# Introduction to Technical Programming

1.1 Different types of hardware of a common system

**Content:**

• System Unit

• Motherboard

• CPU

• Memory

• Input and Output hardware

Learning Outcomes:

Students should be able to:

1.1.1 Identify the components of the system unit

1.1.2 Explain the term CPU and its purpose

1.1.3 Describe the term CPU and explain the impact of using various types of CPU’s

1.1.4 Describe different types of CPUs regarding use, and power

1.1.5 Define the term computer memory

1.1.6 Discuss the **primary purpose of memory**

**1.1.7 Differentiate between different types of memory and their purpose (Range: RAM, ROM, CMOS, Cache memory, Flash Memory)**

**1.1.8 Explain how data is stored** on memory

1.1.9 Define the purpose of the motherboard and its components

1.1.10 Describe different types of input hardware regarding use and classification e.g., direct, and indirect entry

1.1.11 Discuss how data is transferred between memory i.e., primary, and secondary and the CPU

1.1.12 Describe the Flow/transfer of data between components. (Range: USB – PnP, U3, Point-to- point connections)

* + 1. Describe the factors to consider when choosing an input device. (Range: Ergonomic considerations, Wireless vs cables)
    2. Describe different types of output hardware
    3. Describe the purpose and use of devices such as docking stations for mobile and laptop computers.

1.2 **Purpose of software**

Content:

* Software as a component of a computerised system
* Categories of software

Learning Outcomes:

*Students should be able to:*

* + 1. Describe the term software
    2. Describe the purpose and function of software
    3. Describe the basic concepts of software. (Range: Software as programs. Identify software components. Concept of a graphical user interface (GUI).)
    4. Contrast: System software vs application software
    5. Differentiate between: Shareware, Freeware, Open-Source Software and Proprietary software, Firmware
    6. Discuss the process of how software is obtained and installed.
    7. Differentiate between online software and installed software.
    8. Discuss the following terms in relation to software. (Range: Compatibility issues, Versions, patches, and service packs, Updating software)

1.3 **The Linux shell**

Content:

• Working with the Linux Bash terminal

Learning Outcomes:

Students should be able to:

* + 1. Launch a new Linux terminal on the Raspberry Pi
    2. Use the man command to get help
    3. Expand a Linux file path and explain each element
    4. List the contents of the current folder using the ls command
    5. Change directly location using the cd command
    6. Create a new folder using the mkdir command
    7. Remove a folder using the rmdir command
    8. Remove a file using the rm command
    9. Rename a file using the mv command
    10. Copy a file using the cp command
    11. Clear the command prompts screen using the cls command
    12. Run an executable file from the command line

**Problem solving in computer programming**

2.1 **Problem solving process and concepts**

Content:

* Problem solving
* Application of problem-solving constructs
* Developing solutions

*Learning Outcomes:*

*Students should be able to:*

2.1.1 Define the term problem solving

2.1.2 Define the term computational thinking

2.1.3 Describe the phases of the PLDC (Program Development Life Cycle)

2.1.4 Describe the purpose of problem solving leading to solutions

2.1.5 Explain and apply various problem-solving steps. Polya, G., 1957) (Range:Understand the problem (task/problem description or scenario/user stories) State in own words Clarity on what needs to be done

What is known or given?

What is missing or needed?

Devise a plan/algorithm (storyboard – visual or textual)

Look for patterns

Look at related problems, known solutions

Examine simpler or special cases

Make a table, create diagram, use guess and check, work backwards, identify sub-goal

Carry out the plan/implement the algorithm (write the code)

Look back/test (see if it works)

Check results against original problem. Does it make sense? Is there another solution?)

2.1.6 Use appropriate tools and techniques to present a solution. Range:

User stories (written by the client and provide the requirements)

Noun-verb analysis of user stories

List of nouns provides identification of objects and state

List of verbs provides identification of behaviour

Acceptance tests (does the program meet the requirements?)

2.2 **Construct an algorithm and present a solution to a given problem**

Content:

* Problem solving
* Algorithm design
* Flowcharts

*Learning Outcomes:*

*Students should be able to:*

2.2.1 Define the term algorithm and its purpose in the problem-solving process. (Range: Basic concepts of an algorithm. What is an algorithm? Develop a clear understanding of the problem presented.)

2.2.2 Implement and understand the basic algorithmic constructs used to create a **flowchart.** Range: Input, Output, Processing and Calculations, Selection Iteration

2.2.3 Create a flowchart to present a particular algorithm and its associated tasks

2.2.4 Interpret a basic flow chart and describe its intended operation / function

* 1. **Building and running C/C++ applications**

Content:

* The C and C++ compiler
* Compiler artefacts
* Compiler design

*Learning* Outcomes*:*

*Students should be able to:*

* + 1. Define *the* term compiler
    2. Define the term source code
    3. Explain what a decompiler is used for
    4. Define the term interpreter
    5. Explain the difference between a compiler and an interpreter
    6. Explain what a binary is and when it is produced
    7. Explain the difference been C and C++ compiler
    8. List and Explain the basic three stage compiler design (Front Middle Back end)
  1. **C/C++ on the Desktop**

Content:

* C/C++ compiler environment on desktop PC
* IDE on desktop PC

*Learning Outcomes:*

*Students should be able to:*

* + 1. Install and configure C/C++ compiler on desktop PC
    2. Define the term IDE
    3. Explain what an IDE is used for
    4. Install and configure IDE on desktop PC

1.1 Different types of hardware of a common system

After you have completed this module, you should be able to :

* Identify the components of the system unit
* Explain the term CPU and its purpose
* Describe the term CPU and explain the impact of using various different types of CPU’s
* Describe different types of CPUs with regard to use, and power
* Define the term computer memory
* Discuss the primary purpose of memory
* Differentiate between different types of memory and their purpose (Range: RAM, ROM, CMOS, Cache memory, Flash Memory)
* Explain how data is stored on memory
* Define the purpose of the motherboard and its components
* Describe different types of input hardware with regard to use and classification e.g. direct and indirect entry
* Discuss how data is transferred between memory i.e. primary and secondary and the CPU
* Describe the Flow/transfer of data between components. (Range: USB – PnP, U3, Point-to-point connections)
* Describe the factors to consider when choosing an input device. (Range: Ergonomic considerations, Wireless vs cables)
* Describe different types of output hardware
* Describe the purpose and use of devices such as docking stations for mobile and laptop computers.
* Describe the term software
* Describe the purpose and function of software
* Describe the basic concepts of software. (Range: Software as programs. Identify software components. Concept of a graphical user interface (GUI).)
* Contrast: System software vs application software
* Differentiate between: Shareware, Freeware, Open Source Software and Proprietary software, Firmware
* Discuss the process of how software is obtained and installed.
* Differentiate between online software and installed software.
* Discuss the following terms in relation to software. (Range: Compatibility issues, Versions, patches and service packs, Updating software)
* Launch a new Linux terminal on the Raspberry Pi
* Use the man command to get help
* Expand a Linux file path and explain each element
* List the contents of the current folder using the ls command
* Change directly location using the cd command
* Create a new folder using the mkdir command
* Remove a folder using the rmdir command
* Remove a file using the rm command
* Rename a file using the mv command
* Copy a file using the cp command
* Clear the command prompts screen using the cls command
* Run an executable file from the command line

# 1.1 Different types of hardware of a common system

### Introduction

Our daily lives have become increasingly reliant on computers for everyday tasks. As a result, computer programming has become an essential skill. Even though not everyone has the skill, those who choose to be programmers must understand how computers work. Knowledge of computer architecture increases the understanding of some programming concepts, not simply knowing them. This book introduces the types of hardware to begin with a basic understanding of how a system functions.

### Identify the components of the system unit

**Vocabulary**

A system unit is the part of a computer that houses the primary devices that perform operations and produce results for complex calculations.

In this section, we will explain what a system unit is and its purpose on a computer. The system unit contains components such as the motherboard, the central processing unit (CPU), random access memory (RAM), and other devices. This unit performs most of the tasks that a computer is required to perform. In general, system units are used to differentiate between the computer itself and its peripheral devices, such as the monitor, keyboard, and mouse. In layman's terms, a system unit is also called a chassis or a tower. Figure 1.1 illustrates some components making up a system unit. Some of the components are too small to show diagrammatically and we will try to explain them in detail.

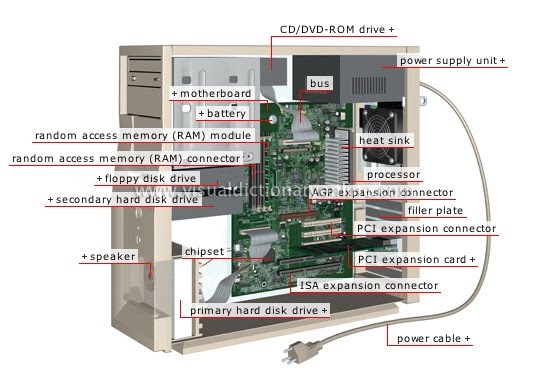


Figure 1. 1: System Unit components

**General functions of components**

Providing proper system functioning is dependent on each component's unique functions. Here are some of the general functions of the main components:

* **Motherboard**. Since connecting all computer nodes to one another is this device's primary function, it is essentially just a series of cables connecting the processor and memory modules.
* **Processor** The computer's "brain," or core, or the element that handles the majority of the work is called the processor. The processor receives data from other parts of the computer and uses it to make computations. The data is returned to the many devices that utilise it after mathematical calculations.
* **RAM** Data storage and quick access are goals of RAM. After the power is cut off, all data on it is lost. Therefore, RAM is referred to being volatile. Volatile memory is a type of memory that maintains its data only while the device is powered. If the power is interrupted for any reason, the data is lost.
* Hard Drive- The hard drive is intended to serve as the computer's permanent storage for files such as music, movies, pictures, and documents.
* **Video Card** converts the image in the computer’s memory into a video signal for the monitor.
* **Power Supply** provides power to the PC. It also performs the functions of stabilization and protection against minor interference in the electrical network.

