L4 spreadsheet: libreoffice calc in place of microsoft excel
COMMAND: :x
COMMAND: /\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\$/ b x
END-OF-CYCLE:
L4 spreadsheet: libreoffice calc in place of microsoft excel
INPUT:   'sample-split.txt' line 6
PATTERN: L5 slides: libreoffice impress in place\\
COMMAND: :x
COMMAND: /\\\$/ N
PATTERN: L5 slides: libreoffice impress in place\\\nof microsoft \\
COMMAND: /\\\\/ s/\\\\\n//g
MATCHED REGEX REGISTERS
  regex[0] = 39-41 '\
'
PATTERN: L5 slides: libreoffice impress in placeof microsoft \\
COMMAND: /\\\$/ b x
COMMAND: :x
COMMAND: /\\\$/ N
```

```
PATTERN: L5 slides: libreoffice impress in placeof microsoft \\\npowerpoint
COMMAND: /\\\\/ s/\\\\\n//g
MATCHED REGEX REGISTERS
  regex[0] = 52-54 '\
'
PATTERN: L5 slides: libreoffice impress in placeof microsoft powerpoint
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L5 slides: libreoffice impress in placeof microsoft powerpoint
INPUT:   'sample-split.txt' line 9
PATTERN: L6 image processing: gimp, inkscape, imagej in place of\\
COMMAND: :x
COMMAND: /\\\\$/ N
PATTERN: L6 image processing: gimp, inkscape, imagej in place of\\\nadobe photoshop
COMMAND: /\\\\/ s/\\\\\n//g
MATCHED REGEX REGISTERS
  regex[0] = 55-57 '\
'
PATTERN: L6 image processing: gimp, inkscape, imagej in place ofadobe photoshop
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L6 image processing: gimp, inkscape, imagej in place ofadobe photoshop
INPUT:   'sample-split.txt' line 11
PATTERN: L7 browsing: mozilla firefox in place of microsoft edge
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L7 browsing: mozilla firefox in place of microsoft edge
INPUT:   'sample-split.txt' line 12
PATTERN: L8 email: mozilla thunderbird in place of microsoft outlook
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L8 email: mozilla thunderbird in place of microsoft outlook
INPUT:   'sample-split.txt' line 13
PATTERN: L9 symbolic math: sympy, sagemath in place of maple,\\
COMMAND: :x
COMMAND: /\\\\$/ N
PATTERN: L9 symbolic math: sympy, sagemath in place of maple,\\\nmathematica
COMMAND: /\\\\/ s/\\\\\n//g
MATCHED REGEX REGISTERS
  regex[0] = 52-54 '\
'
PATTERN: L9 symbolic math: sympy, sagemath in place of maple,mathematica
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L9 symbolic math: sympy, sagemath in place of maple,mathematica
INPUT:   'sample-split.txt' line 15
PATTERN: L10 array math: octave, numpy scilab in place of matlab
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
```

```
L10 array math: octave, numpy scilab in place of matlab
INPUT:   'sample-split.txt' line 16
PATTERN: L11 literature: jabref, calibre in place of mendeley
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L11 literature: jabref, calibre in place of mendeley
INPUT:   'sample-split.txt' line 17
PATTERN: L12 video editing: kdenlive in place of camtasia
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L12 video editing: kdenlive in place of camtasia
INPUT:   'sample-split.txt' line 18
PATTERN: L13 screen recording: obs in place of camtasia
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L13 screen recording: obs in place of camtasia
INPUT:   'sample-split.txt' line 19
PATTERN: L14 fetch files: wget, curl
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L14 fetch files: wget, curl
INPUT:   'sample-split.txt' line 20
PATTERN: L15 security: ufw, netstat, nmap
COMMAND: :x
COMMAND: /\\\\$/ N
COMMAND: /\\\\/ s/\\\\\n//g
COMMAND: /\\\\$/ b x
END-OF-CYCLE:
L15 security: ufw, netstat, nmap
```



Version Control

Type	📒 Lecture
Date	@February 21, 2022
Lecture #	2
Lecture URL	https://youtu.be/tzYx6DKtvGw
Notion URL	https://21f1003586.notion.site/Version-Control-bff372d37e544b94a72544dafd13d3c7
Week #	7

Version Control

A group of programmers are working on a major project → tons of code in various files (modular approach)

Each programmer has multiple versions of each file they worked on

So, the basic idea of version control → trace back to a working version of code

Some version control systems are ...

SVN → Centrally hosted & managed version system

It has one master that keeps track of what version of the code that we are officially supporting

Then the master is going to ensure that the same version is replicated by others

Git → Distributed version control system

RAID

Redundant Array of Inexpensive/Independent Disks

It is a data storaga virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement or both

Since it is an array, it not only provides redundacy, but also speeds up the data access speed

As the speed limit of SATA interface is 6Gb/s, and we have 3 disks in an RAID array

When we access a file that is stored in 3 parts in those 3 disks, we can effectively utilize upto 18Gb/s of bandwidth

Workstations usually come with a RAID controller (it is a PCI card) and it determines how fast we can access data and the capacity of disks too

Hot Spare → It is a disk that we keep plugged in but it is not in use, but if any of the disks were to fail, the hot spare is automatically put to use

In a RAID config, **usable space < actual space**

Suppose, we have 4 x 8TB disks, so the actual space if 32TB, but the usable space might be 26TB

Git

- Distributed
- If the “master” server is hacked
  - Not much of an issue as every user (collaborator) has a copy of everything
- It doesn’t require a server, but we can have a server
- **remote** → It is a server with which we sync our code, usually considered as the master version
- **Protocol used** → `git`
- Options to run git →

- locally run git server
- campus git server
- gitlab
- github

Git Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf