



WeCloudData

# Linux and Shell Commands

Data Engineering Diploma



# **Linux** **Introduction**

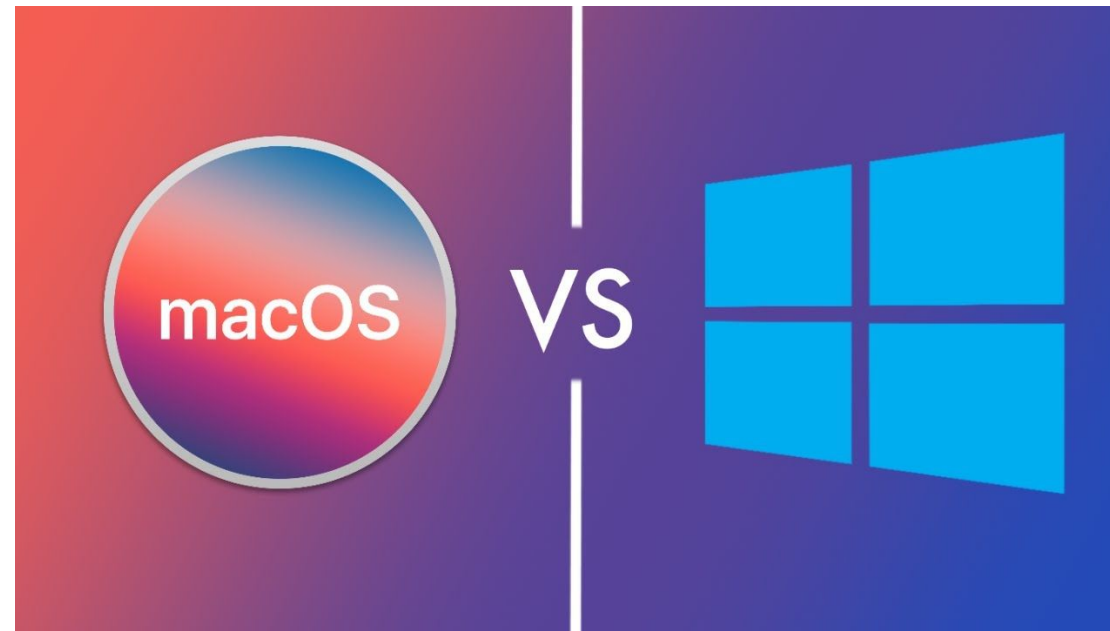
- **Linux Introduction**
- Linux Distros and Shell Architecture
- Operating System
- Files Permissions
- Environment Variables
- Commands
- Special Characters
- Shell Scripts

**Agenda.**



# Class Poll: Which OS are you using?

Linux Introduction





# Powered by Linux...

Linux Introduction





# Powered by Linux...

Linux Introduction



OpenAI



ChatGPT



# Powered by Linux...

Linux Introduction





# Linux Kernel and Shell

Linux Introduction

**The world is pretty much  
running on linux**






# Why data engineers need to learn linux?

## Linux Introduction

- Cloud
- Automation
- Databases
- Docker
- Big Data
- etc.



# **Linux** **Distributions**

- Linux Introduction
- **Linux Distrios and Shell Architecture**
- Operating System
- Files Permissions
- Environment Variables
- Commands
- Special Characters
- Shell Scripts