### 1.1.2 Explain the term CPU and its purpose

**VOCABULARY**

The computer's central processing unit (CPU) is the portion of a computer that retrieves and executes instructions. The CPU is essentially the brain of a computer system.

The acronym CPU stands for Central Processing Unit. The CPU is placed into a specific square-shaped socket found on all motherboards by inserting its metallic connectors or pins found on the underside. Each socket is built with a specific pin layout to support only a specific type of processor.

CPU is the part of a computer that handles the data and activities of the various physical components of the computer. It transfers instructions between a computer's hardware and software. It is also known as a processor, microprocessor, or central processor. Inputs enter a computer and travel to the CPU

### 1.1.3 Describe the term CPU and explain the impact of using various types of CPU’s

The CPU executes the instructions and delivers the results to the associated output. Basically, the CPU is the heart of a computer. It takes in the necessary information and processes it, thus allowing the computer to function. A CPU has three main parts: arithmetic logic unit (ALU), control unit (CU), and memory unit. Based on the von Neumann architecture, the CPU is equalled to the brain of a computer since it controls everything in the computer system. John von Nuemann a Hungarian-American [mathematician](https://en.wikipedia.org/wiki/Mathematician), [physicist](https://en.wikipedia.org/wiki/Physicist), [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist), [engineer](https://en.wikipedia.org/wiki/Engineer). He made great contribution in computer architecture.

**Instruction Processing cycle**

When a program is being carried out, the CPU implements the fetch – decode – execute cycle, which recurs repeatedly until arriving at the STOP instruction. Here are the common five stages of the fetch decode execute cycle.

1. **Fetch- Instruction from Memory (Instruction Fetch, IF)**

Each instruction is stored in memory and has its own address. The processor takes this address number from the program counter, which is responsible for tracking which instructions the CPU should execute next. Once the instruction is fetched, the program counter (PC) will have the address of the next instruction to be executed. A program counter is also known as an instruction counter, instruction pointer, instruction address register or sequence control register.

**VOCABULARY**

A program counter (PC) is a CPU register in the computer processor which has the address of the next instruction to be executed from memory.

1. **Decode the instructions into binary (Instruction Decode, ID).**

All programs to be executed are translated into Assembly instructions. Assembly code must be decoded into binary instructions, which are understandable to your CPU. This step is called decoding.

1. **Execute action and move to next step or calculate address (EXE).**

While executing instructions, the CPU can do one of three things: Do calculations with its ALU, move data from one memory location to another, or jump to a different address.

1. **Access memory operand (MEM).**

The CPU must give feedback after executing an instruction, and the output data is written to the memory.

1. **Write back result to register (WB).**

If we need to store the result in the destination location, it is done during the writeback stage, and the register file is updated

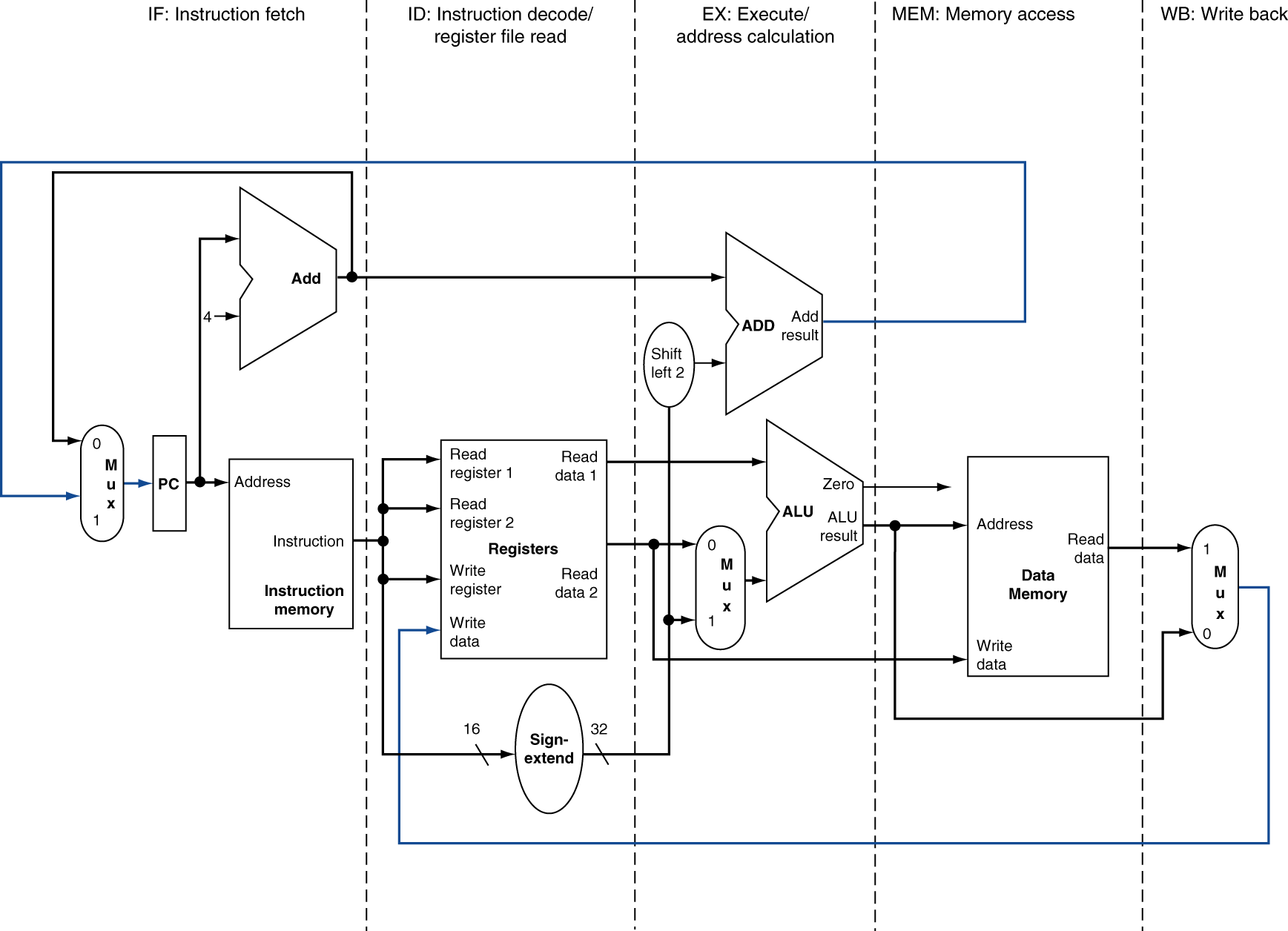


Figure 1.2: 5 stages of processing

Each instruction must be completed in one cycle. Logically, instruction execute one at a time but physically instructions execute in parallel through a technique called pipelining. Pipelining is an implementation technique where multiple instructions get overlapped during execution. Pipelining enhances computer throughput.

A CPU's speed determines how many operations it can carry out. The rate at which one operation is completed in a second is measured in hertz. A computer's speed is typically expressed in gigahertz. The CPU can complete one million simple jobs at 1 GHz speed. The simplest actions a processor can take are considered "simple tasks."

The more instructions your CPU can process in a second, the faster it is, but do not be deceived by this figure. The performance of your computer is impacted by a variety of factors, not only the CPU's speed. To obtain independent results, numerous additional variables, like CPU architecture, cache capacity, and bus speed, must be considered.

The central processing unit (CPU) has three components:

* Control Unit
* Arithmetic Logic Unit
* Registers

**Control Unit**

The control unit reads instructions from memory, decodes them, and then puts them into action. The control unit serves as a middleman, decoding the instructions provided to the processor, directing the other units, such as the Arithmetic Logic Unit (below), using control signals, and sending the processed data back to memory.

**Arithmetic Logic Unit (ALU)**

Digital circuitry inside the processor called an arithmetic logic unit (ALU) handles arithmetic and logical operations by loading data from input registers. The ALU completes the required operations by connecting many transistors after receiving the necessary instructions from the control unit, and then it records the outcomes in an output register. The system clock, memory, secondary storage, and data and address buses are all necessary for the CPU to operate effectively. Some of the arithmetic operations include addition subtraction, multiplications, and divisions. The result of arithmetic computations is put into the accumulator.

**Register** – saves the most frequently used instructions and data.  Processor registers are a common name for the registers used by the CPU. An instruction, a storage address, or any other data, including bit sequences or single characters, may be stored in a processor register.

There are 6 types of central processing units Single Core, Dual Core, Quad Core, Hexa Core , Octa Core, and Deca Core.

Single Core CPU- consists of a single CPU and can execute one instruction at a time.- An example is the Intel 4004 released in 1971.

Dual Core CPU- Consists of two cpu’s that act like one CPU meaning the CPU can multitask e.g Pentium D released in 2005.

Quad-Core CPU-A quad-core CPU has **four cores on a single CPU processor**. The CPU is the greatest choice for multitasking since it equally distributes the workload among its cores. An example of quad core CPU is Athlon II X4 which was released in 2009.

Some computers utilize two or more processors. These consist of separate physical microprocessors located side by side on the same board or on separate boards. Each CPU has an independent interface, separate cache, and individual paths to the system front-side bus. As you will notice from Figure 1.3, the name depicts the number of cpus e.g. Hexa core has 6, octa core has 8 and deca core has 10 cpu’s.



