Advanced System Programming

# Agenda

- Introduction
- Course Content Discussion
- Raspberry Hardware setup
- Qemu setup
- UML setup
- System Programming introduction

# Ubuntu System

# Ubuntu System - Labsetup

- Ubuntu 20.04 installation on VM or physical system
  - RAM: 4GB – 8GB
  - Hard disk space: 50GB (minimum)
  - Capable of connecting to internet.
- Following packages should be installed on Ubuntu [ This is performed using command apt-get install]

  **Example:**

  sudo apt-get update

  sudo apt-get install <package name>
  - build-essential
  - vim
  - tree
  - htop
  - openssh-client
  - linux-tools-common
  - linux-tools-generic

# Ubuntu System - Labsetup

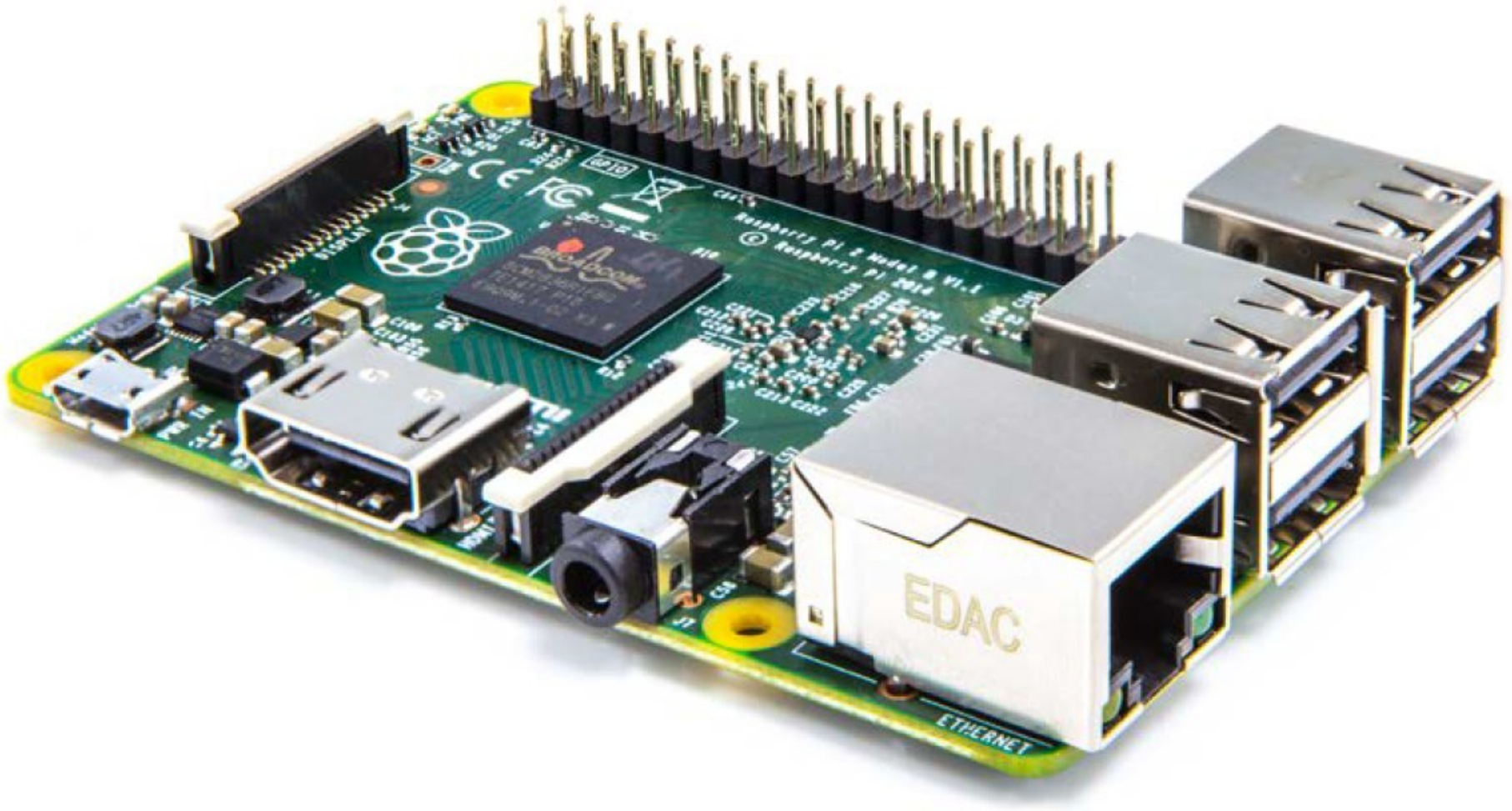
- linux-tools-`uname -r`
- qemu
- qemu-utils
- qemu-kvm
- virt-manager
- libvirt-daemon-system
- libvirt-clients
- bridge-utils
- qemu-system-arm
- valgrind
- mlocate
- crossbuild-essential-armhf
- git
- bc

# Ubuntu System - Labsetup

- bison
- flex
- libssl-dev
- make
- libc6-dev
- libncurses5-dev
- Fuse
- gdb-multiarch
- autoconf
- openocd

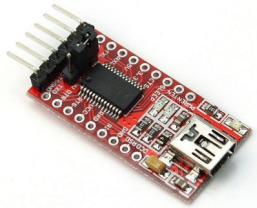
- Setup Eclipse C/C++ for debugging and development.

# Raspberry pi Board



# Raspberry pi hardware required

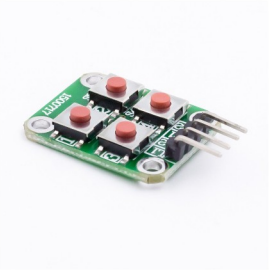
- The set of peripheral required for labs is given below:
  - SD CARD ( https://robu.in/product/sandisk-micro-sd-sdhc-16gb-class-10-memory-card-upto-98mb-s-speed/ )
  - CARD READER ( https://robu.in/product/high-speed-micro-sd-card-reader/)
  - Serial cable ( https://robu.in/product/pl2303-ta-download-cable-usb-ttl-rs232-module-usb-serial/ )
  - LED (https://robu.in/product/grove-blue-led/)
  - SPI OLED (https://robu.in/product/0-96-oled-display-module-spii2c 128x64-7-pin-blue/)
  - I2C OLED (https://robu.in/product/0-96-inch-i2c-iic-oled-lcd module-4pin-with-vcc-gnd-blue/)

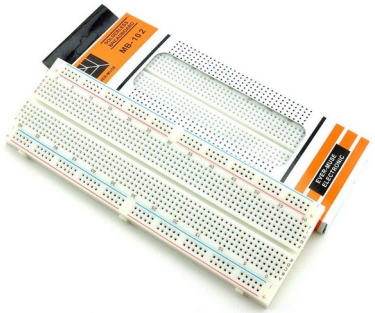
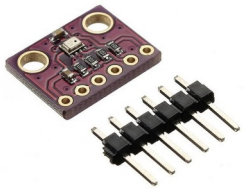
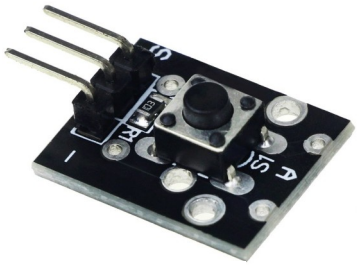






# Raspberry pi hardware required

- BMP280 (https://robu.in/product/bmp280-barometric-pressure-and-altitude-sensor-i2c-spi-module/)
- Matrix Keypad (https://robu.in/product/2-x-2-matrix-4-push-button-keyboard-module/)
- Wires (https://robu.in/product/20-cm-40-pin-dupont-male-male-male-female-female-female-cable-combo/)
- Bread Board (https://robu.in/product/mb102-830-points-solderless-prototype-pcb-breadboard-high-quality/)
- Push button (https://robu.in/product/momentary-tactile-push-button-module-dc-5v-switch/)
- Olimex JTAG (https://www.olimex.com/Products/ARM/JTAG/ARM-USB-OCD-H/)
- SPI to Ethernet Hardware TCP/IP W5500 Ethernet Network Module (https://robu.in/product/spi-to-ethernet-hardware-tcp-ip-w5500-ethernet-network-module/)



