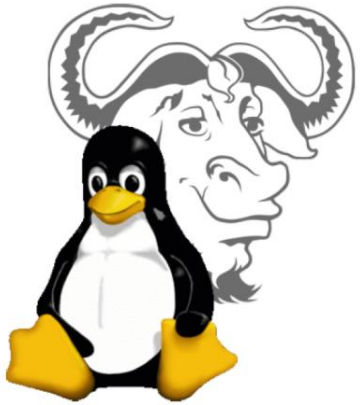
Hands on GNU/Linux!

ME597c

Prof. Yu Hue

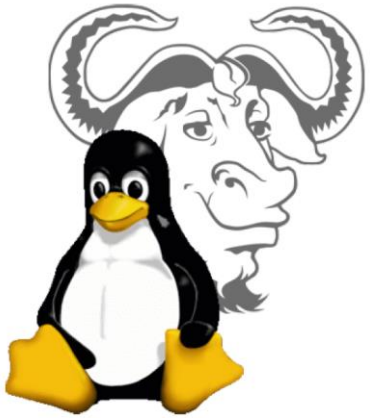
A. Alghooneh

What is GNU/Linux? And what is this Ubuntu?



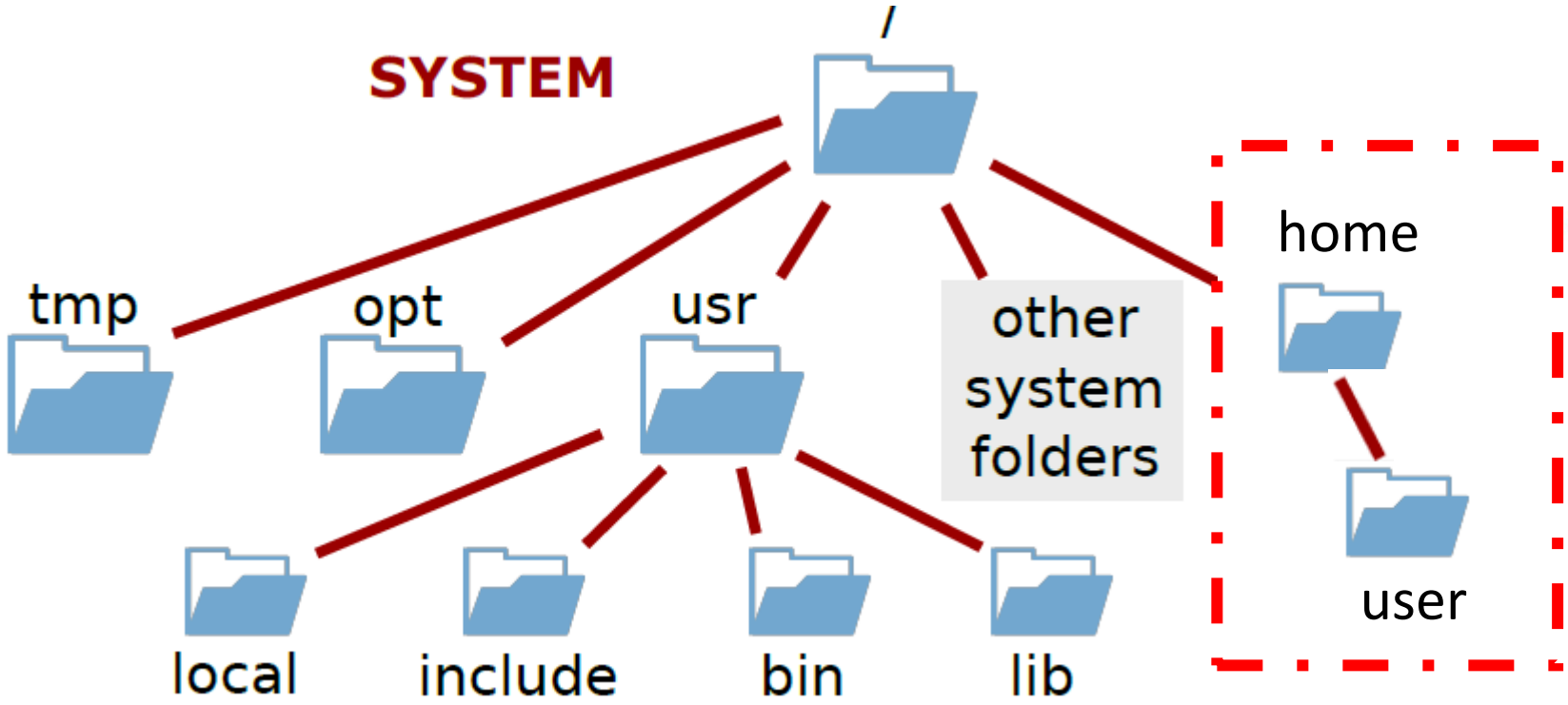
- Linux is a free **Unix-based** kernel; the kernel is the one responsible for hardware and software interactions
- Linux is **free** and **very powerful**; hence popular, examples are Android, ChromeOS, Servers, etc
- Ubuntu is one of the distributions of **Linux**
- Ubuntu is one of the best operating systems for **robotics**, easy to install and work around with
  - Robotics tools are available for this distro
- The **Unix** operating system
- A documentary on **Linux**