Figure 1. 3: Types of CPU's

Intel and AMD are the two key manufacturers of CPU’s.  These manufacturers are both great, though they tend to be used in different situations.

### Describe different types of CPUs with regard to use, and power

Section 1.1.3 outlined the types of processors based on number of cores. Processors can also be categorised based on:

* Data bus width e.g., 32-bit and 64-bit processors
* Brand name and manufactures e.g., Intel and AMD
* Architecture- e.g., Reduced Instruction Set Computing RISC and Complex Instruction Set Computing (CISC)
* General purpose processors based on architecture- in this category we have 5 types which are explained below

**Key Difference between RISC and CISC processor**

* In RISC, the instruction set is reduced, and most of these instructions are very primitive, while in CISC, the instruction set is very large that can be used for complex operations.
* RISC computer’s execution time is very less, whereas CISC computer’s execution time is very high.
* RISC code expansion may create a problem, while CISC code expansion is not a problem.
* In RISC, the decoding of instructions is simple, whereas, in CISC, the decoding of instructions is complex.
* RISC doesn’t require external memory for calculations, but CISC requires external memory for calculations.
* RISC has multiple registers sets present, while CISC has only a single register set.

There are five types of general-purpose processors they are, Microcontroller, Microprocessor, Embedded Processor, DSP and Media Processor.

Microcontroller

A microcontroller is basically a type of computer that comes in different packages and sizes. Feedback on reading input and output is the main function of this Microcontroller. It is also commonly referred to as General Purpose Input Output (GPIO).



Figure 1. 4: Example of a microcontroller (  
PIC32MX Series Microcontrollers)

Microprocessor

The general-purpose processors are represented by the microprocessor in embedded systems. There are different varieties of microprocessors available in the market from different companies. The microprocessor is also a general-purpose processor that consists of a control unit, ALU, a bunch of registers also called scratchpad registers, control registers and status registers.



Figure 1.5:Example of Microprocessor

Microprocessors are cheap, offer high speed, small in size, consumes less power and generate less heat.

Microprocessors uses a clock signal to control the rate at which instructions are executed, synchronize other internal components and to control the data transfer between them.

**Embedded Processor**

An embedded processor is one type of processor which is designed to control mechanical functions and electrical functions. It consists of several blocks such as the processor, timer, an interrupt controller, program memory and data memory, power supply, reset and clock oscillator circuits, system application-specific circuits, ports and interfacing circuits. Embedded processors are designed especially for coping with the wishes of an embedded gadget.

Embedded processors may be observed in transportable devices like virtual watches, PDAS, digital cameras, GPS gadgets and MP3 players. They also can be located in large structures such as visitor’s lights, systems controlling energy vegetation and factory controllers.

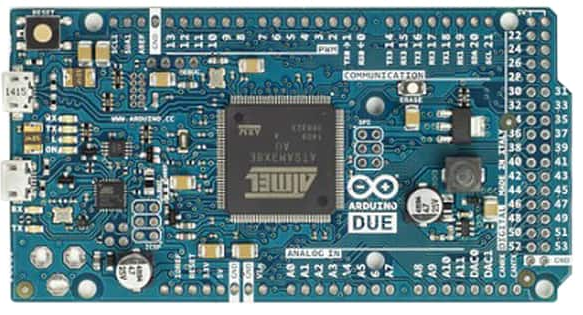


Figure 1. 6:Embedded processors

Digital Signal Processor

The digital signal processor is one type of processor used for measuring, filtering and/or compress digital or analog signals. Digital Signal Processors (DSP) take real-world signals like voice, audio, video, temperature, pressure, or position that have been digitized and then mathematically manipulate them. A DSP is designed for performing mathematical functions like "add", "subtract", "multiply" and "divide" very quickly.

The signal processing means analysis and manipulation of signal and can be done via computer or Application Specific Integrated Circuits (ASIC), Field Programmable Gate Array (FPGA) or Digital Signal Processor (DSP) to obtain the clear signal. The DSP processors are used in an oscilloscope, barcode scanners, mobile phones, printers, etc.

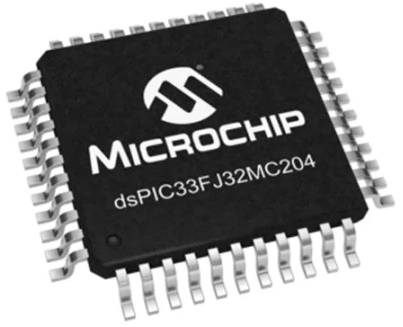


Figure 1. 7: Example of 16bit Digital Signal Processor 40MIPS

### Define the term computer memory

VOCABULORY

Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.

The memory is divided into large number of small parts called cells. Each location or cell has a unique address, which varies from zero to memory size minus one. For example, if the computer has 64k words, then this memory unit has 64 \* 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535. The CPU is responsible for selecting memory cells to read or write data.

The concept of memory and storage can be easily conflated as the same concept; however, there are some distinct and important differences. Put succinctly, memory is primary memory, while storage is secondary memory. We will explain these terms in detail in section 1.1.7.

### 1.1.6 Primary purpose of memory

Memory is central to a computer's operation because it forms the critical link between software and the CPU. The basic function of computer memory is essentially to store data. Depending on the type of data it stores and the role it plays in computer operation, however, memory performs several different functions. Although all of these functions involve data storage, RAM, ROM, flash memory and hard drives each perform a different and necessary function to keep a computer and its peripherals working. Computer memory likewise decides the size and number of programs that can be run all the while, and helps to optimize the capabilities of capable microprocessors

Once the data stored in computer memory, the data will remain there forever. Each time memory is full, then the data can be deleted in part or in whole to be replaced with new data. The performance of a computer depends on memory and CPU. CPU cannot store programs or a large set of data permanently. They are only capable of storing basic instructions required to operate the computer. Therefore, it is mandatory to have the memory to run a computer system properly.

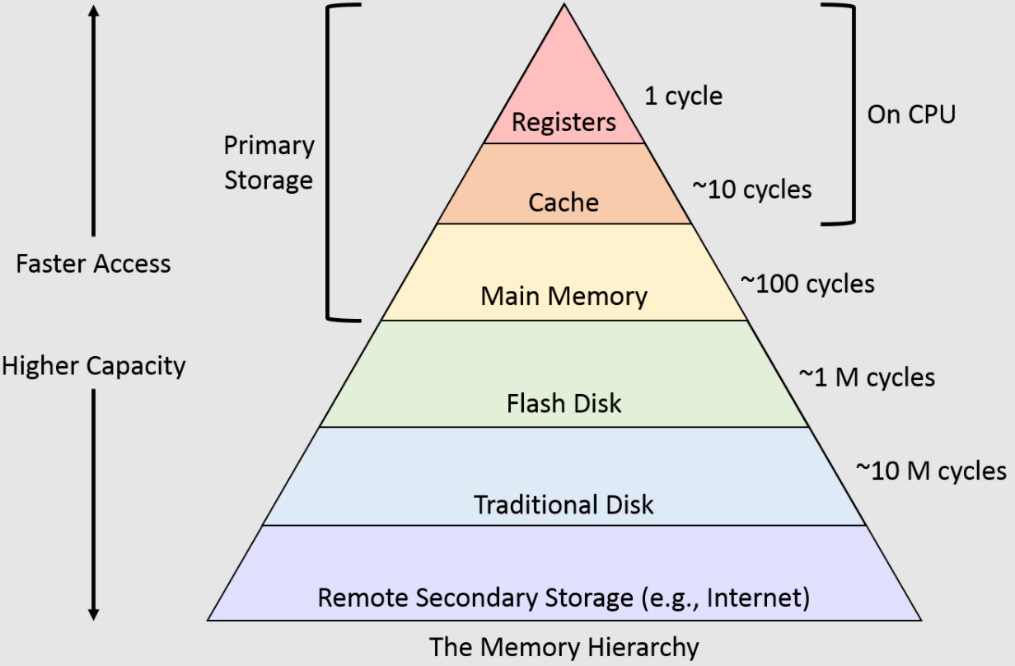
### 1.1.7 The different types of memory and their purpose (Range: RAM, ROM, CMOS, Cache memory, Flash Memory)

There are basically 2 broad categories of memory.



Figure 1. 8:Categories of Computer Memory

To further assist us in understanding computer memory, we will use a computer hierarchy chart and then explain each of these types.



In computer architecture, the memory hierarchy is an improvement of computer storage into a hierarchy based modal on response time. It affects performance in a computer algorithm, predictions, architectural design, and lower-level programming constructs involving the locality of reference.

**Registers**

Register memory is the smallest and fastest memory in a computer. A register temporarily holds frequently used data, instructions, and memory address that are to be used by CPU. They hold instructions that are currently processed by the CPU. All data is required to pass through registers before it can be processed.  Some of the widely used registers include Accumulator or AC, Data Register or DR, the Address Register or AR, Program Counter (PC), I/O Address Register, and these are explained in Table 1.1.

|  |  |
| --- | --- |
| **Type of register** | **Uses** |
| Data register | stores data, which is being transmitted to or received from a peripheral device. |
| Program Counter | It holds the address of the memory location of the next instruction to be fetched. |
| **Instructor Register** | hold instruction codes and passes them to the Control Unit to be decoded. |
| **Accumulator Register** | is a type of register for short-term, intermediate storage of arithmetic and logic data in a computer's central processing unit (CPU). |
| Address Register | it is a 12-bit register that stores the address of a memory location where instructions or data is stored in the memory. |

Table 1. 1: Common registers

**Cache Memory**

Cache memory is a high-speed memory, which is small but faster than the main memory (RAM). The CPU can access it more quickly than the primary memory. So, it is used to synchronize with high-speed CPU and to improve its performance.