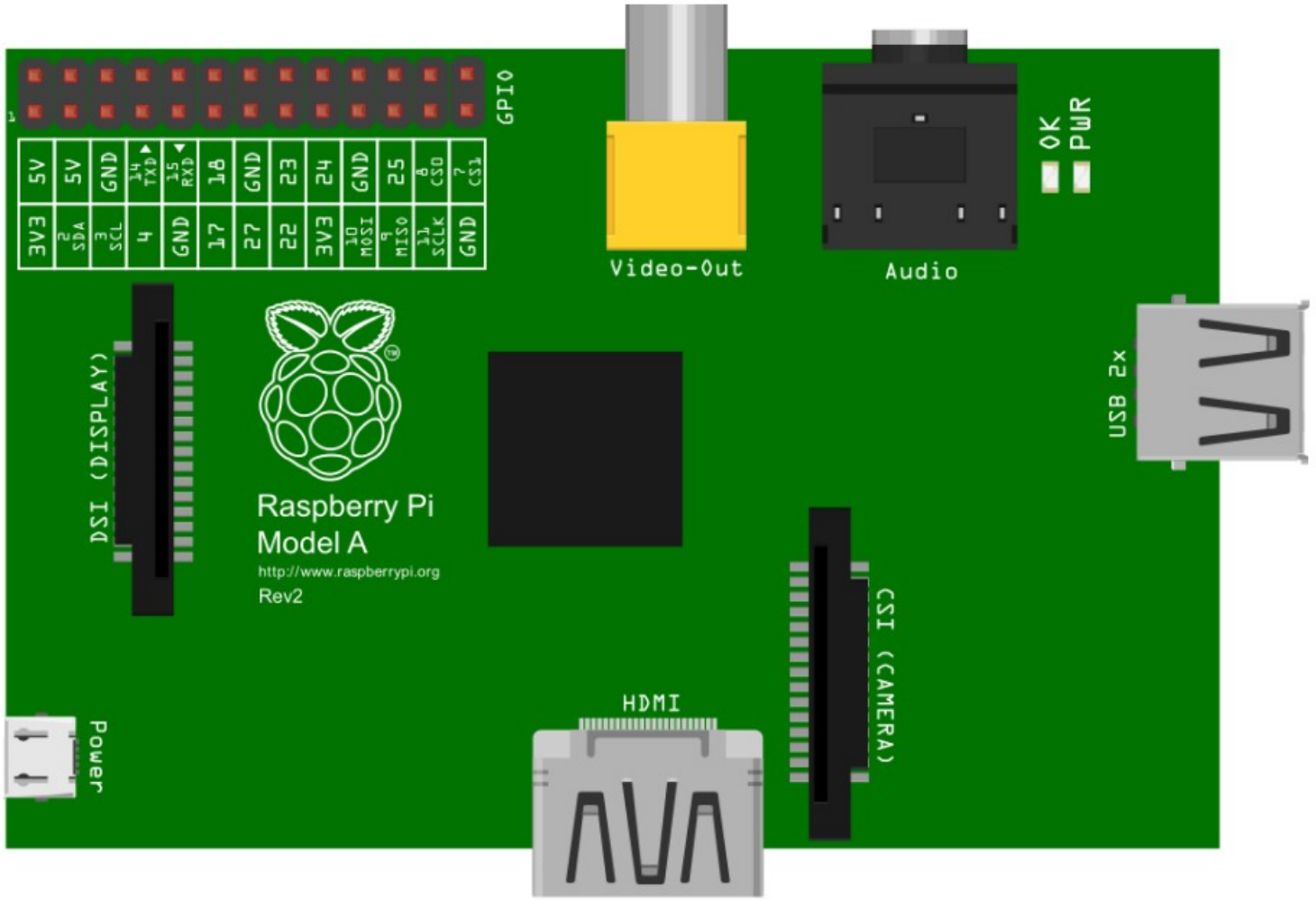




# Raspberry pi setup

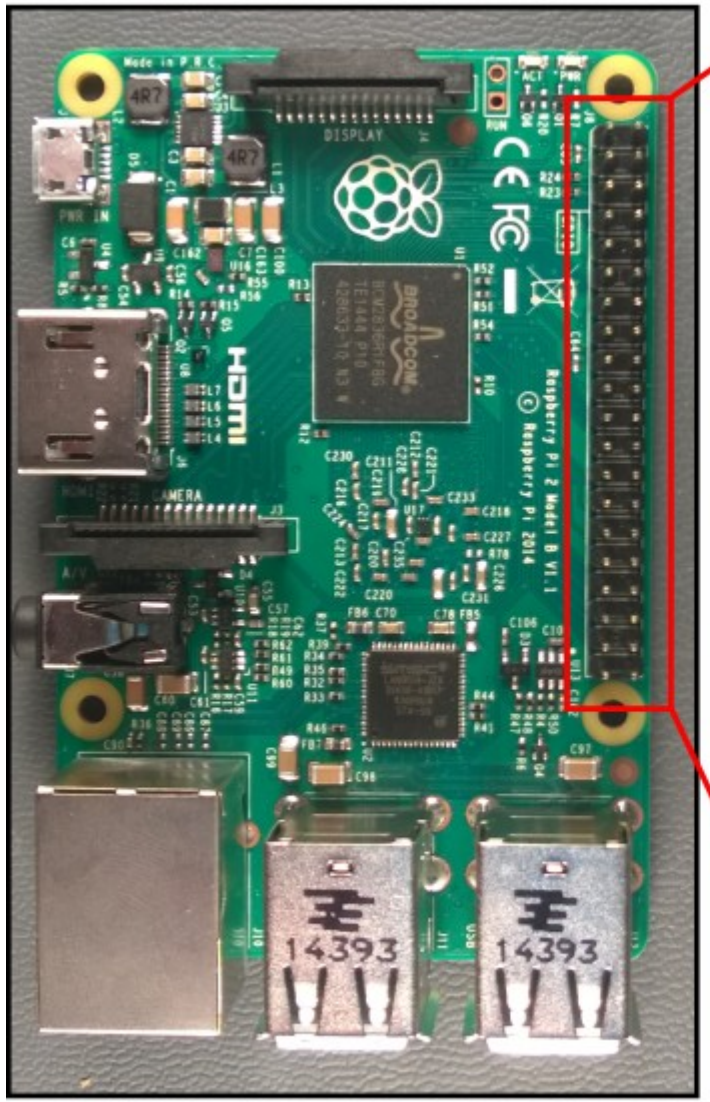
- Download Raspberry pi os and Setup the board.

https://www.raspberrypi.com/software/

# Setting up Raspberry pi Board



# Setting up Raspberry pi Board

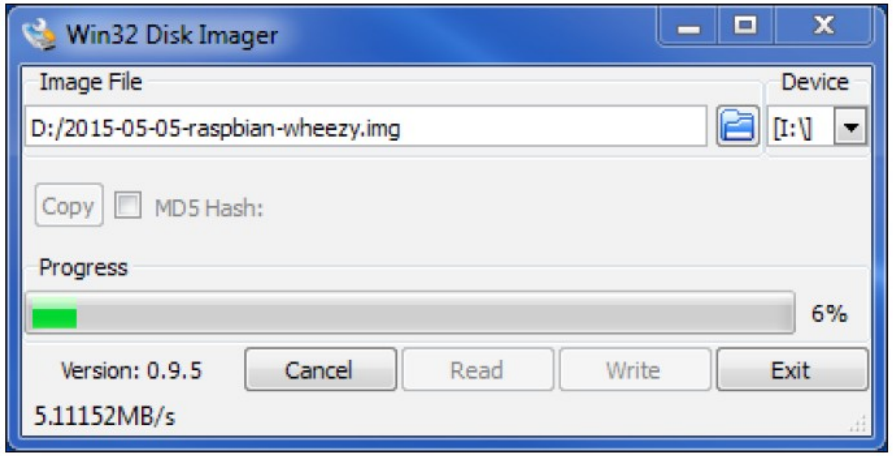


| Alternate Function | | | | | | Alternate Function |
|---|---|---|---|---|---|---|
| | 3.3V PWR | 1 | | 2 | 5V PWR | |
| I2C1 SDA | GPIO 2 | 3 | | 4 | 5V PWR | |
| I2C1 SCL | GPIO 3 | 5 | | 6 | GND | |
| | GPIO 4 | 7 | | 8 | UART0 TX | |
| | GND | 9 | | 10 | UART0 RX | |
| | GPIO 17 | 11 | | 12 | GPIO 18 | |
| | GPIO 27 | 13 | | 14 | GND | |
| | GPIO 22 | 15 | | 16 | GPIO 23 | |
| | 3.3V PWR | 17 | | 18 | GPIO 24 | |
| SPI0 MOSI | GPIO 10 | 19 | | 20 | GND | |
| SPI0 MISO | GPIO 9 | 21 | | 22 | GPIO 25 | |
| SPI0 SCLK | GPIO 11 | 23 | | 24 | GPIO 8 | SPI0 CS0 |
| | GND | 25 | | 26 | GPIO 7 | SPI0 CS1 |
| | Reserved | 27 | | 28 | Reserved | |
| | GPIO 5 | 29 | | 30 | GND | |
| | GPIO 6 | 31 | | 32 | GPIO 12 | |
| | GPIO 13 | 33 | | 34 | GND | |
| SPI1 MISO | GPIO 19 | 35 | | 36 | GPIO 16 | SPI1 CS0 |
| | GPIO 26 | 37 | | 38 | GPIO 20 | SPI1 MOSI |
| | GND | 39 | | 40 | GPIO 21 | SPI1 SCLK |

# Setting up Raspberry pi Board – SD Card Setup

- On a Windows PC, the best way to burn the image to your SD card is to use the Win32 Disk Imager utility. This can be downloaded from http://sourceforge.net/projects/win32diskimager.

- It doesn't have an installer, and launches directly from the EXE file. Now, it's time to create your SD card image:
  - Insert your SD card into the PC and launch the Win32 Disk Imager.
  - Select the SD card device drive letter (make sure it's right!).
  - Choose the Raspbian image file you've just downloaded.
  - Click on the Write button to create the SD card image.



# Setting up Raspberry pi Board – SD Card Setup

- On a Linux PC, you'll need to use the gparted and dd utilities to burn the image on your SD card. Carry out the following steps to create your SD card image:
  - Extract 2015-09-24-raspbian-jessie.img to your Home folder.
  - Insert your SD card into the PC.
  - If you're not already in a shell terminal window, open one (you can use Ctrl + Alt + T on most graphical-based desktop systems).
  - Type the following command in the shell terminal:

    $ sudo fdisk -l
  - In the list check, your SD card appears as a drive device (for example, /dev/sdb). It's crucial that you ensure you use the right device in the next step. We'll assume that your device is /sdb.
  - To burn the image to the SD card, type the following command:

    $ sudo dd if=2015-09-24-raspbian-jessie.img of=/dev/sdb
  - Hit Enter and go make a cup of tea or coffee as this will take a while. You'll know that it's finished when the command ($) prompt re-appears.