Why Ubuntu and Linux?



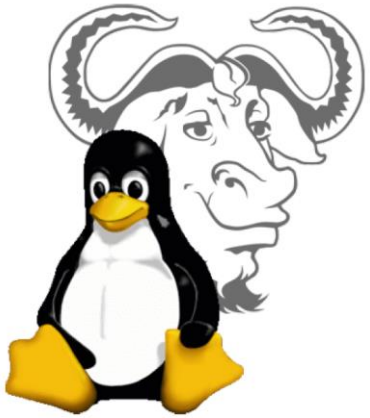
- Free
- Open-source, e.g. privacy privileges, immediate support
- More than 50 percent use **Linux-based** platforms
- It's all about the **Workflow**!

File system structure



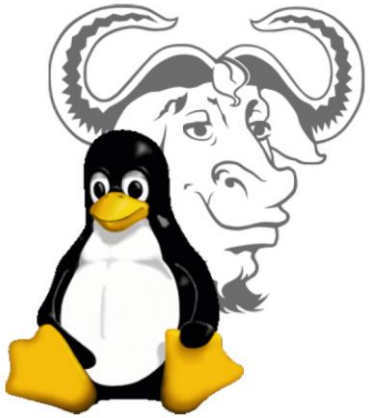
- We call root **/** because of the tree structure of Linux
- [Video explaining the directories in full]
- No weird volume letters like **C** or **D**

The Terminal

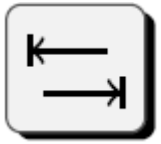


- There are two ways to open Terminal: **`Ctrl+Alt+T`** or by using the mouse (right click -> open terminal).
- To **p**rint the current **w**orking **d**irectory you can use **`pwd`** command
- Use **ls <dir>** to see the content of the directory (blue = directory)
- Use **`cd <dir>`** to change directory
  - Absolute path: start with a / (forward slash) (root)
  - Relative path: start not with a slash (name or .(working directory) or ..(parent directory))
- Use **`cd ..`** to go the parent directory
- Use **`cd .`** for the current directory
- Use **~** for home directory

Bash Command Convention

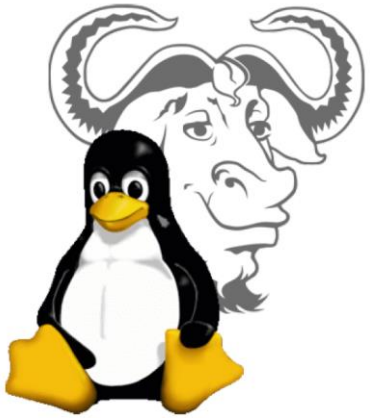


- **`${PATH}/command [options] [parameters]`**
- **`[options]`** are put like **`-x`** or **`-f`**
- **`[parameters]`** are the input of the command
- E.g. **`tar -xfv nvidia_3070gtx.tar`**
- **`.file`** **means that "file" is hidden**
- Change the PATH variable so as to just use **`command [options] [parameters]`**
- Pressing the tab key can auto-complete your command