When CPU needs the data, first, it looks inside the L1 cache. If it does not find anything in L1, it looks inside the L2 cache. If again, it does not find the data in L2 cache, it looks into the L3 cache. If data is found in the cache memory, then it is known as a cache hit. On the contrary, if data is not found inside the cache, it is called a cache miss or miss penalty. The solution to reducing cache miss is through implementing several or multilevel caches. **Multilevel Caches** is one of the techniques to improve cache performance by reducing the *“MISS PENALTY”*.

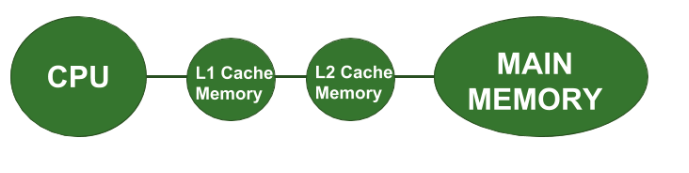


Figure 1. 9: Multilevel caches

**Primary Memory**

Primary Memory is of two types: RAM and ROM.

**Random Access Memory (RAM)**

It is a volatile memory meaning it does not store data or instructions permanently. When you switch on the computer the data and instructions from the hard disk are stored in RAM.

**Advantages of RAM**

1. It is a faster type of memory in a computer.
2. It requires less power to operate.
3. Program loads much faster
4. More RAM increases the performance of a system and can multitask.
5. Perform read and write operations.
6. The processor can read information faster than a hard disc, floppy, USB, etc.

**Disadvantages of RAM**

1. Less RAM reduces the speed and performance of a computer.
2. Due to volatile, it requires electricity to preserve the data.
3. It is expensive than ROM
4. It is unreliable as compared to ROM
5. The Size of RAM is limited.

There are two types of RAM : Static RAM SRAM) and Dynamic RAM (DRAM)

|  |  |  |
| --- | --- | --- |
| parameter | SRAM | DRAM |
| Read & Write | Faster | Slower than SRAM |
| Storage component | Uses transistor to store single bit of information | Uses separate capacitor to store each bit of data |
| Price | Expensive than DRAM | Cheaper than SRAM |
| Power consumption | More | Less |
| Uses | Cache memory | Main memory |
| Density | Less dense | Highly dense |
| Storage per bit | Can store many bits per chip | Cannot store many bits per chip |

**Read-Only Memory (ROM)**

ROM is a memory device or storage medium that is used to **permanently** store information inside a chip. It is a read-only memory that can only read stored information, data or programs, but we cannot write or modify anything. A ROM contains some important instructions or program data that are required to start or boot a computer. It is a **non-volatile** memory.

There are five types of Read Only Memory:

**MROM (Masked Read Only Memory):-** program or data is pre-configured by the integrated circuit manufacture at the time of manufacturing. Its permanent.

**PROM (Programmable Read Only Memory):** It is a type of digital read-only memory, in which the user can write any type of information or program only once.

**EPROM (Erasable and Programmable Read Only Memory)-** stored data can be erased and re-programmed only once in the EPROM memory.

**EEPROM (Electrically Erasable and Programmable Read Only Memory)-** **A high voltage electrical charge is used to erase recorded data from EEROM.**

**Flash ROM: data or instructions** can be written or programmed in small units called block or sector.

**Advantages of ROM**

1. It is a non-volatile memory
2. It is static, so it does not require constant refreshing every time.
3. These cannot be changed accidently
4. It is cheaper than RAM.
5. Simple and reliable

**Disadvantages of ROM**

1. Store data cannot be updated or modify except to read the existing data.
2. It is a slower memory than RAM to access the stored data.

**CMOS- complementary metal-oxide semiconductor**

Alternatively referred to as a RTC (real-time clock), NVRAM (non-volatile RAM) or CMOS RAM, CMOS is short for complementary metal-oxide semiconductor. CMOS is an onboard, battery powered semiconductor chip inside computers that stores information. This information ranges from the system time and date to system hardware settings for your computer.

**Which devices use CMOS?**

* [Microprocessors](https://www.computerhope.com/jargon/c/cpu.htm)
* Microcontrollers
* [Digital logic circuits](https://www.computerhope.com/jargon/l/logicirc.htm)
* [SRAM](https://www.computerhope.com/jargon/s/sram.htm) (Static RAM)

**Secondary Memory**

The secondary storage devices which are built into the computer or connected to the computer are known as a secondary memory of the computer. It is also known as external memory or auxiliary storage. The secondary memory is accessed indirectly via input/output operations. It is non-volatile and can not be accessed directly by the CPU.

**Examples of secondary memory**

1. Hard Disk Drives
2. Solid State Drive
3. Pen Drive/USB flash drive
4. Secure Digital Card -SD Cards
5. CD/ DVDs

**Cloud Storage**

A service paradigm known as "cloud storage" involves sending and storing data on remote storage systems, where it is then maintained, managed, backed up, and made accessible to users across a network, most often the internet. Users often pay a monthly, per-consumption fee for the storage of their cloud data. Some of the providers are AWS.

### 1.1.8 Explain how data is stored on memory

Data is first transformed into straightforward numbers (Binary Digits-Bits) that a computer can easily store. Bit is the basic unit of memory. At a time, it can be either on or off. Generally, bits are represented using electrical voltage. Voltage presence indicates that the bit is in ON state. Voltage absence indicates that the bit is in OFF state. Here, OFF state is considered as 0. ON state is considered as 1. Computer memory is the collection of several bits. Group of 8 bits are called byte. Second, circuitry within the computer records the numbers. Third, software or programs are used to organize, move, and manipulate the numbers.

**Memory Units**

The smallest unit is a bit

I Byte=8 bits

1 Kilobyte=1024 bytes

1 Megabyte=1024 Kilobytes

1 GigaByte=1024 MegaBytes

1 TeraByte=1024 GigaBytes

### 1.1.9 Define the purpose of the motherboard and its components

A motherboard, often referred to as an mboard, mainboard, base board, system board, planar board, or main circuit board, is the main board and the building block of a computer. A logic board is what it is known as on Apple computers. The motherboard serves as a single platform to connect all of the parts of a computer together. It connects the CPU, memory, hard drives, optical drives, video card, sound card, and other ports and expansion cards directly or via cables. The motherboard of the computer houses the CPU, RAM expansion slots, ROM, USB ports, and PCI slots. It enables communication between the RAM, CPU, and every other piece of hardware. Figure 1.10 shows an example of a motherboard.



Figure 1. 10: i7 Gigabyte Q87M Motherboard

Apart from the other components- the motherboard has a Northbridge and Southbridge

**Northbridge:** An integrated circuit in the motherboard's chipset is in charge of establishing a connection between the AGP, CPU interface, and memory. It has direct connections to the CPU interface, AGP, and memory, unlike southbridge. Northbridge's main responsibility is to provide bus-based communication between the CPU and external devices.

**Southbridge:** This motherboard component is an integrated circuit with a single purpose for which it was produced. I/O controllers, hard drive controllers, and integrated hardware all depend on it.

Some of the popular manufacturers of motherboards includes Intel, ASUS, ABIT, BioStar, Gigabyte and MSI.

**Functions of the Motherboard**

**The functions of a computer motherboard are as follows:**

1. The motherboard acts as the central backbone of a computer on which other modular parts are installed such as the CPU, RAM and hard disks.
2. The motherboard also acts as the platform on which various expansion slots are available to install other devices / interfaces.
3. The motherboard is also responsible to distribute power to the various components of the computer.
4. They are also used in the coordination of the various devices in the computer and maintain an interface among them.

Below are the six different types of Motherboards:

* **AT & Baby AT**
* **Advanced Technology eXtended ATX**
* **BTX (Balanced Technology Extended):**
* **DTX Discontinuation Transmission**
* **LPX (Low Profile eXtension)**
* **microATX**
* **NLX-  New Low Profile Extended**

### 1.1.10 Describe different types of input hardware with regard to use and classification e.g. direct and indirect entry

Hardware- Hardware (sometimes abbreviated to HW) can be defined as the physical components that a computer system needs to function.

Software-   consists of written, machine-readable instructions or [programs](https://www.techtarget.com/searchsoftwarequality/definition/program) that tell physical components what to do and when to execute the instructions.

Hardware can be split into input devices, processing devices, output devices and storage devices.

Input devices are used to get data and instructions into the computer. Processing devices are used in the processing of data once it gets into the computer. Input devices can further be split into direct and indirect input hardware.

Direct data entry devices are specific purpose devices designed to automate or speed up the entry of data into the system by minimising human data entry. They have a wide range of uses including in education, retail and in business. They consist of either specialist hardware, software (or both) and come in a number of different forms. Examples includes:

Magnetic stripe reader-– used to read data from magnetic stripes on mostly banking cards, membership cards or hotel door cards. The stripe on the cards holds data such as membership information.



Figure 1. 11: Magnetic Stripe Reader

Chip readers-read data from the chip on bank cards or shopping cards. The chip and pin reader works by inserting the card into a slot and then entering a PIN (personal identification number)

A picture containing person, indoor, remote, game

Description automatically generated

Figure 1. 12: Chip reader

PIN pads- use to enter data into Automated Teller Machines (ATM), EFTPOS system, entry doors and handheld devices.

Optical mark reader- used to read and input information from a form made in pen or pencil. Typically used to read multiple choice questions. The OMR shines on to the form and less light is reflected where a pencil mark has been made.



Figure 1. 13: Optical Mark Reader

Barcode reader- used to scan codes directly from the products, books and membership cards. Figure 1.14 shows a picture of a Barcode Reader.