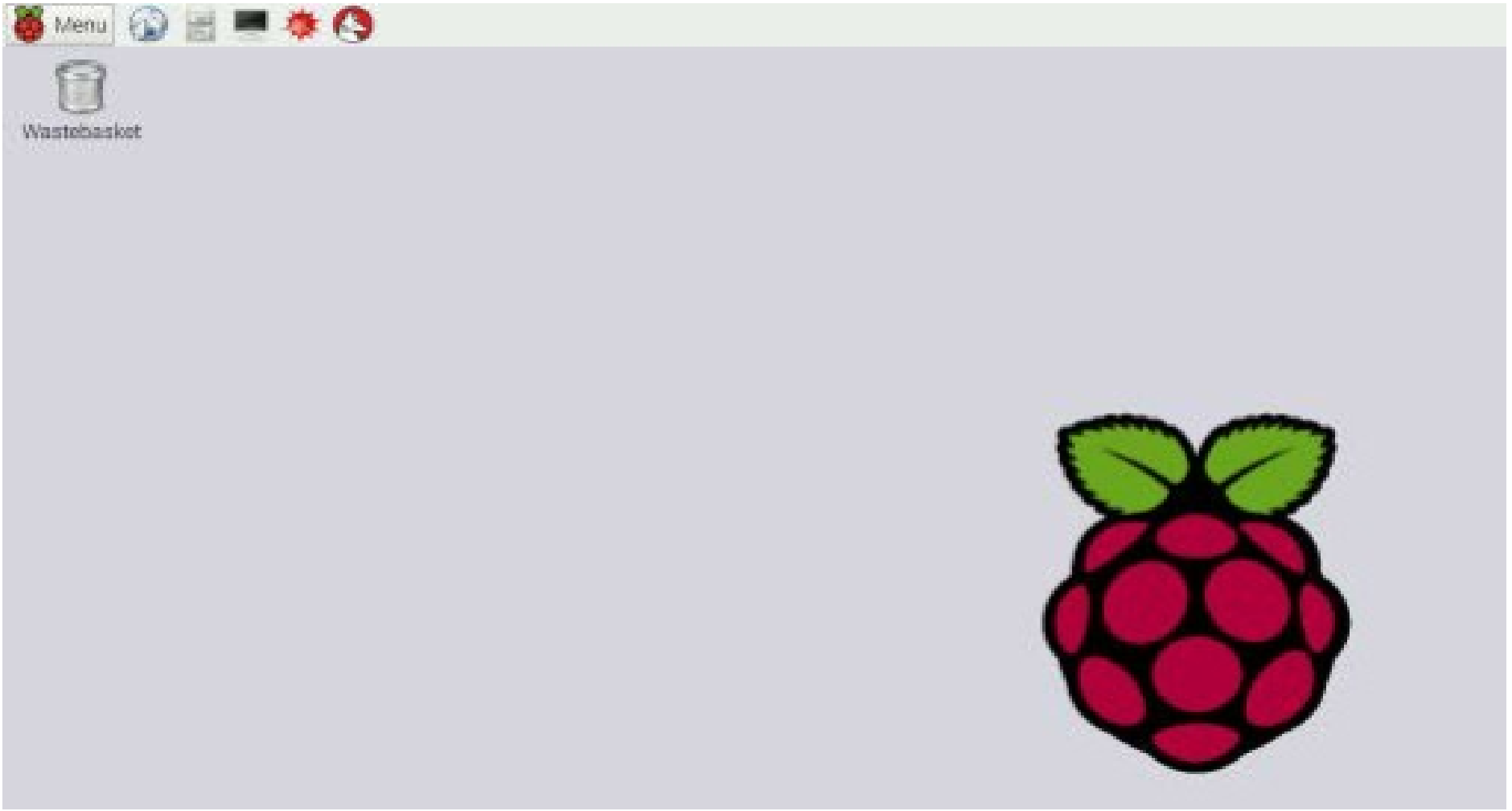
# Setting up Raspberry pi Board – SD Card Setup

- When the command prompt does re-appear, type the following command:

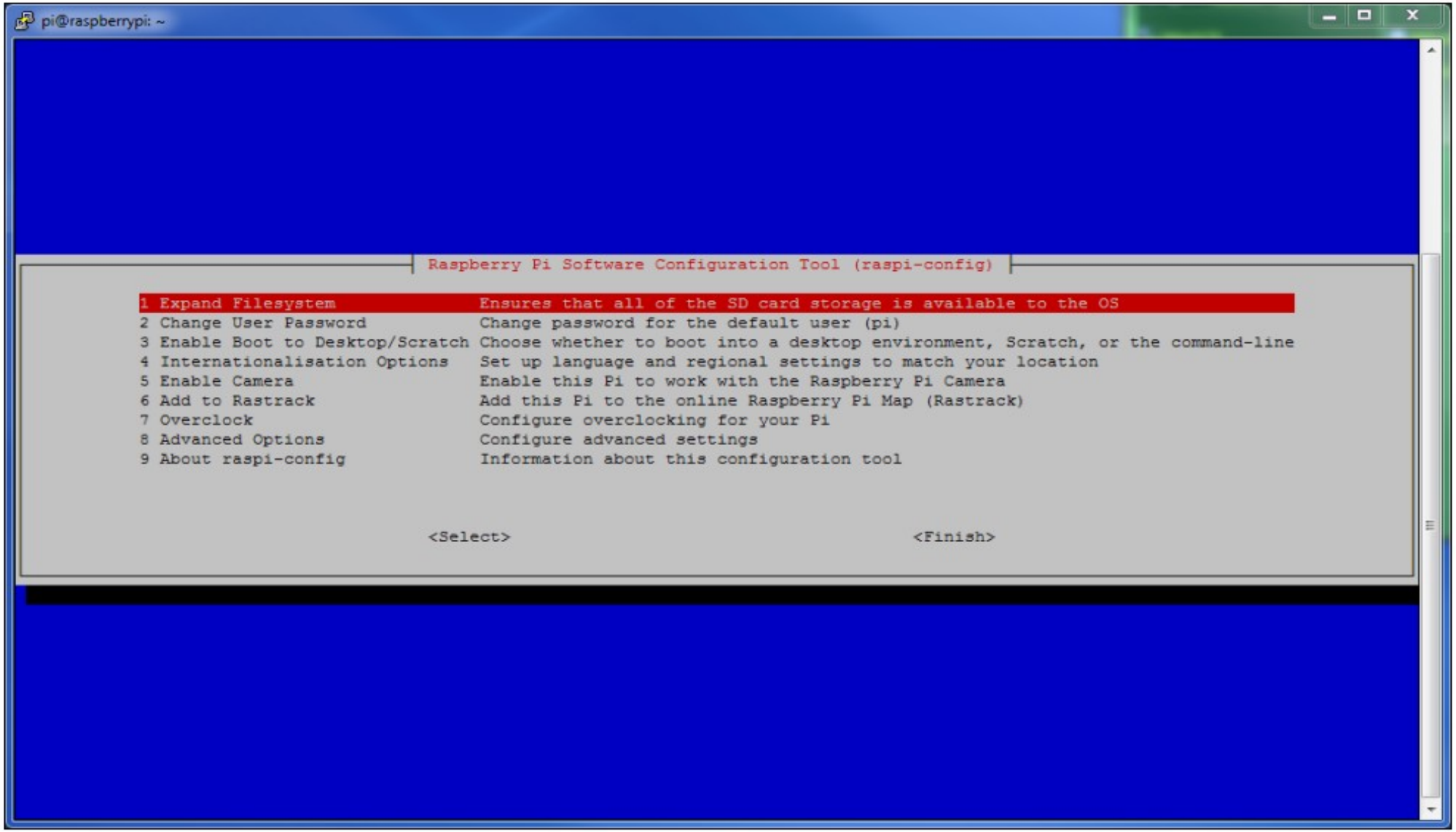
  $ sudo sync

- Once that command has finished, you can remove the SD card from the PC.

# Setting up Raspberry pi Board – SD Card Setup



# Setting up Raspberry pi Board – SD Card Setup



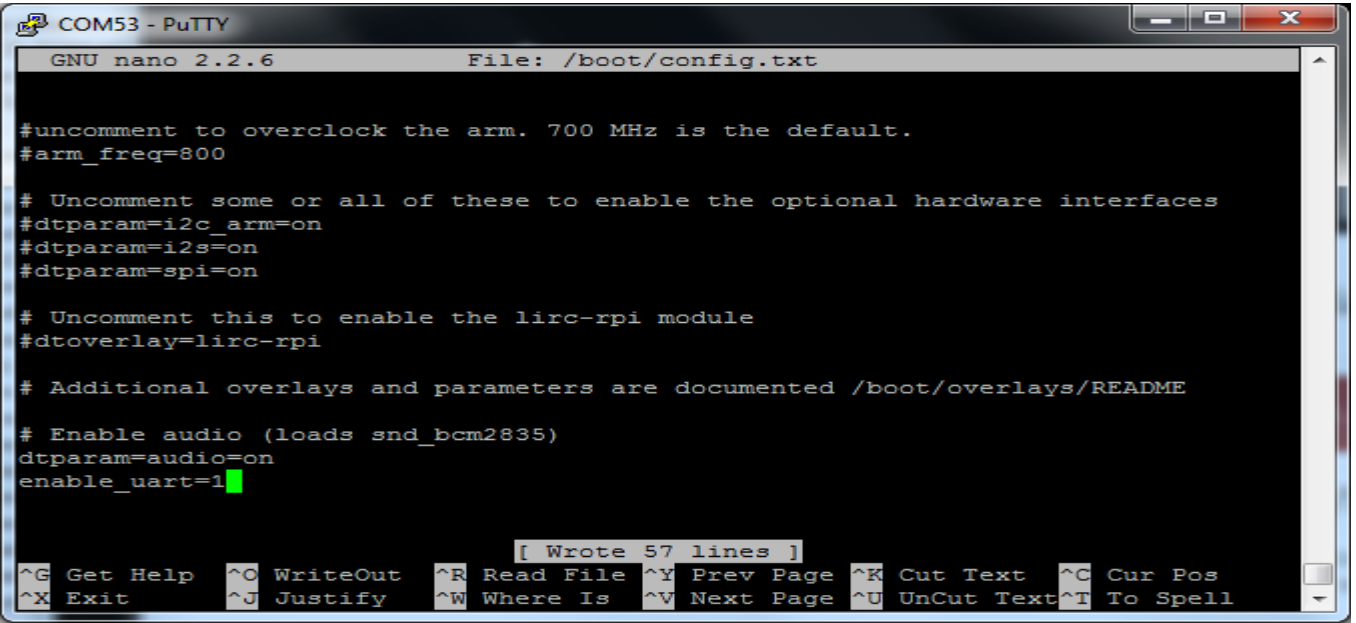
# Setting up Raspberry pi Board – Enabling Serial Console