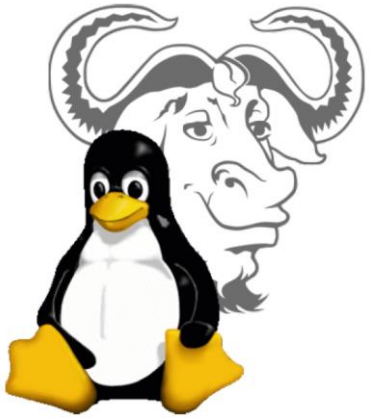
- Pressing the tab key twice list all the available suggestions

Most used commands



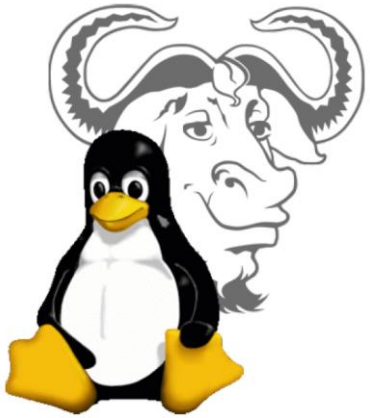
- `mkdir` makes a new directory e.g.; `mkdir catkin_ws`
- `rm` removes a file e.g.; `rm ~/catkin_ws/gpu_info.cpp`
- `rm -r <DIR> removes a directory`
- with each of the above commands there are options available that you can see calling it with the option `-h` or –help or `man [command]` to find out
- `cp` to copy file from `; cp -r <source file|dir> <destination dir>`
- same with `mv` (move) ; `mv <source dir> <destination dir>`
- you can use place holders : `cp *.cpp <destination dir>` this

Most used commands



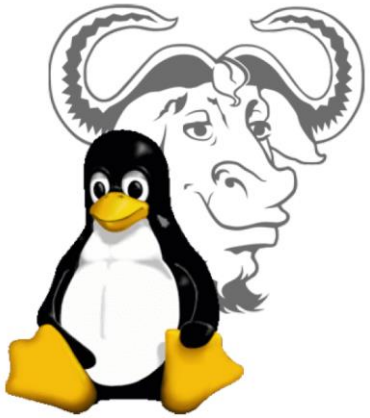
- How to search?
- Look at **`grep (content)`** and **`find (names)`**
- To search for a **filename** in a folder use **`find <in-folder> -name <filename>`**
- To search for a certain **string** in a file use **grep**; **`grep <string> -n <where>`**
- Add with "**echo**" and remove with "**sed**"
- Now that you found the file to see the content you can use **less** (leave with q, search with /)

Most used commands



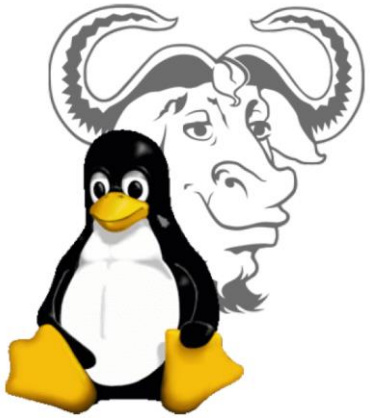
- You can connect commands to each other
- To run them simultaneously **`command1 && command2`**
- To call them after each other **`command1;command2`**
- What is the difference between **`;`** and **`&&`** ?
- **Piping** is a handy concept
- How piping works? It connects the output of one command to the input of the other e.g. **`command1 | command2`**
- For example **`ps -aux | grep firefox`**
- Now try searching for a file like this:
  `find <directory> | grep <filename>`

Ctrl+Shift+Alt Delete?