Figure 1. 14: Barcode Reader

If the data is in human readable form, it must be converted into machine readable form so that a computer can process it. This process of data conversion is time consuming and error prone that causes a major bottleneck in data processing. Some examples of indirect input devices are: keyboard, mouse and joystick. When you press any key on keyboard, it converts that character into series of electronic pulses and sends to CPU.

**Exercise**

Categorise the following devices into direct and indirect input hardware.

* Mouse
* Joy Stick
* Light pen
* Track Ball
* Scanner
* Graphic Tablet
* Microphone
* Magnetic Ink Card Reader(MICR)
* Optical Character Reader(OCR)
* Bar Code Reader
* Optical Mark Reader(OMR)

### 1.1.11 How data is transferred between memory and the CPU

The connections between the CPU and Memory are shown in Figure 1.15.

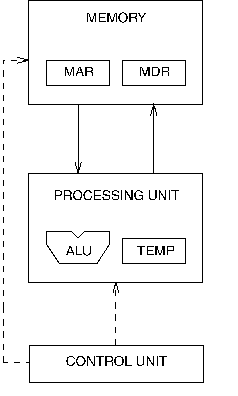


Figure 1. 15:CPU interaction with memory

Communication between memory and processing unit consists of two registers:

* Memory Address Register (MAR).

The Memory Data Register (MDR) keeps the data which is transferred between the Memory and the CPU.

* Memory Data Register (MDR).

The Memory Address Register (MAR) holds the memory location of data that needs to be accessed.

Reading Data

1. The address of the location is put in MAR.
2. The memory is enabled for a read.
3. The value is put in MDR by the memory.

To write,

* 1. The address of the location is put in MAR.
  2. The data is put in MDR.
  3. The **Write Enable** signal is asserted.
  4. The value in MDR is written to the location specified.
* Buses are the means by which data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory. Address bus Carries the addresses of data (but not the data) between the processor and memory.
* Data bus carries data between the processor, the memory unit, and the input/output devices.
* Control bus carries control signals/commands from the CPU (and status signals from other devices) to control and coordinate all the activities within the computer.

### 1.1.12 Describe the Flow/transfer of data between components. (Range: USB – PnP, U3, Point-to-point connections)

The pursuit of the fastest speed and throughput has been one of the key goals in the development of computer architecture. By utilizing technological potential in the design of the computer components and by giving the computer an appropriate structure and organization, this goal was and is still achieved in two ways. Due to rapid technological advancement in the manufacture of integrated circuits, where component speed and density are continuously rising while costs are falling, the advancement of computer components is quite rapid.

**VOCABULARY**

Data Bus- In computer terminology, a bus is a communication system that allows the transfer of data between components within a computer, or between separate computers.

Computer bus can be in the form of wired cables or electrical wires embedded in the computer motherboard PCB (Printed Circuit Board).

The function of a data bus is to either allow these components to communicate with each other or with the outside world. A data bus can transfer data to and from the memory of a computer, or into or out of the central processing unit (CPU) that acts as the device's "engine." A data bus can also transfer information between two computers. The amount of data that can be transferred by a data bus is referred to as **bandwidth.** If we consider the speed of bus or bit transfer, then one wire or bus transfers millions of bits per second.

Most modern computers today use both parallel and serial buses. • Parallel data buses: o Carry data on many wires simultaneously o Each wire carries one bit of data o Most common parallel buses: Advanced Technology Attachment (ATA), PC card, Small Computer System Interface (SCSI) • Serial data buses: o Has one wire that carries all the bits of data o Most common serial data buses: Universal Serial Bus (USB), FireWire, Serial ATA, Serial Attached SCSI. Figures 16 and 17 illustrates a data bus and a computer bus respectively.



Figure 1. 16: Data Bus



Figure 1. 17: Computer Bus

Apart from data bus, there is also an address bus and a control bus.

* The address bus is used to transfer address bits to the memory.
* The control bus is used to transfer control bits from control units to other components of the computer.

Data transfer from Universal Serial Bus

When the software requires data transfer to occur between itself and the USB, it sends a block of data called an *I/O Request Packet (IRP)* to the appropriate pipe, and the software is later notified when this request is completed successfully or terminated by error.

The actual data is sent across the bus in packets. Each *packet* is a bundle of data along with information concerning the source, destination and length of the data, and also error detection information.

Each packet is made up of a set of components called *fields* including the following, summarised in figure 4 :

Table 1: A typical data packet. numbers represent size of field in bits, unless otherwise indicated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sync (8) | PID(8) | Address | Endpoint | Data (0-123 bytes) |

* An eight bit "*SYNC*" synchronisation field used by inputs to correct their timing for accepting data. Marks the start of a data packet.
* The 8 bit Packet Identifier (*PID*) which uses 4 bits to determine the type, and hence format, of the packet data. The remaining 4 bits are a 1's complement of this, acting as check bits. Part of this field determines which of the four groups (token, data, handshake, and special) that the packet belongs to, and specifies an input, output or setup instruction.
* An *address field* which gives the address of the function on the end of the pipe to be used
* The 4 bit *endpoint field*, giving the appropriate endpoint which sends or receives the packet.
* A *data field* consisting of 0-1023 bytes

### 1.1.13 Describe the factors to consider when choosing an input device. (Range: Ergonomic considerations, Wireless vs cables)

Input devices are peripherals from which computers receive data. Keyboards, mice, scanners, and webcams are some examples. If you select the right accessories, you can extend the capabilities of your computer and increase productivity. Make the most of your next computer accessory purchase by understanding the factors to consider when selecting input devices. Here are some of the factors to consider:

1. **User Needs**- this is the urgency of use of the device in the computer room by users also whether it will satisfy the needs of the user.
2. **Initial cost-**the amount it can cost when buying/purchasing the devices
3. **Maintenance Cost**-the amount that can be used to maintain the servicing of these devises should be considered.
4. **Mode Of Transmission**- how will you transport them to the computer room should be considered
5. **Compatibility With Available Hardware**- will the devices fit to other devices in the room already. A device that fits your needs but doesn't work with your computer is useless.
6. **User- Friendliness**- whether the devices will be used to solve problems and easy to be used by users
7. **Wireless vs wired connection**-Cables is typically faster than a Wi-Fi connection, and it offers other advantages as well. Cable connection is more secure and stable than Wi-Fi. You can test your computer's speeds on Wi-Fi versus an Ethernet connection easily.

### 1.1.14 Describe different types of output hardware

**VOCABULARY**

Any peripheral that accepts data from a computer and prints, projects, or reproduces it is known as an output device.

Output devices are used to provide results of the computer to the users. Examples includes **Monitor, Printer, Projectors and Plotters**. The output may be audio, video, hard copy – printed paper, etc. Output devices convert the computer data to human understandable form. We give input to the computer using input devices and the computer performs operations on the data and displays the output to the user using the output device.

**Monitor** - A computer’s principal output device is a monitor, often known as a Visual Display Unit (VDU). It displays the processed data like text, images, videos, audios, etc. There are two types of monitor viewing screens:

* Cathode-Ray Tube (CRT)
* Display on a Flat Panel Monitor with a Cathode-Ray Tube (CRT)
* Plasma Monitor

**Printer**- it is an output device that creates a hard copy of the processed data or information. Printers are divided into two categories:

* Impact Printer
* Non-Impact Printers

1. **Impact Printers**- In impact printers, characters are printed on the ribbon, which is then smashed on the paper. Here, the characters are printed on to the paper by striking an ink ribbon against it with a hammer or print head. Impact printers are relatively cheap making them ideal for large scale printing. Most of these are used in cooperates. Common examples of impact printers include Dot matrix printer, Daisy wheel printer, Line printer and Chain printer.
2. **Non-Impact Printers:**These printers print characters without the use of a ribbon. These printers are fast because they print one full page at a time producing quality printout. The common examples are laser and inkjet printers.

**Plotter**

A plotter is a [printer](https://www.techtarget.com/whatis/definition/printer) that interprets commands from a computer to make line drawings on paper with one or more automated pens. Some of the examples of plotters include drum plotters, flatbed plotters, electrostatic plotters and inkjet plotters.

**Projector**

A projector is an output device that reproduces images by projecting them onto a screen, wall, or other surface using images created by a computer or Blu-ray player.

### 1.1.14 Describe the purpose and use of devices such as docking stations for mobile and laptop computers.

A docking station is a cradle for a portable media player that serves to charge and connect the unit to a receiving device. Docking stations enable users with a [laptop computer](https://www.computerhope.com/jargon/l/laptop.htm) to convert it into a [desktop computer](https://www.computerhope.com/jargon/d/desktopc.htm) when at the office or at home. For example, a business user could use a laptop on the road to create a document. When they return to the office, they could attach the laptop to the docking station to use their monitor, speakers, and office [printer](https://www.computerhope.com/jargon/p/printer.htm). Docking stations are now commonly used by developers in increasing the number of displays to reduce number of times of switching through tabs if they are to use one display.



Figure 1. 18: Ultraslim Docking station

## 1.2 Purpose of software

In section 1.1 our discussion mainly centred on hardware components. We must remember that for computers and devices to function, there must be instructions or programs controlling them. Without software, the hardware will not work as expected. We defined software as consisting of written, machine-readable instructions or [programs](https://www.techtarget.com/searchsoftwarequality/definition/program) that tell physical components what to do and when to execute the instructions. In this section we are going to dive deeper into the different software categories and their purposes.

* + 1. Describe the term software

VOCABULARY

Software – set of instructions, data or programs used to operate computers and execute specific tasks.

Software can be thought of as the variable part of a computer, while hardware is the invariable part. The two main categories of software are [application](https://www.techtarget.com/searchsoftwarequality/definition/application) software and [system software](https://www.techtarget.com/whatis/definition/system-software). Application software is software that fulfils a specific need or performs tasks. System software is designed to run a computer's hardware and provides a platform for applications to run on top of.