You can enable/disable the serial console with either editing /boot/config.txt or raspi-config (which will edit /boot/config.txt for you)

❑Option 1. Enabling in /boot/config.txt

You can pop your SD card into a computer and edit config.txt with a text editor like SimpleText, WordPad or whatnot. You can also edit on a pi with sudo nano /boot/config.txt

At the bottom, last line, add enable_uart=1 on it's own line



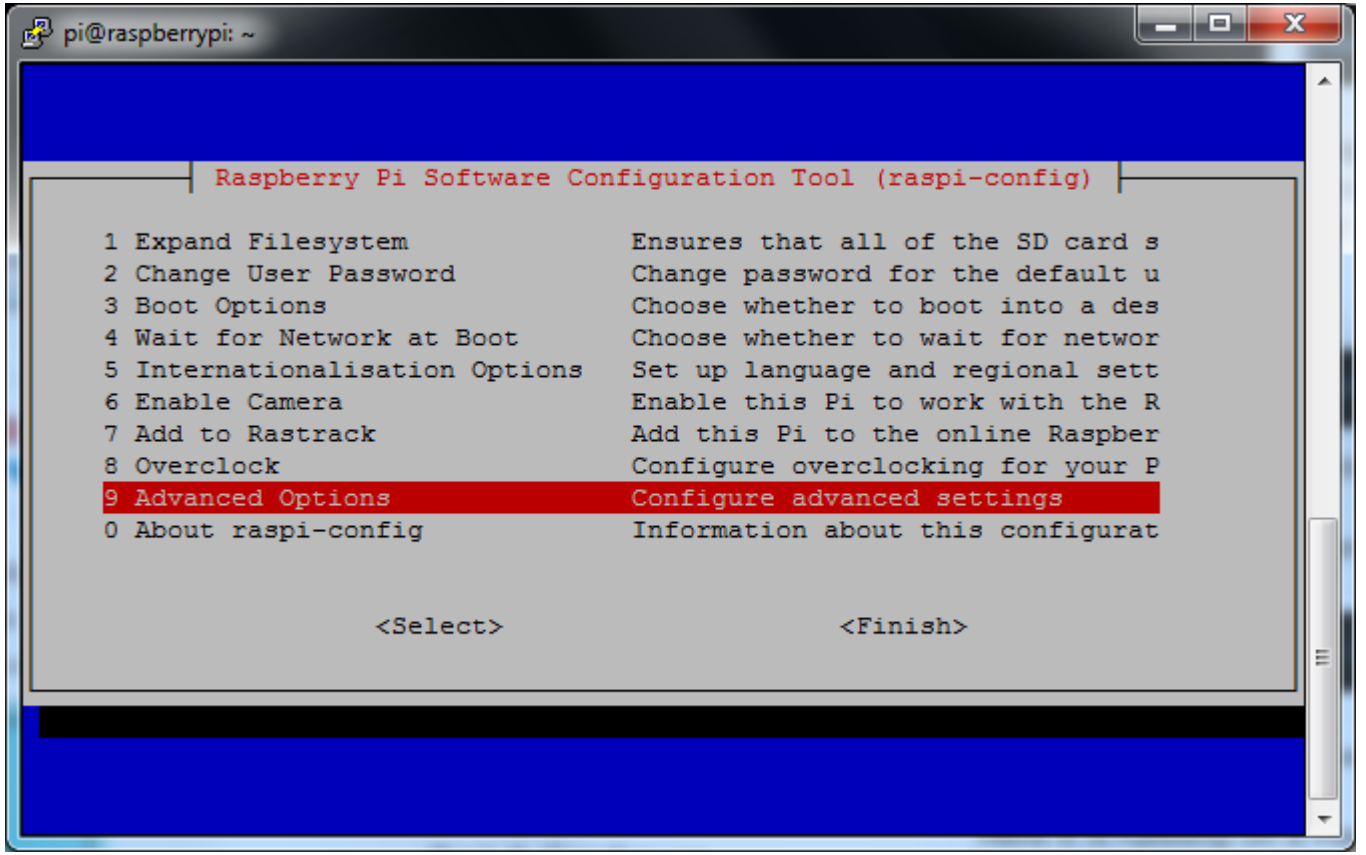
# Setting up Raspberry pi Board – Enabling Serial Console