- **`Ctrl+C`** crashes the current process in the terminal
- Get the list of the processes – **`ps -aux`**
- Pick up the id <pid>
- **`kill -9 <pid>`** closes the process with the id of **`<pid>`**
- **`killall <pname>`** kill all the process with name **`<pname>`**
- **`htop`**

Install library program or whatever



Look into /etc/apt/sources.list, keyrings, sources.list.d

Update your installed software

- **`sudo apt update:`** update the repo list and certs
- **`sudo apt upgrade:`** upgrade takes the action

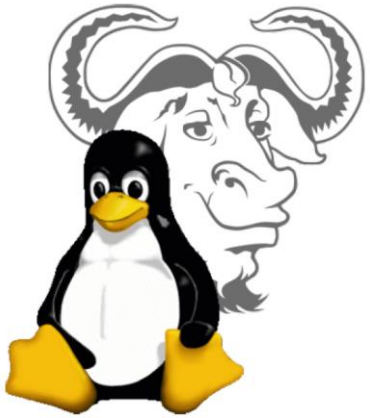
Install or remove <program>

- **sudo apt install <program>**
- **sudo apt remove <program>**
- **sudo apt purge <program>**
- **`Sudo apt autoremove`**

Search for a package

- **`apt-cache search SEARCH_TERM`**

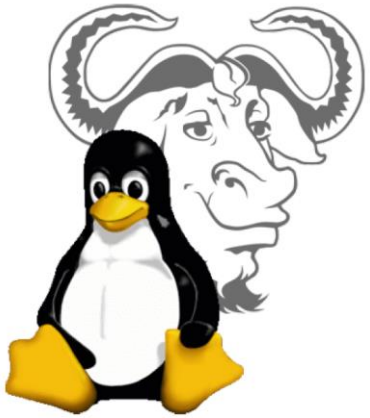
Environment and Path variables



- **environment variables** are a set of dynamic named values stored within the system that is used by the applications.
- The **PATH variable** is an **environment variable** containing an ordered list of paths that **Linux** will search for executables when running a command.
- You can use **echo $variable** to see the variable content.
- You can change or define an environment variable with export
  e.g. export ROS_DOMAIN_ID=4

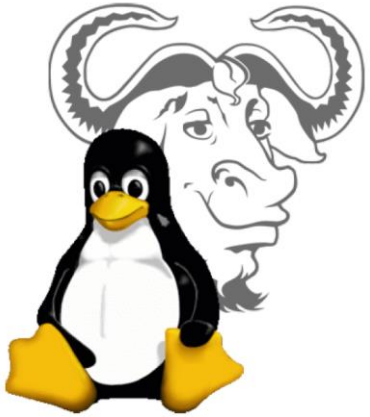
- You can add a path to the path variable like:
  e.g. export PATH= $PATH:/new/path

~/.bashrc and ~/.profile



- The **.bashrc** file is a script file that's executed when a subshell is triggered. e.g. each time you open a terminal
- The **.profile** file is a script file that's executed when a user is logged in.
- Both of the scripts are in bash and can contain environment settings and configs.
  - **For instance, when you are installing cuda, and cuDNN, you should configure the environment variables in either .profile or .bashrc**
  - **or when you install something like ros, you should do it in .bashrc**
- E.g. check the **~/.bashrc** with **cat ~/.bashrc | less**

-stdout and stderr

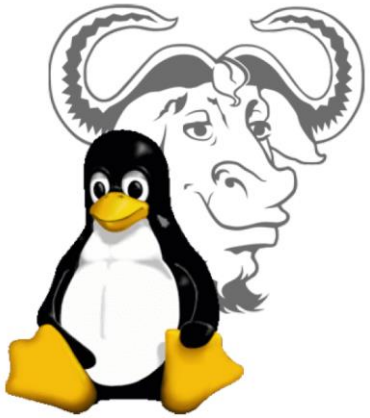


- **stdout**: standard output
- **stderr**: standard errors
- They are ways for the program to talk to the user
- Your **screen** is the standard output,
- By default, commands take input from the standard input and send the results to the standard output.
- Standard error sometimes denoted as stderr, is where error messages go. By default, this is your screen.
- Most of the time you need to log the stdout and stderr, in order to show the system behavior to another person (TAs)