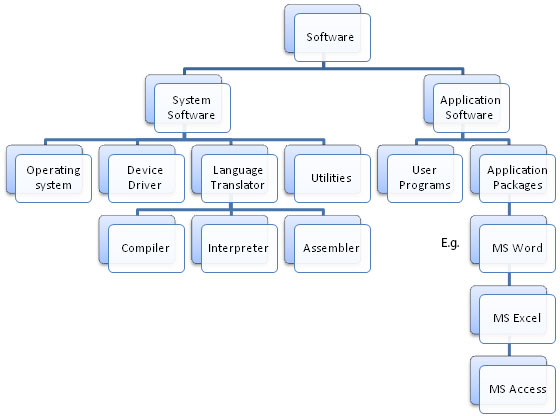


Figure 1.2. 1: Classification of software

System Software

These software packages are made to run the hardware and application software on a computer. The actions and features of the hardware and software are coordinated by the system software. Additionally, it manages how the computer hardware functions and offers a setting or platform in which all other software can operate.

**Operating Software**-An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently. Key functions are file management, memory management, resource management, security management among others. The three basic types of operating systems are stand-alone, server, and embedded.

1. Stand-alone operating system- This is an operating system that runs on a desktop computer, a laptop computer, or a handheld computer is called a stand-alone operating system. Examples are Microsoft Windows, Mac Os, and Linux.
2. Server operating system- these are specifically designed to support the network. Examples includes include windows server, UNIX, Solaris, and Netware.
3. Embedded Operating System-An embedded operating system uses on mobile devices and a wide variety of consumer electronics. It resides on a ROM chip. Common examples are windows mobile Plam OS, iPhone OS, BlackBerry, Google Android, Windows mobile, and Symbian OS.

**Device Drivers**-this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks.

**Utility software**- This software analyses and maintain a computer. This software is focused on how OS works on that basis it performs task to enable smooth functioning of computer. Examples of utility software includes antivirus, backup software, uninstaller, screen saver, file compression, virus scanner, file manager, disk compression tool all are utility software. Some of the utility software come along with operating software for example windows defender, disk cleanup, disk optimisation tools.

**Language Translators**-Translators are computer programs that translates program written in each programming language into a functionally equivalent program in a different language. There are mainly three Types of translators which are used to translate different programming languages into machine equivalent code: **Compiler, Interpreter, Assembler.**

1. **Compiler** is a computer program that translates code written in a high-level language to a low-level language, object/machine code. Examples of languages using compilers are C, C++, C#, Java VB etc.

**HOW DOES A COMPILER WORK?**

**Here are the stages involved in compiling a program.**

1. An interpreter program executes other programs directly, running through program code and executing it line-by-line. Examples of interpreted languages are Ruby, Perl, Python, PHP etc.

**HOW DOES INTERPRETER WORK?**

Four things happen inside an interpreter

* + 1. **Lexing** — The lexer breaks the line of code into tokens.
    2. **Parsing**- The parser employments these tokens to produce a structure, here, an Unique Language structure Tree, to portray the relationship between these tokens.
    3. **Compiling**- The compiler turns this AST into code object(s).
    4. **Translating**- The translator executes each code.

1. **Assembler**-An assembler translates assembly language into machine code. An assembler takes basic computer instructions and changes them into a design of bits that the computer processor can utilize to perform its fundamental operations.

Compilers take less execution time compared to an interpreter whereas assemblers take more time than the compiler. An interpreter analyses every line; hence is slower than running compiled code but it can take less time to interpret program code than to compile and then run it. Interpreters are written for multiple platforms; this means code written once can be run immediately on different systems without having to recompile for each.

Application software consists of application packages and user programs. Applications packages are always bought as a suite of programs for instance, Microsoft Office, LibreOffice and they provide common end-user needs. Examples of application programs include word processors e.g. Microsoft Word, WordPad, LibreOffice Writer, presentation programs e.g., PowerPoint, databases e.g. Microsoft Access, LibreOffice Base etc. User programs could be any example of programming languages such as C, C#, C++, Java, Kotlin, Go, R, Perl etc.

### Describe the purpose and function of software

**Functions of computer software**

* **Software** can make the computer compare data, make logical decisions, do mathematical calculations, store and retrieve data and instructions from primary or secondary sources, and carry out sequences of tasks.
* The software enables computer users to obtain what they need from the computer.
* **Software makes** the computer work towards giving the outputs in the manner the user wants them to, such as output on a screen, printouts, sounds, sending emails, etc.
* **Software** is used to translate programs written in different languages into machine languages, e.g., compilers, interpreters, and assemblers.

### 1.2.3 Basic concepts of software.

Operating systems serve as interfaces between users and computer hardware. Figure 1.2.2 illustrates how the operating systems interacts with hardware and providing interface for users. Interfaces are separated into two categories: command-line and graphical.

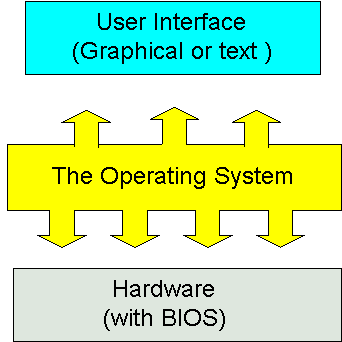


Figure 1.2. 2: Interfacing for users

* **Command-line Interface**: In a command-line interface, users interact with the computer through commands. Command line interface is difficult to work with as it requires knowledge of command language.
* **Graphical User Interface (GUI):** As a graphical user interface does not need you to memorize the command language, it is easier to understand. Users interact with the computer through icons and menus.

### 1.2.4 Contrast: System software vs application software

Table 1.2.1 compares system software and application software.

Table 1.2. 1:Comparison of system software and application software

|  |  |  |
| --- | --- | --- |
| System Software | | Application Software |
| System software is mainly designed for managing system resources. | Application software is designed to accomplish specific tasks. | |
| Programming of system software is complex. | Programming of application software is comparatively easy. | |
| A computer can not run without system software. | A computer can run without application software | |
| System software do not depend on application software | Application software depend on system software and can not run without system software. | |
| System software are typically written or programmed in a low-level language such as machine language or assembly language. | Application software are mostly written in high level languages such as Java, C++ etc. | |

Figure 1.2.3 illustrates an overview of computer hardware and software.

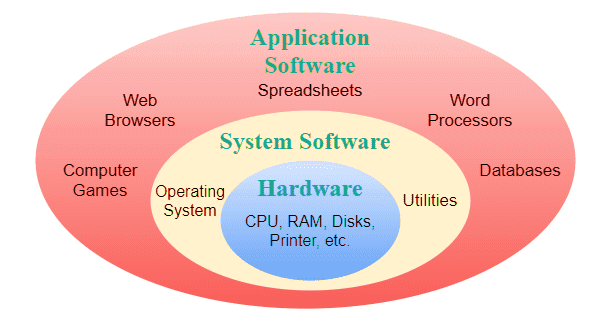
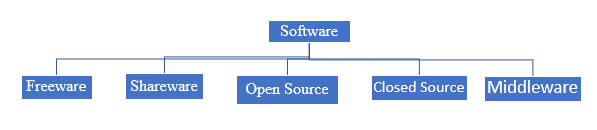


Figure 1.2. 3: Overview of computer

* + 1. Differentiate between: Shareware, Freeware, Open-Source Software and Proprietary software, Firmware

Computer software can also be classified according to availability and shareability and the categories are freeware, shareware, open source closed source and midlleware.



**Freeware**

Freeware is the software that is available to use for free of cost without any limitations. Example- Some of the google software is freeware.

**Shareware**

The software is copyrighted and distributed for free only for testing purposes. After the trial period ends, you must pay. For example, Microsoft Office (if you download MS-Office from the Microsoft website they give a one-month free trial after that you must pay for it yearly or monthly).

**Open Source**

This is provided for use, modification, and redistribution. Open-source software is downloaded from the internet at no cost. Example- Google play store apps, etc.

**Middleware**- is software that is used to bridge the gap between applications and other tools or databases. Some examples of middleware activities include handling data and [API](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API) management, authentication and messaging services.

**Closed source software**

Closed source software is software for which the source code is not freely available.  It is developed and provided to the user as a fully compiled, executable set of files. The developer often provides support to users after purchase and ensures that the software works as expected. This is usually developed if you can not find the specific software you want off the shelf and you approach software development houses to custom make for your specific needs.

**Middleware**

Middleware is software that bridges gaps between other applications, tools, and databases to provide unified services to users. It is commonly characterized as the glue that connects different software platforms and devices together.

**Proprietary**

Proprietary software is computer software where the source codes are publicly not available only the company that has created can modify it. A good example is Microsoft products where users must get a licence key to install them on to their devices.

**Firmware**

Firmware is a type of software that is embedded directly in a piece of hardware to make the hardware work as intended. Firmware is programmed by the manufacturer and is installed on a digital device right in the factory. All computing devices have firmware.

### 1.2.4 Discuss the process of how software is obtained and installed

Computer software can be installed from a compact disk or any other portable devices. Software which comes loaded on a compact disc is autoplay allowing to start the installation process immediately. If not, you will need to load the storage device and locate the installation file with .exe extension. Another alternative would be to install from the command line. In this case, you will need to navigate to the folder containing the installation file and run the executable setup file. Many times, this can be done by typing setup or install at the prompt to start the installation.

Today, the most common way to get new software is to download it from the Internet. software development houses are allowing customers to access software from their cloud storage after payment. Once a payment is received the user will receive a key which allows them to install the software. Applications like Microsoft Office and Adobe Photoshop can now be purchased and downloaded right to your computer. You can also install free software this way. For example, if you wanted to install the Google Chrome web browser you just download it from their server and install it to your personal computer. The installation file will be saved to your computer in .exe format.