❑ Option 2. Enabling via Raspi-Config

Using a monitor and keyboard, log into the shell and run

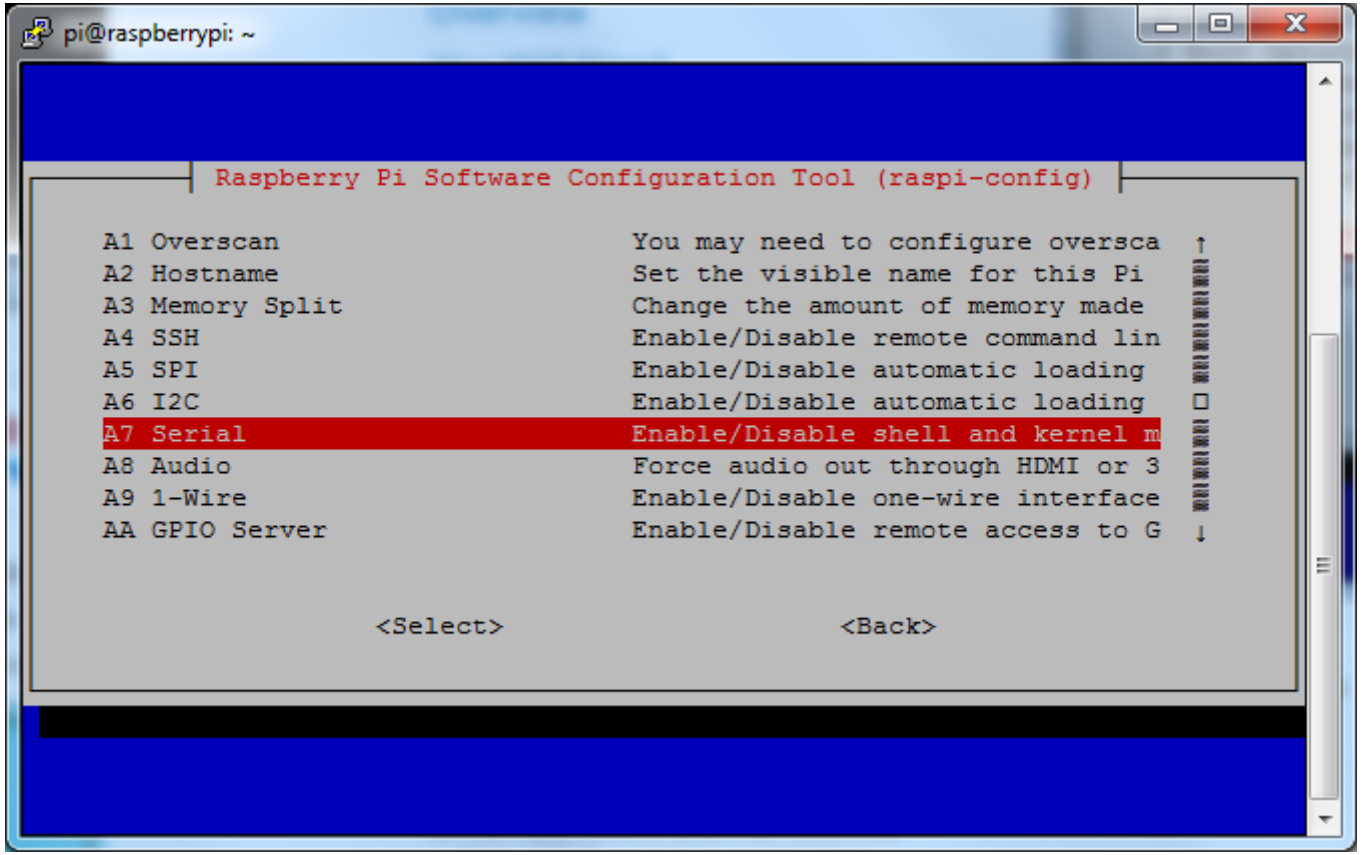
**# sudo raspi-config**

go down to Advanced Options



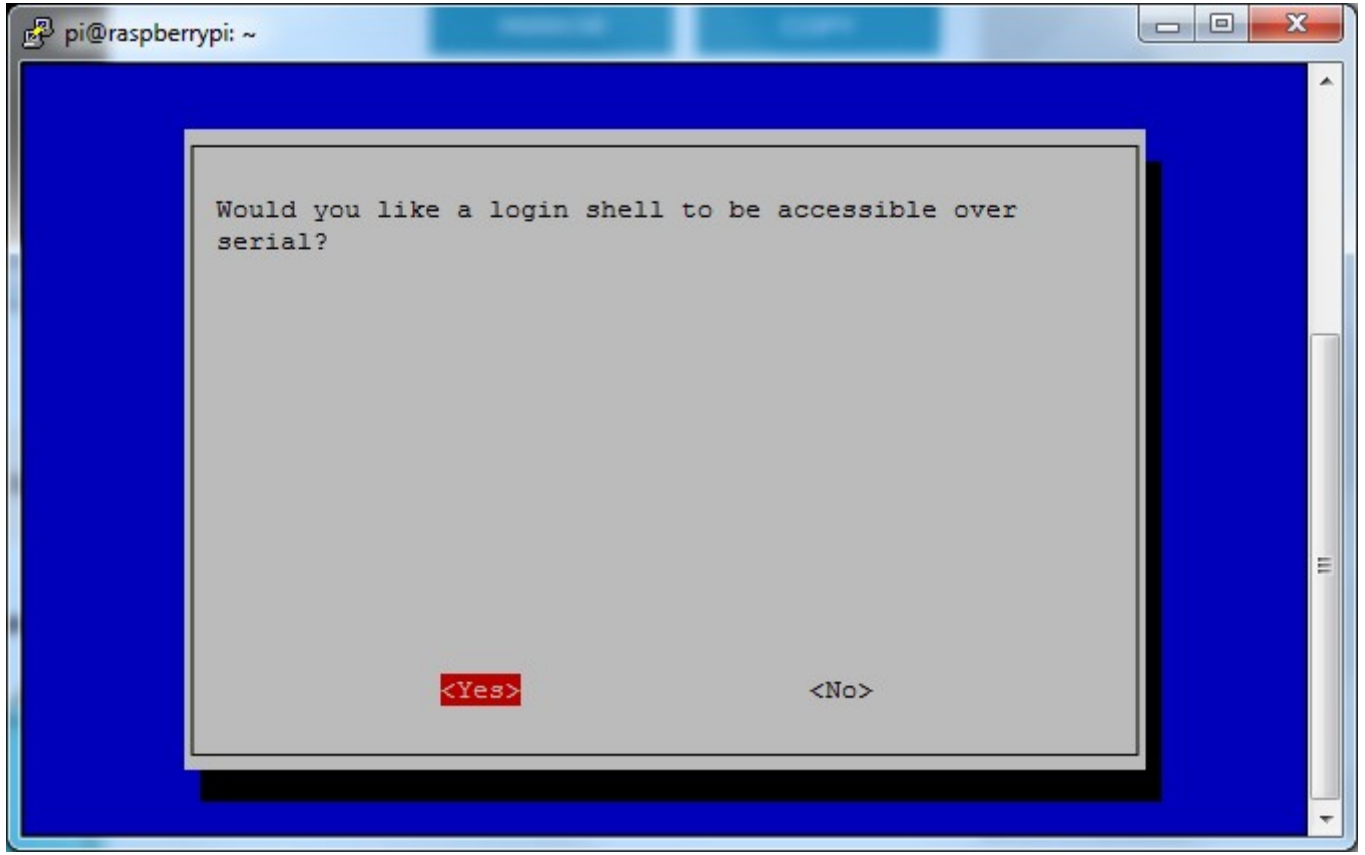
# Setting up Raspberry pi Board – Enabling Serial Console

Hit enter and then go down to Serial



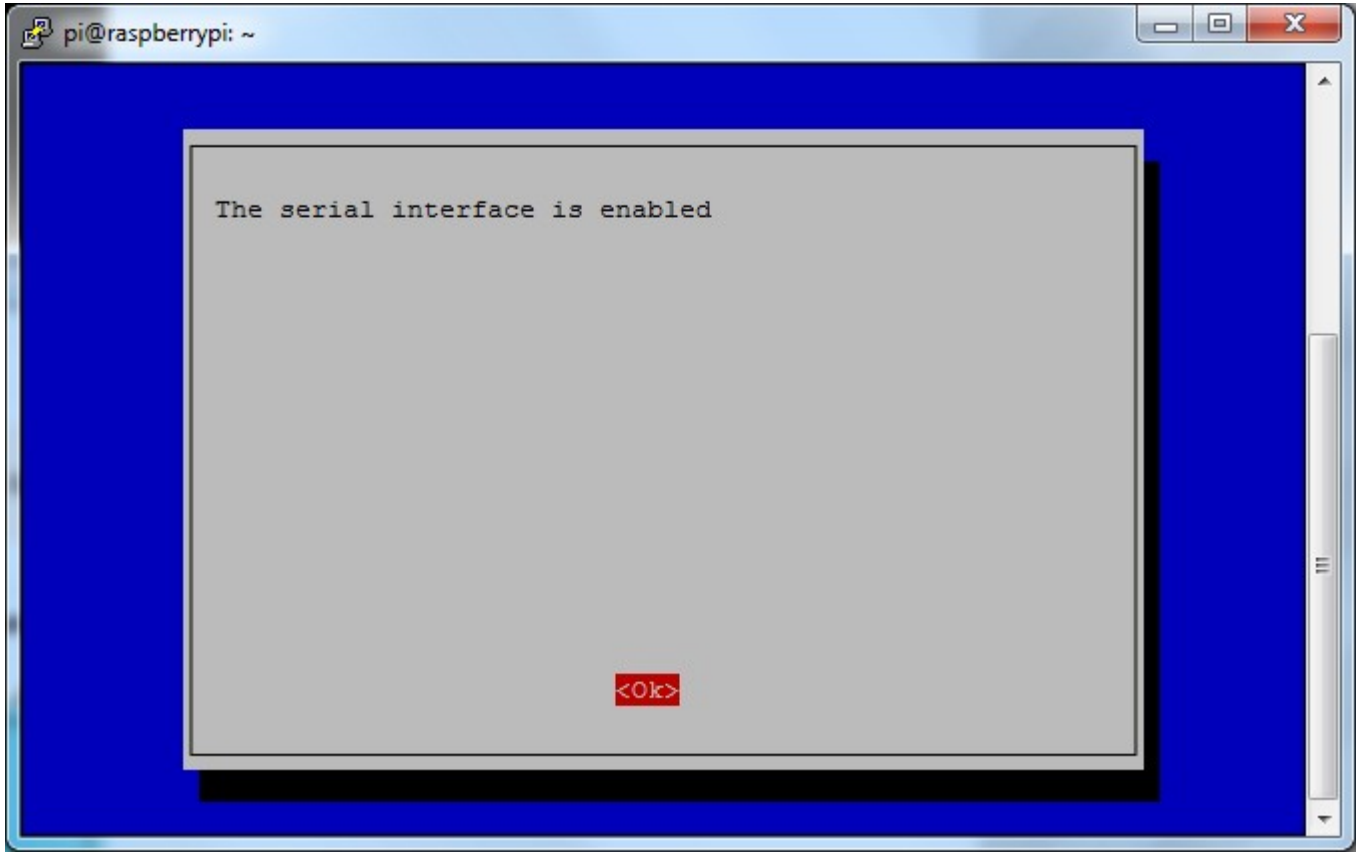
# Setting up Raspberry pi Board – Enabling Serial Console