**Agenda.**



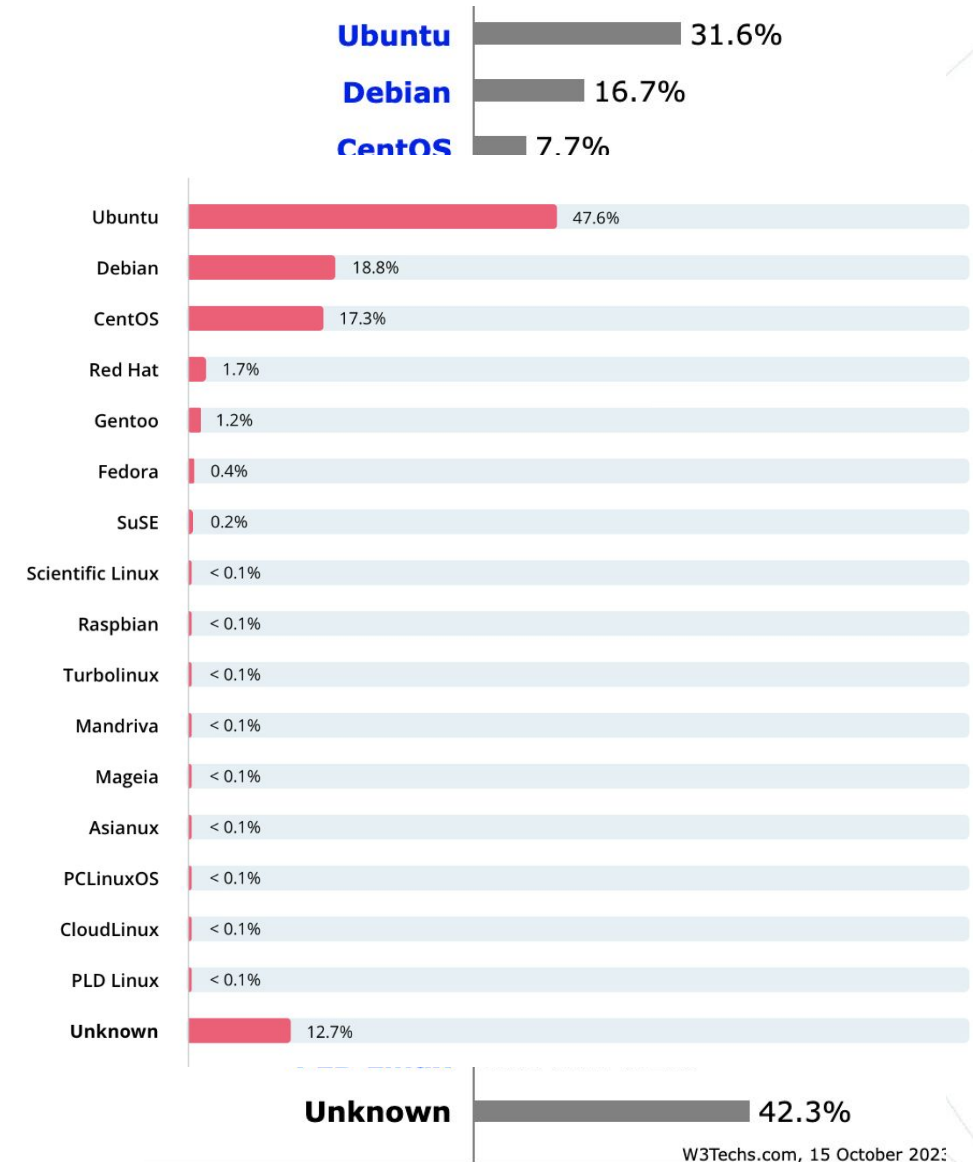
The diagram illustrates the lineage of Linux distributions, showing how they branch off from Ubuntu. The central node is Ubuntu, which branches into numerous other distributions. The branches are color-coded and labeled with the names of the distributions. Some distributions have further sub-branches, creating a complex web of relationships. The diagram highlights the diversity of the Linux ecosystem and the influence of Ubuntu as a base distribution.



# Linux Distrios

Linux OS

- **Ubuntu**
- **CentOS**
- **Debian**
- **RedHat**
- **PearlOS (for Mac)**
- **Fedora**