### Differentiate between online software and installed software.

**Installed software** is also referred to as desktop software. Installed software must be purchased in physical form (such as a CD) or downloaded from the internet. In either case, the program is installed using an installation program. Once the software is installed, it is ready to use.

**Benefits of installed software**

1. Installed software can be viewed when offline
2. Installed software is accessible on any device irrespective of connectivity
3. Installed software offers a better User eXperience
4. User has full control of the data

**Online software** is any program accessed over an Internet connection using a Web browser. The website pages act as a user interface. Online software run on a secure, external server, which also stores any data collected.

**Benefits of online software**

1. No software to install
2. Complete flexibility on device type
3. No maintenance required.
4. Connect from anywhere at any time with internet
5. Centralised storage offering safe and secure data

### Discuss the following terms in relation to software. (Range: Compatibility issues, Versions, patches and service packs, Updating software)

**VOCABULARY**

Compatibility is the capacity for two systems to work together without having to be altered to do so.

Compatibility issues come up when users are using the same type of software for a task, such as word processors, that cannot communicate with each other. This could be due to a difference in their versions or because they are made by different companies. There are two types of compatibility: forward compatibility and backward compatibility.

Products that are designed to be compatible with future versions of themselves are referred to as **forward compatible**. Products designed for compatibility with older versions are said to be [**backward compatible**](https://www.techtarget.com/whatis/definition/backward-compatible-backward-compatibility).

Any application which can allow users to revert or rollback to the previous version is said to be backward compatibility.

**Software Version** means a unique build of the software released by a company identified by the alpha-numerical sequence assigned to it together with a corresponding release date e.g. [version 5](https://www.lawinsider.com/dictionary/software-version).9.1 released on 13 October 2022.

**Software Patches**

A software patch is a program that makes changes to software installed on a computer. Software companies issue patches to fix bugs or security problems in their programs or add new functions to the software.

**Service Packs**

A service pack (SP) is a collection of updates and fixes for an operating system or a software program. SPs are usually made up of patches. Many of these patches are often released before a larger SP, but the SP allows for an easy, single installation.

Both system software and application software serve to facilitate user interaction with computer systems and the accomplishment of various activities. There are variations in their designs and intended uses, though. Application software is created to address specific user demands in order to carry out specific tasks, as opposed to system software, which is created to manage system resources or processes.

## 1.3 The Linux shell

The shell, to put it simply, is a program that receives keyboard commands and sends them to the operating system for processing. A program called bash serves as the shell on the majority of Linux computers. You can communicate with the shell by using the terminal emulator, which launches a window. Linux employs a wide range of different terminal emulators.  Among them are gnome-terminal, konsole, xterm, rxvt, kvt, nxterm, and eterm, among others. On Raspberry Pi, terminal we can run Linux commands.

### 1.3.1 Launch a new Linux terminal on the Raspberry Pi

To launch the terminal on Raspberry Pi, you will need to navigate to the Raspberry Icon on the Desktop. Click Accessories and select terminal. Figure 1.3.1 illustrates how the Raspberry terminal looks like.

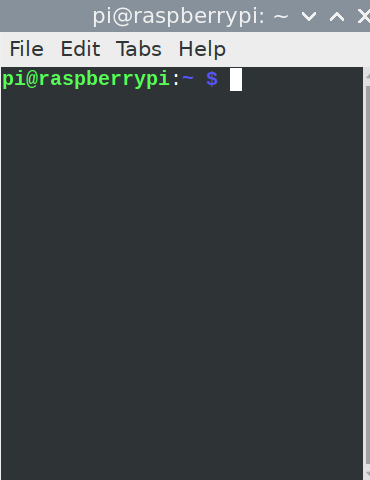


Figure 1.3. 1:Raspberry Terminal

### 1.3.2 Use the man command to get help

**man** is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. In Linux, man is an interface to view the system's reference manual. A user can request to display a man page by simply typing man followed by a space and then argument. Here its argument can be a command, utility or function. A manual page associated with each of these arguments is displayed. The syntax of man command is as follows:

$ man -options argument

Figure

For example:

man - -usage

Figure 1.3.2 shows the output of man –usage command to print usage syntax.

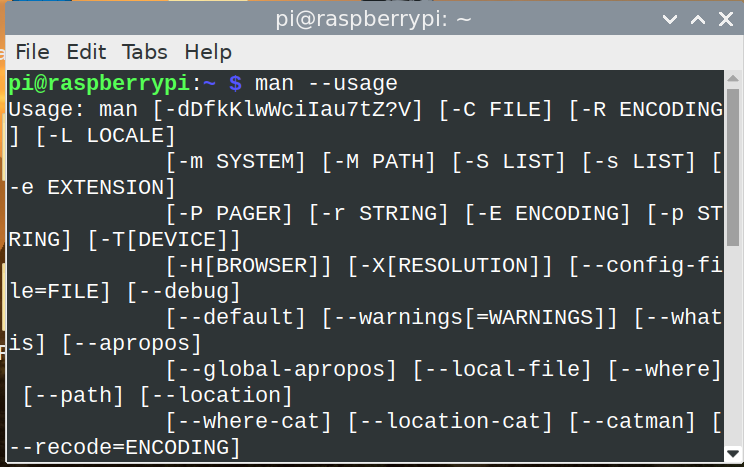


Figure 1.3. 2: Using man command

Another example could be bringing up the help by using man -h command as illustrated in Figure 1.3.3.

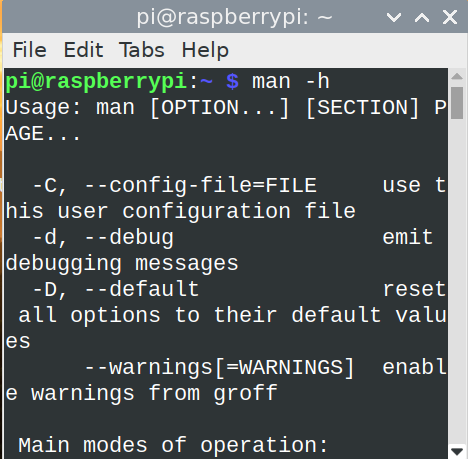


Figure 1.3. 3: using man -h command

So, to avoid buying a lot of books and saving trees- man is your answer. It provides the documentation at your fingertips.

### 1.3.3 Expand a Linux file path and explain each element

A file path is the human-readable representation of a file or folder’s location on a computer system. Files and folders on Linux are given names containing the usual components like the letters, numbers, and other characters on a keyboard. But when a file is inside a folder, or a folder is inside another folder, the / character shows the relationship between them. Figure 1.3.4 illustrates the file structure of the file add2.py. Just like in Windows where the file structure starts with C: on Raspberry Pi home is the root directory. If for and instance you want to see the exact file path, navigate to it on the terminal and type in the command: readlink -f <filename> as shown in Figure 1.3.4, we managed to find out the full path of the file called add2.py which is in /home/pi/Desktop folder

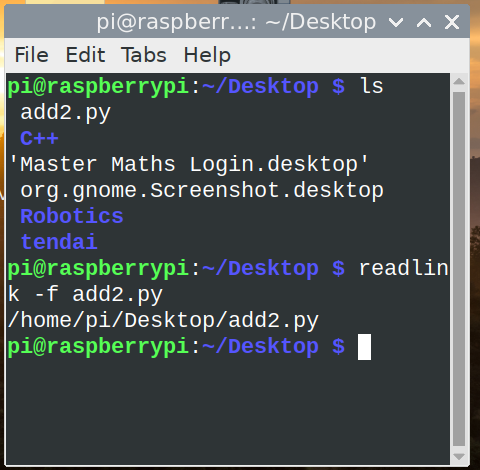


Figure 1.3. 4: File structure

### 1.3.4 List the contents of the current folder using the ls command

To list contents of the directory, the ls commands does the trick. See Figure 1.3.4

### 1.3.5 Change directly location using the cd command

To navigate from one directory to another, Linux uses cd (change directory) just like in Windows. Figure 1.3.5 illustrates how we used the cd command to navigate from Desktop folder to C++ folder.

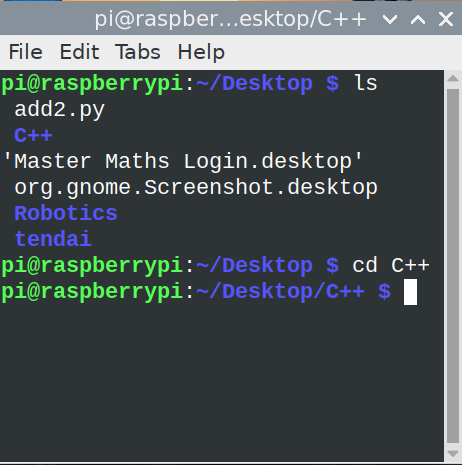


Figure 1.3. 5: Using cd command

If you are in a specific directory, you can navigate straight to the root directory by typing cd /. cd .. will make you move one level less from the current directory.

### 1.3.6 Create a new folder using the mkdir command

**mkdir** command in Linux allows the user to create directories. The user running this command must have sufficient permissions to create a directory in the parent directory; otherwise, a "permission denied" warning may appear.

**Syntax**

mkdir [options...] [directories ...]