Select Yes



# Setting up Raspberry pi Board – Enabling Serial Console

It should now be enabled

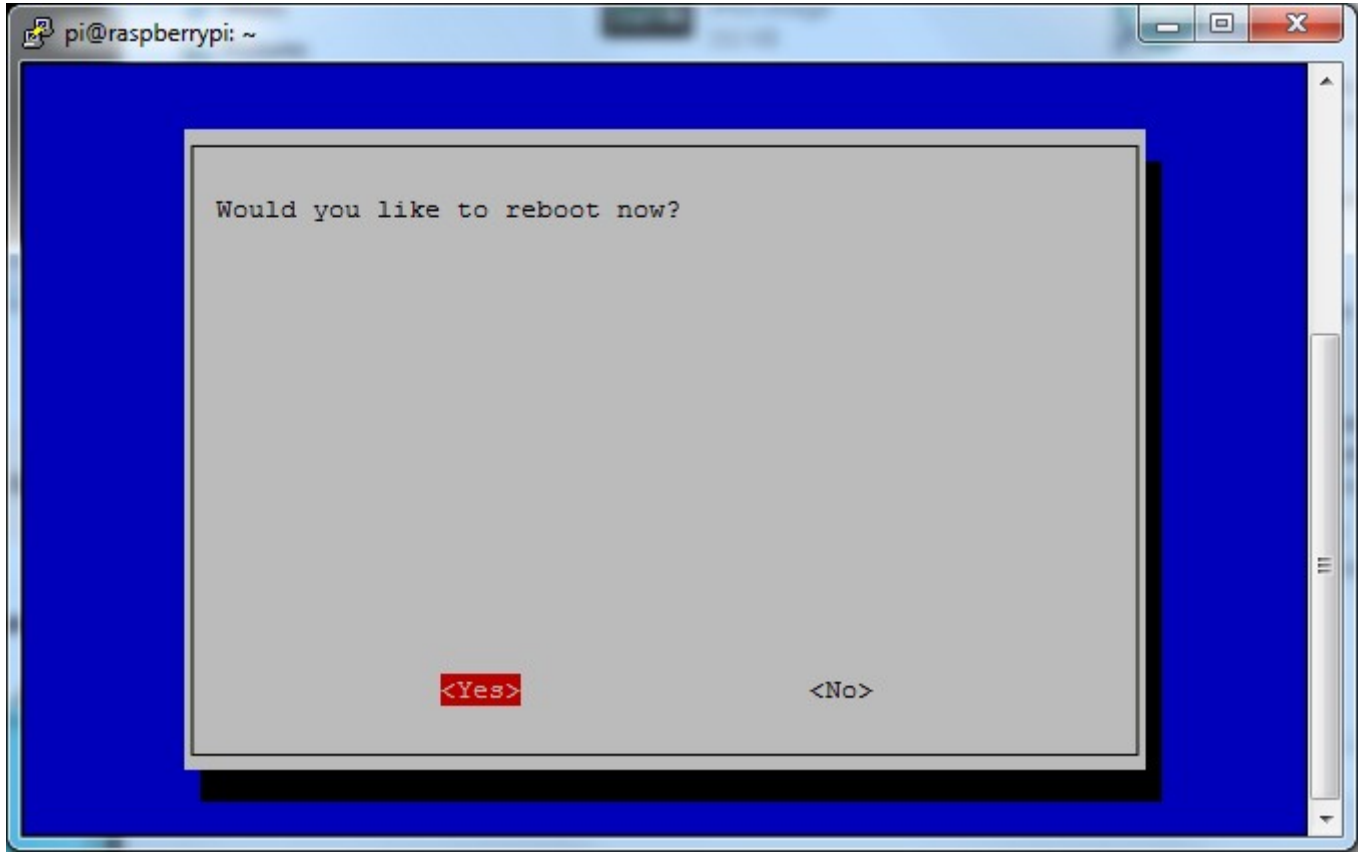


# Setting up Raspberry pi Board – Enabling Serial Console

Hit return then select Finish

When it asks you to reboot, go to Yes and hit return

OK the serial console is now enabled!



# Setting up Raspberry pi Board – Enabling Serial Console

https://www.raspberrypi.com/documentation/computers/config_txt.html

# Building UML Kernel Image

- Downloading the Kernel and extract

  www.kernel.org

  # tar -xvzf linux-5.1.16.tar.gz

- Configuring the Kernel

  # make ARCH=um menuconfig

- Build Kernel

  # make ARCH=um

- Boot system

  # ./linux  rootfstype=hostfs rootflags=/home/test/uml/rootfs/ rw mem=64M  init=/bin/sh

# Building Qemu Kernel Image

- Downloading the Kernel and extract

  www.kernel.org

  # tar -xvzf linux-5.1.16.tar.gz

  # cd linux-5.1.16/

- copy configuration file

  # cp arch/arm/configs/vexpress_defconfig  .config

- configure kernel

  # make menuconfig ARCH=arm  CROSS_COMPILE=arm-linux-gnueabihf-

- Build  kernel

  # make ARCH=arm  CROSS_COMPILE=arm-linux-gnueabihf-

# Building Qemu Kernel Image

- Build kernel modules

```
# make ARCH=arm  CROSS_COMPILE=CROSS_COMPILE=arm-linux-gnueabihf- modules
```

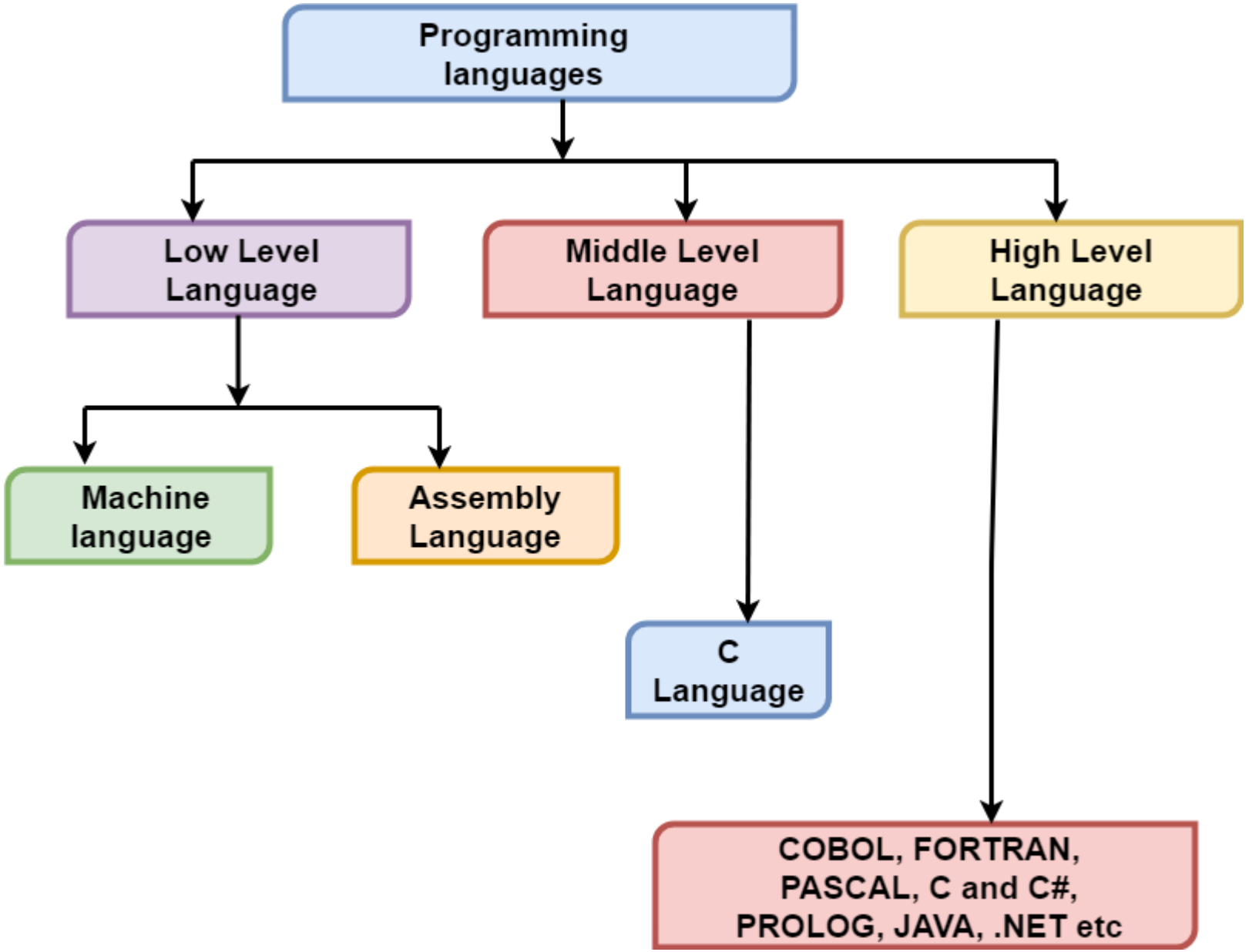
- Boot kernel image with GUI

```
# qemu-system-arm -M vexpress-a9 -m 512M -dtb linux-5.1.16/arch/arm/boot/dts/vexpress-v2p-ca9.dtb -kernel linux-5.1.16/arch/arm/boot/zImage -initrd rootfs.img.gz -append "root=/dev/ram rdinit=/linuxrc"
```

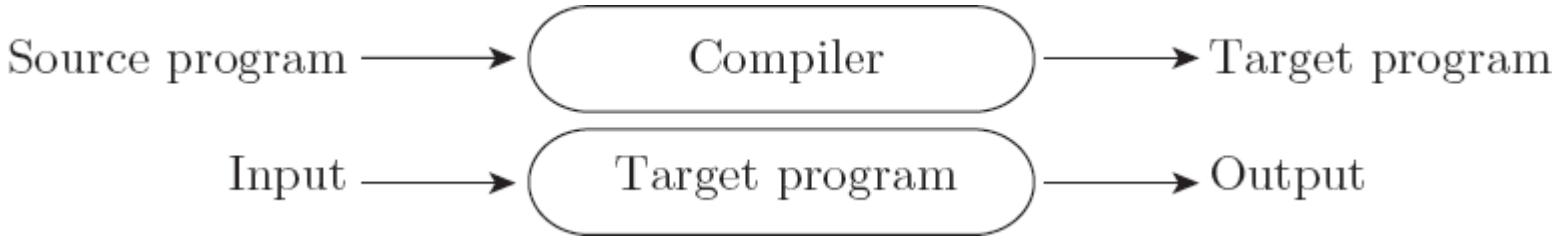
- Boot kernel image without GUI

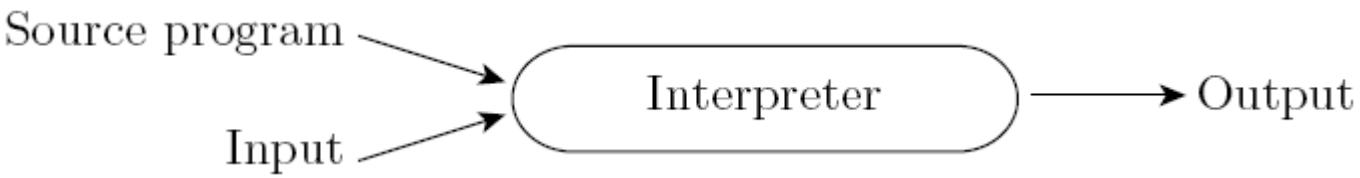
```
# qemu-system-arm -M vexpress-a9 -m 512M -dtb linux-5.1.16/arch/arm/boot/dts/vexpress-v2p-ca9.dtb -nographic -kernel linux-5.1.16/arch/arm/boot/zImage -initrd rootfs.img.gz -append "root=/dev/ram console=ttyAMA0 rdinit=/linuxrc"
```