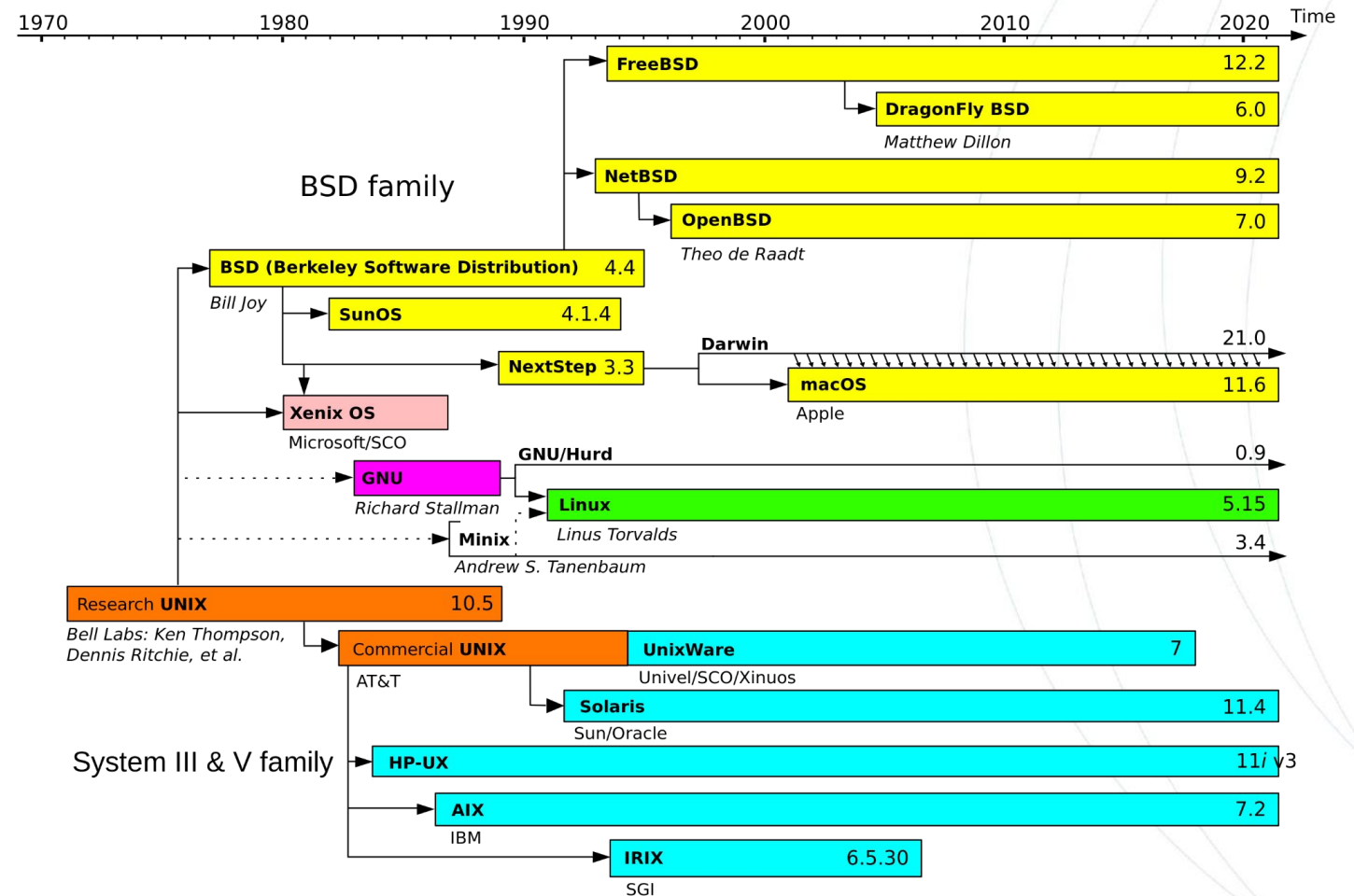
Percentages of websites using various subcategories of Linux



# macOS

Linux OS

macOS is unix-like, not exactly  
linux which is open source





# Why Linux

Linux OS

Advantages	Disadvantages
Open Source	Learning Curve
Stability and Reliability	Software Compatibility
Security	Hardware Support
Customizability	Fragmentation
Cost	Lack of Standardization





# **Linux** **Operating System**



- Linux Introduction
- Linux Distros and Shell Architecture
- **Operating System**
- Files Permissions
- Environment Variables
- Commands
- Special Characters
- Shell Scripts

**Agenda.**



# What is operating system?

Linux OS



# How do we work with an OS?

Linux OS



# What about Windows?

Linux OS




# Is it possible to use Linux on Windows?

Linux OS



# What do you use an operating system for?

- Accessing folders and files
- Running applications
- Edit documents/files
- Play audio/video
- ????



