MA112

Devendra Ghate

23/11/2020

Introduction to Linux

- Linus Torvalds released in 1991
- Comes in many flavours
- Runs on most supercomputers
- Very flexible and open-source
- Complete control with the user (e.g., linux runlevels)

Linux Shell

- Way to interact with Linux
- `bash`, `zsh`, `sh`, `ksh`
- Preferable to other methods since repeated tasks can be automated.

Linux Commands

- `pwd`, `ls`, `whoami`, `man`, `cd`, `cp`, `mv`, `mkdir`
- Text editors: `vim`, `emacs`, `nano`, `gedit`, `sublime`
- Browser: `firefox`
- IDEs (Integrated Development Environment): `codeBlocks`, `atom`, `eclipse`

Introduction

- Any machine that does a computation is a computer.
- Fixed program computer (Calculator) Vs. stored program computer

Introduction

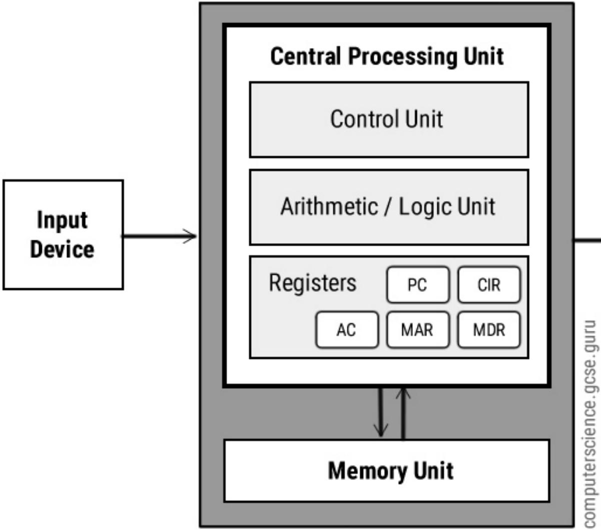
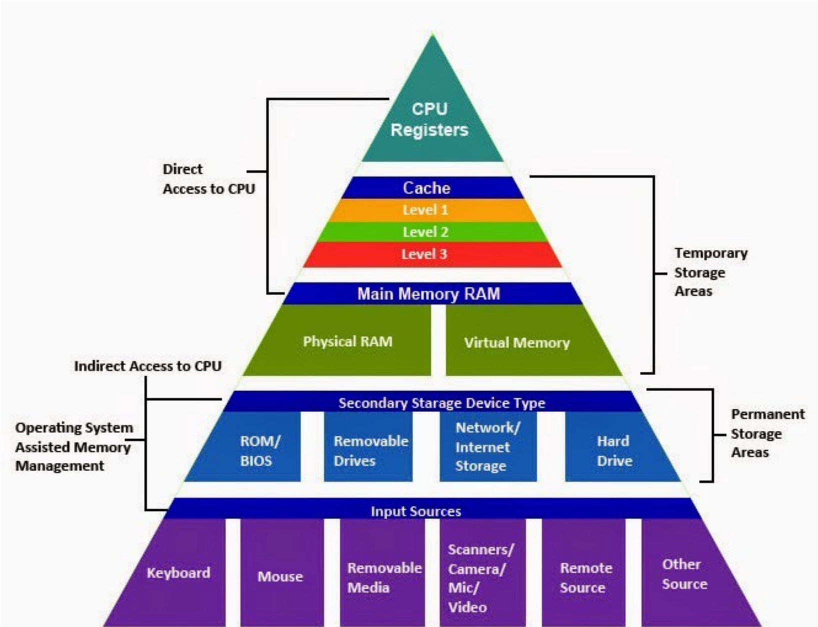


Figure 1: Von Neumann Architecture

Introduction



Turing Complete Machine

Alan Turing showed that any operation can be performed using six primitive instructions

1. Move left
2. Move right
3. Read
4. Write
5. Erase
6. Nothing/Halt

Keep in mind, he was working with tapes rather than the current rotating hard disks.

Computer Programming

Writing instructions at this abstraction level is hard. Hence, people invented various programming languages to issue instruction sets to computer.

Programming Language

- Similar to human languages. However, only *one-way* traffic. *Computer can read any program. Goal of programming is to write a human-readable program.*

We will learn,

- General programming building blocks (applicable to all languages)
- Syntax for one language (`C++`)

Programming Language Classification

- Higher-level Vs. Lower-level
- Application specific Vs. General Purpose

Programming Language Examples

- BASIC, FORTRAN, C, C++ (Lower level, high performance - Scientific Computing)
- Python, Matlab, Julia, R (Higher level language - Scientific Computing)
- Shell, PHP, perl, java, javascript (Internet)
- Lua, Go (Special purpose)
- Haskell (Functional Programming)

What exactly are we going to learn?

- Declarative knowledge (**What**) ($\sqrt{4} = 2$)
- Imperative knowledge (**How**)

Algorithm to calculate root of *n*

```
1. Start with an arbitrary positive start value x
   (the closer to the root, the better).
2. Initialize y = 1.
3. Do following until desired approximation is achieved.
  a. Get the next approximation for root using
     average of x and y
  b. Set y = n/x
```

```cpp
#include <iostream>
using namespace std;
class gfg {
    /*Returns the square root of n.
      Note that the function */
public:
    float squareRoot(float n)
    { /*We are using n itself as initial approximation
        This can definitely be improved */
        float x = n;
        float y = 1;
        float e = 0.000001; /* e decides the accuracy*/
        while (x - y > e) {
            x = (x + y) / 2;
            y = n / x;
        }
        return x;
    }
};
```

```cpp
/* Driver program to test above function*/
int main()
{
    gfg g;
    int n = 50;
    cout << "Square root of " << n <<
        " is " << g.squareRoot(n);
    getchar();
}
```