# Classification of Programming Languages

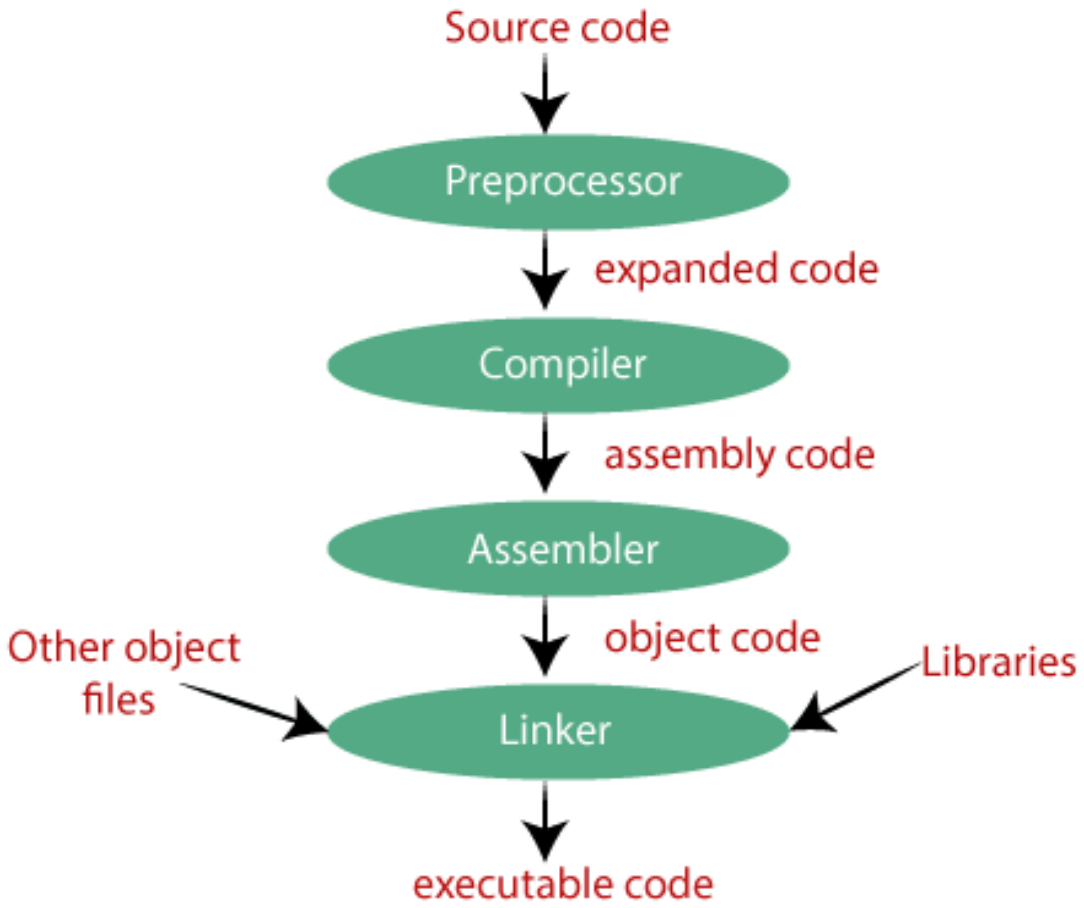


# Compilation V/S Interpreration



# Compilation Stages



# Introduction to Linux

- In the early 1990s, Torvalds became interested in a freeware product called Minix were written by Andrew S. Tanenbaum. Developed by Andrew S.Tanenbaum, Minix was a clone of the commercial UNIX operating system.

- Linux version 0.02, released on October 5, 1991, consisted of only the Linux kernel and three utilities:
  - bash : a command-line interface
  - update : a utility for flushing file system buffers
  - gcc : a C++ compiler

- Today used on 7-10 million computers with 1000's of programmers working to enhance it around the world.

# Introduction to Linux

- GNU Project: Richard Stallman on September 27th 1983.

- The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system.

- GNU's kernel isn't finished, so GNU is used with the kernel Linux. The combination of GNU and Linux is the GNU/Linux operating system, now used by millions.

# What is Linux

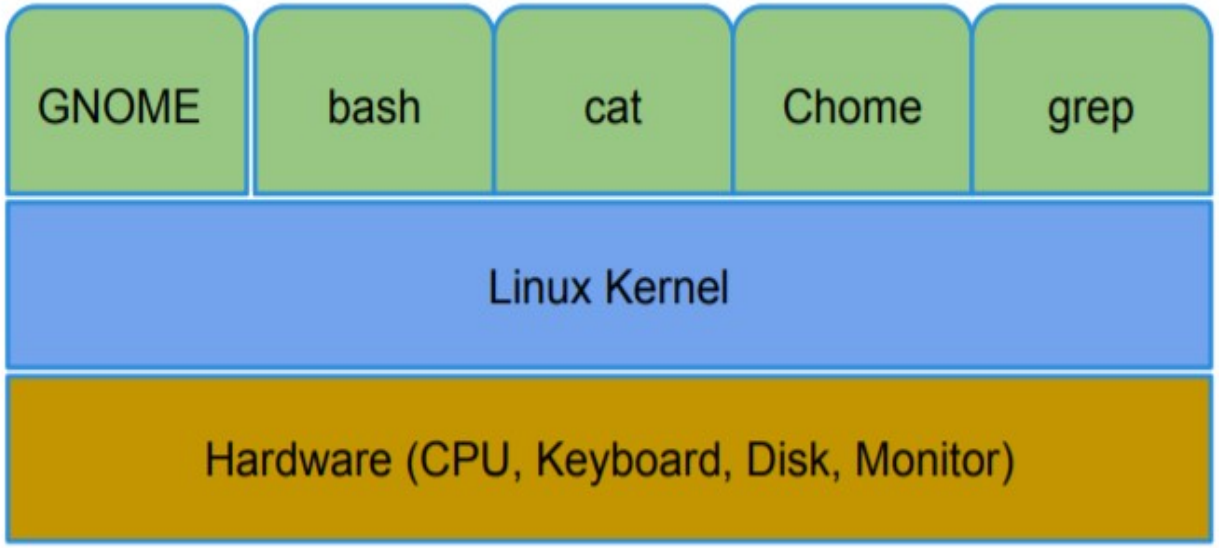
- A fully-networked 32/64-Bit Unix-like Operating System
  - Unix Tools Like sed, awk, and grep (explained later)
  - Compilers Like C, C++, Fortran, Smalltalk, Ada
  - Network Tools Like telnet, ftp, ping, traceroute
- Multi-user, Multitasking, Multiprocessor
- Has the X Windows GUI
- Coexists with other Operating Systems
- Runs on multiple platforms
- Includes the Source Code

# Component of Linux

- The Linux Kernel
- Libraries
- Utilities
- User Interface

# Linux Kernel

Very simple answer: it's a program that makes your hardware look and feel like an OS to other programs



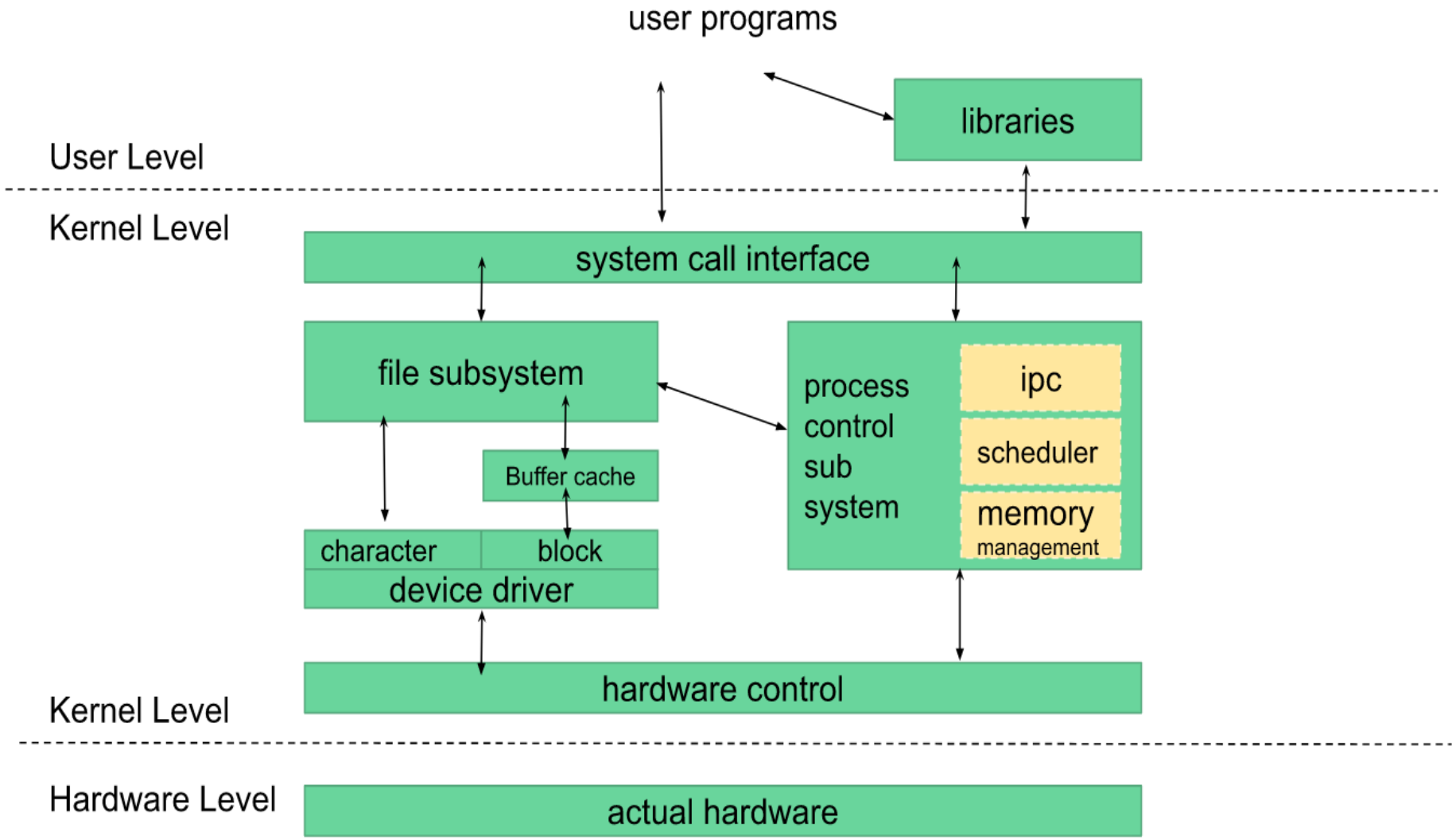
# Linux Kernel

- **Libraries** are pre-written code “pieces” that application programmers use in their programs.
- **Utilities** maintaining the file system, editing text files, managing running processes, and installing new software packages.
- **User Interface** command-line interface (CLI) and a graphical user interface (GUI).