# Linux Commands

- Linux Introduction
- Linux Distros and Shell Architecture
- Operating System
- **Linux Commands 101**
- Use Case: Working with Files
- Use Case: Environment
- Use Case: Special Characters
- Use Case: Shell Scripts

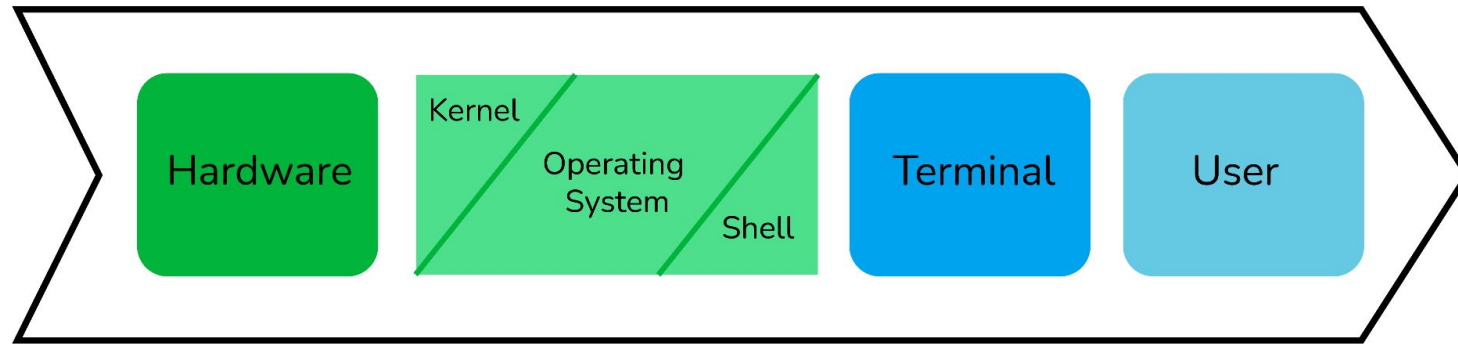
**Agenda.**





# Shell

## Linux Commands 101



Components Of Shell Program



**Kernel:** Makes the communication between the hardware and the software possible.

**Shell:** An interface that takes input from user in form of commands, processes it and gives an output.

**Identify what shell is being used:**

```
echo $0
```



# Bash

Linux Commands 101





# Bash

Linux Commands 101



## Some basic commands to get started

Command	Details
who	This will display information about who is logged on, both yourself and others.
whoami	This will display information about the current user i.e. You!
ls	ls is shorthand for LiSt. It will display all the visible files within the directory that you are currently in. The files will be listed alphabetically by default, but this can be altered with flags.
pwd	This will Print the Working Directory. A fancy way of asking the computer to display information about where you are and what directory you are currently working in.
man	This will provide you with the manual for the given command.
tldr	tldr {need installation} to get a short list of the most common uses for that command.





# Commands— Command Structure

A Linux command may include 3 parts: **Command**, Options, Arguments

```
1 $ command options arguments
```

- **Options:** describes the command behaviour
- **Arguments:** are generally files and folders.
- **Option example:** `ls -al ~/demo`
  - `ls` is the command to list the files and folders.
  - `-al` is the option to list all the files in the folder even include all the hidden files and folder.
  - `~/demo` is the directory that we list.



## Commands— How to find Helps

How to find a help when you don't know how to use a command, or even don't know the command?  
There are several ways:

- `command -- help` : detail of the command
- `man command` : manual of the command
- `info command` : manual of the command
- `whatis command` : a brief explanation of the command

useful link: <https://vitux.com/get-help-on-linux-shell/>





# Bash Demo



# **Linux**

## **Use Case: Working with Files**

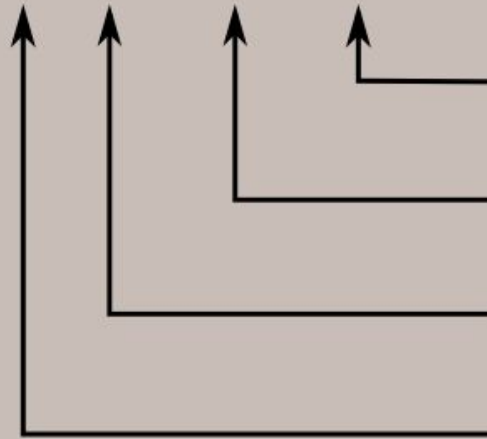




# File Permissions

- File permission rules

- rwx rwx rwx



Read, write, and execute permissions for all other users.