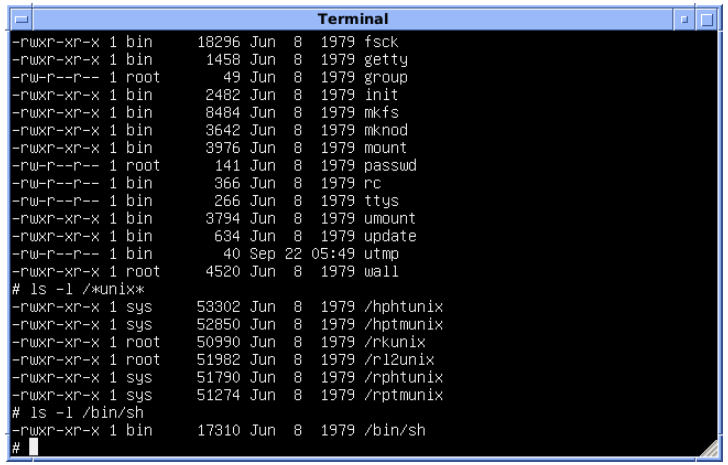
-Save stdout and stderr to a file

- $> **ls > log.txt** – logs the stdout into log.txt
- $> **ls >> log.txt-** append the logs to the log.txt
- $> **ls | tee log.txt** – shows both on the screen and saves to log.txt
- $> **wrongCommand &> log.txt** – saves the stdout and stderr into log.txt
- $> **wrongCommand &>> log.txt** – appends stdout and stderr to the log.txt

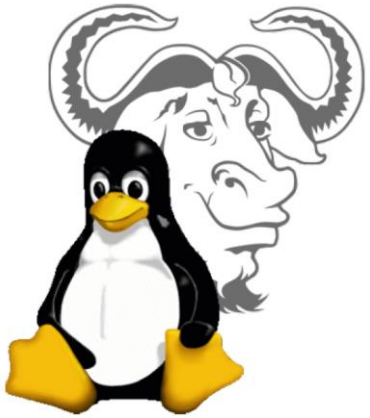
Bourne and Bourne again!



- It's the command line interpreter
- Bash is a Unix shell and command language (.**bash**)
- It is a replacement for the Bourne shell (**.sh**) from Unix.
- Bash written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. Bourne Again Shell (Bash)
- Bash First released in 1989
- Bash he default login shell for Linux distributions.
- Bash was one of the first programs Linus Torvalds ported to Linux, alongside GCC.
- (Bash vs Bourne) e.g. Bash is more similar to modern prog langs, has functions and arrays

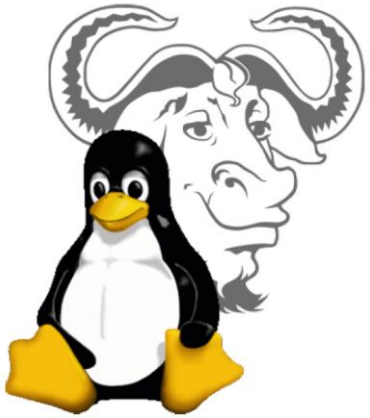


Shell scripts?!



- **Bash** scripts are essentially logical sequence of the same Linux commands
- Logical in the sense that you can use **For-loops, if-else, whiles and etc**
- Difference between source and bashing a script
- So the necessary tutorial for bash for this course is :
  - Define a Variable in bash- var=value, or export envVar=value
  - Write a for in bash- for iter in {range};do{}done
  - Compare integers and strings in bash- (-gt : >) (-lt :<) (-eq: ==) (-ne: !=)
  - Write if-else, then in bash- if [statement]; then {} fi
  - Read user inputs and arguments in bash- $1,$2,…,$n

Shell scripts?!



- **Interesting shells to write:**
  - Write an install script for ros?
  - Write a script that list the devices IPv4s on the local network?
  - Write a script that make the user look away every 20 minutes?
  - Write a script that finds a project/document in system and copy that into a thumb drive/ or any other folder?