# Kernel Space and User Space

- The Concept of kernel space and user space is all about memory and access rights.
- It is a feature of modern CPU, allowing it to operate either in privileged or unprivileged mode.
- One ,may consider kernel to be privileged and user apps are restricted.

# Kernel Space and User Space

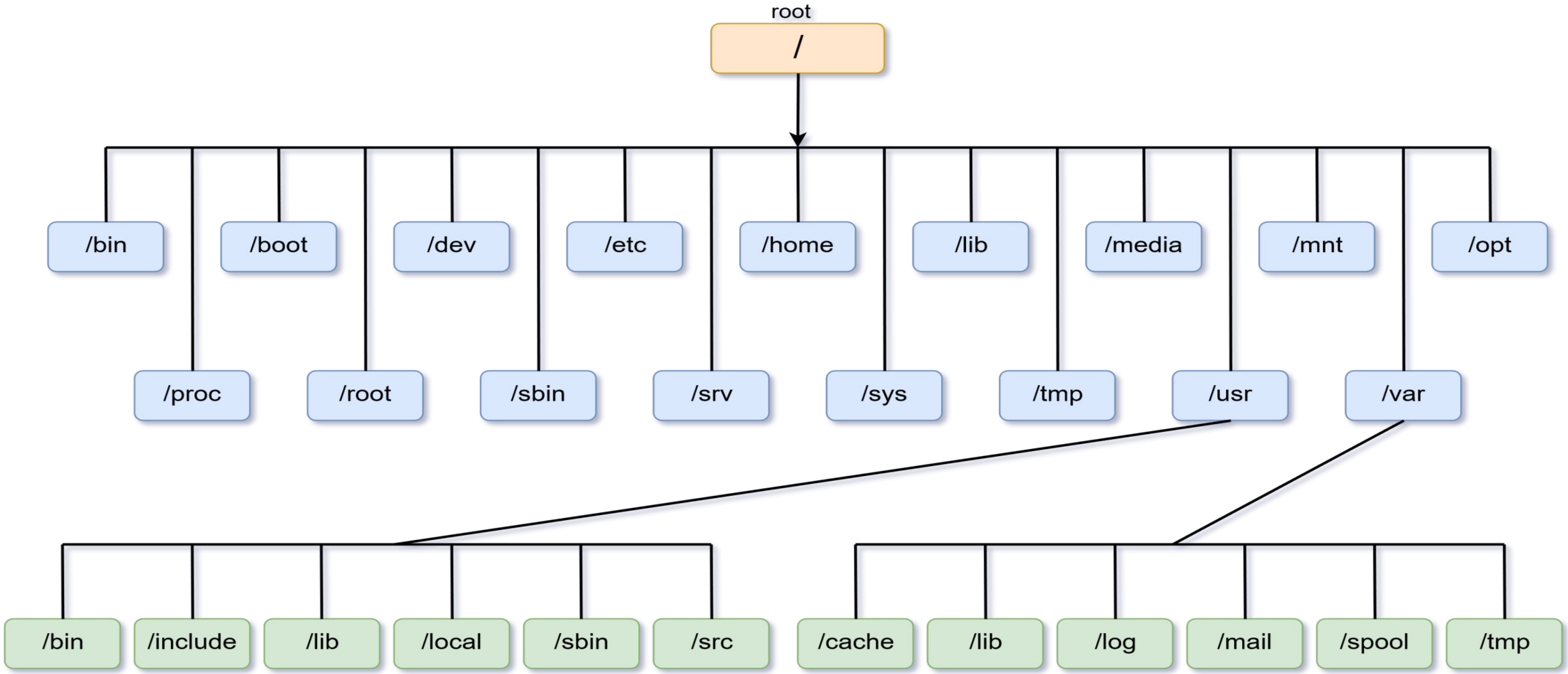



Block Diagram of the System Kernel

# Linux Filesystem Hierarchy

- The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.
  - In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.
  - Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.
  - Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.

# Linux Filesystem Hierarchy



# Linux Filesystem Hierarchy

- **/ (root)**
  - Primary hierarchy root and is the first directory and the root directory of the entire file system hierarchy.
  - It contains all other directories i.e. the sub directories.
  - Only the root user has the permissions to write here.
  - This is not the home directory for the root user
- **/bin (user binaries)**
  - It contains binary executables.
  - Binary executables of common linux commands used by all users in single-user mode are located in this directory.
  - Some files present in this directory are: ls, cp, grep, ping, cat, etc.

# Linux Filesystem Hierarchy

- **/boot (boot loader files)**
  - It contains boot loader files.
  - kernel, initrd, grub, and other files and directories are located in this directory.
  - e.g. vmlinux-2.7.31.25-generic, initrd.img-2.3.32.24-generic, etc.
- **/dev (device files)**
  - It contains the essential files related to the devices attached to the system.
  - This includes terminal devices, USB, network devices and any other devices that are attached to the system.
  - e.g. /dev/usbmon0 , /dev/tty1 , /dev/null , etc.

# Linux Filesystem Hierarchy

- **/etc (configuration files)**

- etc stands for ‘edit to config’
- This directory contains the configuration files that are required by the installed programs.
- The files are host-specific and system-wide configurations needed for the proper functioning of the system.
- This directory also contains shell scripts for system startup and system shutdown that are used to start or stop individual programs.
- The files in this directory should not be edited without proper knowledge of system configuration as improper configuration could brick the system.
- e.g. /etc/passwd , /etc/shadow , /etc/group , /etc/resolv.conf , etc.

# Linux Filesystem Hierarchy

- **/home (home directories)**
  - This directory contains user's home directories, containing saved files and personal settings.
  - Each user will have an separate directory with their username under this directory except the root user because every time a new user is created, a directory is created in the name of the user within the home directory.
  - e.g. /home/user , /home/sage , /home/guest , etc.
- **/lib (system libraries)**
  - This directory contains libraries that are essential for the binaries in /bin and /sbin
  - Library filenames are either ld* or lib*.so.*
  - e.g. ld-2.11.1.so , etc.
- **/media (removable media devices)**
  - Temporary mount directory for removable media such as CD-ROM.
  - e.g. /media/cdrom for CD-ROM ; /media/floppy for floppy drives ; /media/cdrecorder for CD writer ; etc.

# Linux Filesystem Hierarchy

- **/mnt (mount directory)**
  - Temporary mount directory where system administrator can mount file systems.
- **/opt (optional application software packages)**
  - This directory contains add-on applications from individual vendors.
- **/proc (process information)**
  - This is a virtual filesystem providing process and kernel information. This files in this directory are automatically generated, populated and deleted by the system. In Linux, corresponds to a procfs mount.
  - This directory contains information about the processes running in the system.
  - This directory also contains text information about running processes. e.g. /proc/uptime
  - e.g. /proc/{pid} directory contains information about the process with that particular pid that will be mentioned within the brackets.

# Linux Filesystem Hierarchy

- **/root (root directory)**
  - This is the home directory for the root user.
- **/sbin (system binaries)**
  - This directory contains essential system binaries.
  - The linux commands that are located in this directory are used by system administrator, for system maintenance and configuration purpose.

  e.g. fsck , reboot , fdisk , ifconfig , init , etc.

# Linux Filesystem Hierarchy

- **/root (root directory)**
  - This is the home directory for the root user.
- **/sbin (system binaries)**
  - This directory contains essential system binaries.
  - The linux commands that are located in this directory are used by system administrator, for system maintenance and configuration purpose.

  e.g. fsck , reboot , fdisk , ifconfig , init , etc.
- **/srv (service data)**
  - This directory contains site-specific data served by the system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems i.e. server specific services related data.

  e.g. /srv/cvs contains CVS related data , etc.

# Linux Filesystem Hierarchy

- **/sys (system)**
  - This directory contains information about devices, drivers, and some kernel features.
- **/tmp (temporary files)**
  - This directory contains temporary files created by system and the users that will be rebooted when the system is rebooted.
- **/usr (user programs)**
  - This directory contains read-only user data like binaries, libraries, documentation and source-code for second level programs like user utilities and applications.
  - /usr/bin → contains binary files for user programs. If you can't find a user binary under /bin, then we should look under /usr/bin.
  - /usr/include → contains standard include files.

# Linux Filesystem Hierarchy

- /usr/lib → contains libraries for the binaries in /usr/bin and /usr/sbin

- /usr/local → tertiary hierarchy for local data. contains users programs that you install from source. e.g., when you install apache, it goes under /usr/local/apache2

- /usr/sbin → /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, then you should look under /usr/sbin. It also contains non-essential system binaries. e.g. daemons for network-services.

❑ **/var (variable files)**

- This directory contains files whose content is expected to continually change during normal operation of the system—such as logs, spool files, and temporary e-mail file.
- /var/log → contains system log files.

# Thank you