Read, write, and execute permissions for the group owner of the file.

Read, write, and execute permissions for the file owner.

File type:  
- indicates regular file  
d indicates directory



# Change permissions

## Change Mode

Change file owner mode

```
sudo chmod u+x(rw) <filename>
```

Change file group mode

```
sudo chmod g+x(rw) <filename>
```

Change file all users mode

```
sudo chmod a+x(rw) <filename>
```

## Change mode in number format

```
sudo chmod 755 <filename>
```

## Change File Owner, Group and Mode

Check file Group

```
sudo chown <new owner username> <filename>
```

Change file Group

```
sudo chgrp <new group name> <filename>
```

- digital place one “7” represent owner.
- digital place two “5” represent group.
- digital place three “5” represent all user.
- permission execute=1, write=2, read=4.
- 5 means read(4) + execute(1).



Try in system



## Commands— Create files

- **touch filename.txt** : create file with touch
- **echo “Hi” > filename.txt** : create file with echo
- **cat <<EOF > filename.csv** : create file with multi-lines





## Commands— Files and Directories

- **rmdir dir1**: rmdir only work for empty directory.
- **rm** remove a file or directory
  - **rm file1**: is used to delete the file named 'file1'
  - **rm -d dir**: delete an empty directory. -d means delete a dir.
  - **rm -rf dir1**: if a dir contains a file, we use -rfits contents recursively. option -f is “force”, -r recursive
- **mv dir1 new\_dir**: is used to rename or move a file or folder (directory).
- **cp** copy files
  - **cp file1**: is used to copy a file.
  - **cp file newfile**: copy file to a directory

useful link: <https://www.temok.com/blog/400-linux-basic-commands-you-should-know/>





# Commands— Files and Directories

How navigate in the linux file systems? How to create files or directories in the system?

- **cd** go to somewhere
  - **cd ..** :used for going back one level.
  - **cd ../ ..** :used for going back 2 levels
  - **cd -** : used for going (return) to the previous directory.
- **ls** list the files in the directory
  - **ls -l**: shows the details of files and folders in a directory
  - **ls -a**: shows hidden files.
- **tree** shows files and folders in tree form starting from the root
- **mkdir** creates a folder or directory
  - **mkdir dir1**: creates a folder or directory named 'dir1'.
  - **mkdir dir1 dir2**: creates two folders or directories simultaneously
  - **mkdir -p dir1/dir2/dir3**: use option -p to create multiple directories.
- **history** shows previously used commands.
  - **history | grep -i "cat"** : search specific command



- Linux Introduction
- Linux Distros and Shell Architecture
- Operating System
- **Linux Commands**
- Files Permissions
- Environment Variables
- Special Characters
- Shell Scripts

**Agenda.**



# Linux

## Use Case: Environment



# Environment Variables

## What is Environment Variables?

An environment variable is a variable whose value is set outside the program, typically through functionality built into the operating system or microservice. An environment variable is made up of a name/value pair.

## What is Local Variable?

Local Variables – A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.

## What is difference between environment variable and local variable?

- Environment variables are valid **system-wide**.
- Local Variable that is valid in the **current shell** only.
- Environment Variables are usually used as variables **for other programs**.
- Local Variables are usually used as **variables in a shell script**.
- Set Local Variable: **KEY=VALUE**
- Set Environment Variable: **export KEY=VALUE**





# Environment Variables

## Operation on Environment Variables

Set Environment Variables

```
export VARIABLE_NAME=VALUE
```

Unset Environment Variables

```
unset VARIABLE_NAME
```

list all Environment Variables

```
printenv
```

check if the Environment Variable exists

```
echo $VARIABLE_NAME
```

## Persisting Environment Variables

*When an environment variable is set from the shell using the export command, its existence ends when the user's sessions ends. To make an environment persistent for a user's environment, we export the variable from the user's profile script.*

Open the user's profile into a text editor

```
nano ~/.bashrc
```

Add the export command

```
export VARIABLE_NAME=VALUE
```

Apply the changes to bash\_profile

```
source ~/.bashrc
```




Try in system



# Linux

## Use Case: Common Commands



# Linux

## Use Case: Text Manipulation



## Commands— Print

- **echo** print out, the most common use for print
  - **echo Hello, World!** : print out 'Hello, World'
  - **echo -e** : Display a message containing special characters.
  - **echo -e 'Hello\nWorld' >> file.txt**: Redirect the print result to a file
  - **echo \$USER**: print out the USER variable.
  - **echo `date`**: Print out the system date
- **cat** print out a file to display
  - **cat file1**: print out a file
  - **cat file1 | command (sed, grep, awk, grep, etc ...)> result.txt**: general syntax to manipulate a text in a file and write the result in a new file.
- **printf** prints a formatted string.
  - **printf "My name is \"%s\".\nIt's a pleasure to meet you." "John"**  

```
My name is "John".  
It's a pleasure to meet you.
```
  - **%s**, which interprets the argument "John" as a string and inserts it into the output.
  - **\n**. indicates a new line.
  - useful link: <https://www.computerhope.com/unix/uprintf.htm>





## Commands— Text Manipulation and filtering

- **awk** mostly used for pattern scanning and processing.
  - **awk '{print \$5}' filename**: Print the fifth column in a space separated file.
  - **awk '/something/ {print \$2}' filename**: Print the second column of the lines containing "something" in a space separated file.
- **sed**
  - **sed 's/hello/world/' input.txt**: replace all occurrences of 'hello' to 'world' in the file input.txt
  - **sed -e '1d' result.txt**: remove the first line from the file result.txt
  - **sed -n '/stringa1/p'**: display only the lines that contain the word "string1".
  - **useful link**: <https://www.gnu.org/software/sed/manual/sed.html>





## Commands— grep

**grep** Grep Search command for Linux systems.

- Used to search for a pattern/text within one or more files
- It accepts regular expression and wildcards
- Examples :

```
grep "sales" emp.txt
grep 'abc' /etc/passwd/*.txt    #search for lines containing 'abc' in
/etc/passwd
grep null *.py                  #search multiple files
```

```
ls -al | grep ".py"
```

Useful link: <https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/>





# Text Editors – vim

## Open a file

```
vim <filename>
```

### Normal Mode:

```
keyboard <ESC> # normal mode
```

```
keyboard <right/left> or <be> # move cursor
```

```
keyboard <x> # delete
```

### Close files

```
:q # close file without saving
```

```
:wq # close file with saving
```

### Insert Mode(editing mode):

```
keyboard <I> # Insert mode
```

### Close files

```
keyboard <ESC> # switch to normal mode
```

```
:q # close file without saving
```

```
:wq # close file with saving
```



Try in system



# Text Editors – nano

## Editing in Nano

Open a file

```
nano <filename>
```

Search words

```
ctrl <w> and <ENTER>
```

Save

```
ctrl <o> and <ENTER>
```

Exit

```
ctrl <x>
```

## Set Nano as default:

Open ~/.bashrc

```
nano ~/.bashrc
```

export the two Environment Variables

```
export VISUAL=nano  
export EDITOR="$VISUAL"
```

Activate ~/.bashrc

```
source ~/.bashrc
```



Try in system





# Linux Shell Script

- Linux Distrios and Shell Architecture
- Files Permissions
- Environment Variables
- Commands
- Special Characters
- Shell Scripts

**Agenda.**



## Shell Script – What is a shell script?

- A plain text file created using text editor and contains sequence of commands:
  - Provides interface for user to use operating system services.
  - Text editors: vi/vim, emacs, nano, Atom, Sublime Text, Notepad++, TextWrangler
- Executes the commands listed in the script (just as if you had typed the commands in yourself)
- Can automate tedious and repetitive tasks
- The shell gets started when user logs in or start the terminal.
- Terminal in Linux/MacOS or Command prompt in Windows OS
- Different types of Shells in Linux:
  - The Bourne Shell (sh)
  - The Bourne Again Shell (bash) **#Default shell for all Linux versions**
  - The C Shell
  - Korn Shell



## Shell Script – Why and When use?

### Why write Shell Script?

- Speed of development.
  - Shell scripts eliminate repetitive tasks through automation
- No worrying about low-level programming objects.
- Reduce the chance of errors.
- Interactive debugging
- Ease and speed of learning.

### When to write a Shell Script?

- Repetitive tasks
- You need to enter multiple shell commands
  - You will need to do it again in the future
- Need to delegate a task to less experienced or knowledgeable staff.
- System admins use shell scripting for routing backups
- System Monitoring





## Shell Script – Steps to run a shell script.

- Step 1. Open a file using vim

```
vi demoscrypt.sh
```

- Step 2. Write the code

```
#!/bin/bash  
echo "Welcome! This is my first shell script!"  
echo "Displaying the list of directories: "  
ls
```

- Step 3. Make the script executable using chmod command

```
chmod u+x script_name
```

- Step 4. Run your script

```
/path_to/demoscript.sh or bash demoscrypt.sh or ./demoscript.sh
```





# **Linux**

**Use Case: Other Commonly  
Used Commands**

- Linux Distros and Shell Architecture
- Files Permissions
- Environment Variables
- Commonly Used Commands
- Special Characters
- Shell Scripts

**Agenda.**



# Special Characters

Style	Description	Example
\$	Variable expression	x=1 echo \$x
\$?	Exit code	[[ 4 > 5 ]] echo \$?
“”	quote	echo "Today is \$(date)"
&	Background job	ls &
#	Comments	<i># this is a comment</i>
	Pipe, split 2 commands	ps   grep 'bash'
~	Home directory	cd ~
;	Command Separator	((a=2+3)); echo "a=\$a"







# Special Characters

Style	Description	Example
>	redirect to a file by overwriting	echo "Welcome to Linux" > my_file_1.txt
>>	redirect to a file by appending	echo "Learn latest tips about Linux" >> my_file_2.txt
>& or &>	forward standard output or standard error	1>/dev/null 2>&1

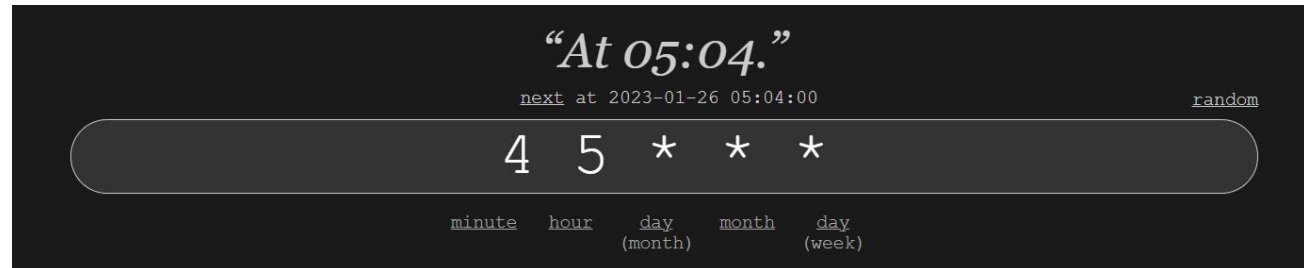




# Cron Job

## A basic way to schedule job

- To list the cronjobs: *crontab -l*
- To edit or create a new cronjob: *crontab -e*
  - Set schedule (reference website: <https://crontab.guru/>)



- Set which file you are going to schedule  
*\*/30 9-17 \* \* 1-5 /path/to/your/bash/script/ &>file or dev/null*



# Exercises after Class



WeCloudData

📍 500-80 Bloor Street West, Toronto

📍 ON

[www.weclouddata.com](http://www.weclouddata.com)