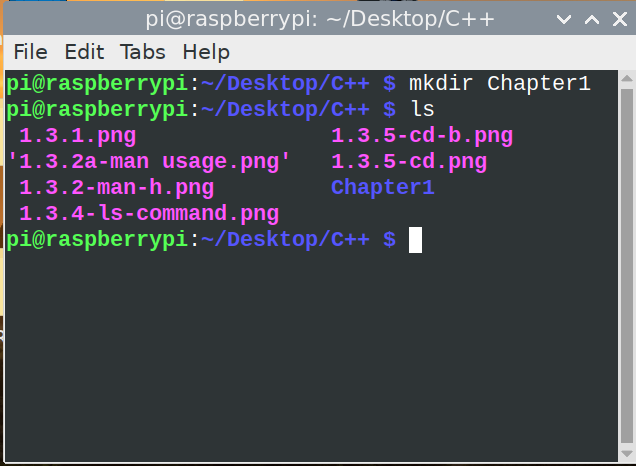


Figure 1.3. 6: Using mkdir command

In Figure 1.3.6, we managed to create a directory called Chapter1 inside the C++ folder. If you want to create two folders at once, just type the mkdir command and folder names and leave space between the names. There are other options which can be used together with mkdir command such as:

**- -v or - -verbose**: It displays a message for every directory created.   
**- -help**: It displays the help related information and exits.

### 1.3.7 Remove a folder using the rmdir command

rmdir command is used remove empty directories from the filesystem in Linux. The rmdir command removes each directory specified in the command line only if these directories are empty. Figure 1.3.7 shows how we removed the directory called Chapter1 from the C++ directory. After listing all files using the ls command, we can see that the directory is not there anymore.

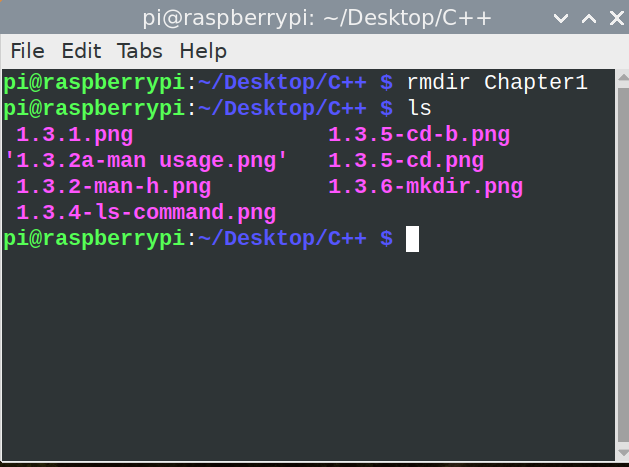


Figure 1.3. 7: Using rmdir command

### 1.3.8 Remove a file using the rm command

The rm command removes the entries for a specified file, group of files, or certain selected files from a list within a directory. When you use the rm command, user confirmation, read permission, and write permission are not required before a file is removed. In Figure 1.3.8, we demonstrate how to remove the file called newfile.py from C++ directory. We also demonstrated how to create a new file using touch command.

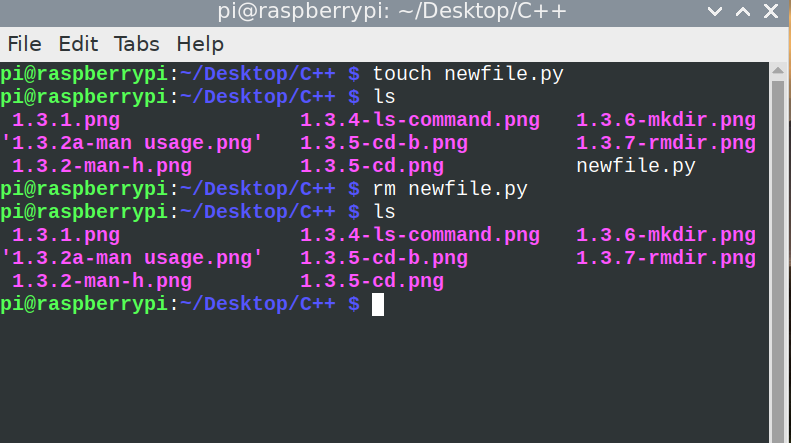


Figure 1.3. 8: Using rm command

### 1.3.9 Rename a file using the mv command

Use the mv command to change the name of a file without moving it to another directory. To rename a file, type the following:

$ mv <currentfilename> <proposedfilename>

Figure 1.3.9 illustrates how we renamed the file called newfile.py to newfile1.py by executing the following command:

$ mv newfile.py newfile1.py

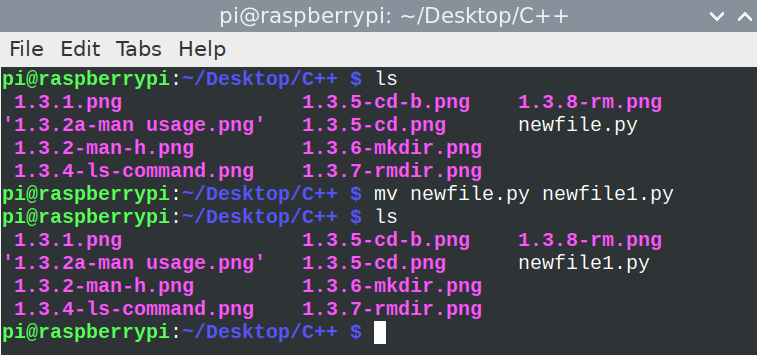


Figure 1.3. 9:Changing filename using mv command

### 1.3.10 Copy a file using the cp command

Use the [cp](https://www.ibm.com/docs/en/ssw_aix_72/c_commands/cp.html) command to create a copy of the contents of the file or directory specified by the source file or source directory parameters into the file or directory specified by the TargetFile or TargetDirectory parameters. If the file specified as the TargetFile exists, the copy writes over the original contents of the file without warning. If you are copying more than one SourceFile, the target must be a directory. Figure 1.3.10 demonstrates the cp command used to copy newfile.py to C++ folder.

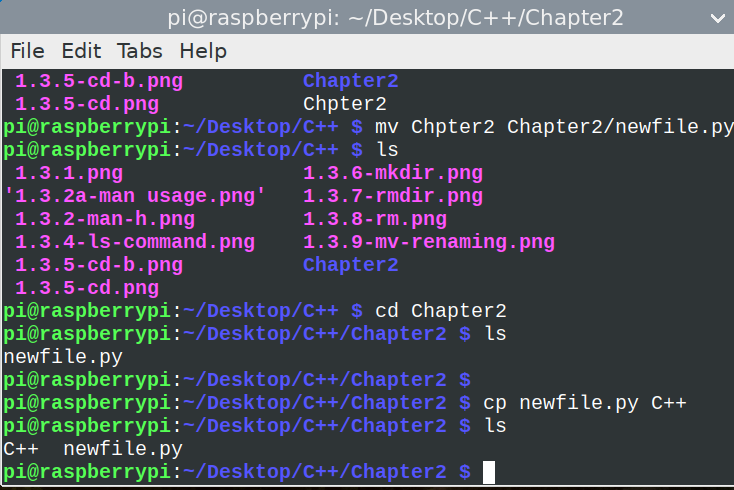


Figure 1.3. 10: Using cp command

### 1.3.11 Clear the command prompts screen using the cls command

clear clears your screen the text on the terminal. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen. clear ignores any command-line parameters that may be present.

### Run an executable file from the command line

To run an executable file the user rights of those files must be set correct. This can be done by doing the following:

Syntax

chmod +x <filename>.

chmod +x sample.sh.

There are more Linux commands to perform different operations. The link below will give a guide of other Linux commands.

**e-Link**

<https://www.guru99.com/linux-commands-cheat-sheet.html>

# Problem solving in computer programming

**Introduction**

The fundamental goal of computer science is problem solving. The problems that we want to solve can come from any real-world problem or perhaps even from the abstract world. We need to have a standard systematic approach to solving problems. First, programmers must comprehend how humans solve problems, then they must comprehend how to transform this "algorithm" into something that a machine can perform, and finally they must comprehend how to "code" the syntax (needed by a computer) to accomplish the task. Sometimes a machine will approach an issue entirely differently than a human would.

## 2.1 Problem solving process and concepts

### 2.1.1 Define the term problem solving

**VOCABULARY**

Problem Solving is the sequential process of analysing information related to a given situation and generating appropriate response options.

Using a computer, issues are resolved by getting some sort of user input (such as keyboard/mouse data or gaming control motions), processing the input, and creating some sort of output (e.g., images, text, sound). Hard disks or network devices may be used to store and transmit data occasionally.

### 2.1.2 Define the term computational thinking

**VOCABULARY**

Computational thinking is an interrelated set of skills and practices for solving complex problems, a way to learn topics in many disciplines, and a necessity for fully participating in a computational world.

Using computational thinking, we can break down complex problems into its component parts and come up with potential answers. Then, we may communicate these solutions in a manner that is understandable to a machine, a person, or both.

The four cornerstones of computational thinking

* **decomposition** - breaking down a complex problem or system into smaller, more manageable parts
* **pattern recognition** – looking for similarities among and within problems
* **abstraction** – focusing on the important information only, ignoring irrelevant detail
* **algorithms** - developing a step-by-step solution to the problem, or the rules to follow to solve the problem

A complex problem is broken down into a succession of smaller, more manageable problems using computational thinking (decomposition). Then, each of these smaller issues can be examined separately, considering how comparable issues have been resolved in the past (pattern recognition), concentrating solely on crucial information, and eliminating irrelevant data (abstraction). Next, straightforward guidelines or procedures to address each of the lesser issues can be created (algorithms).

### 2.1.3 Describe the phases of the PLDC (Program Development Life Cycle)

**VOCABULARY**

The program development life cycle (PDLC) is a set of steps or phases which are used to develop a program in any programming language.

PDLC provides an organized plan for breaking down the task of program development into manageable chunks, each of which must be successfully completed before moving on to the next phase. Figure 2.1 illustrates the six phases of program development lifecycle.

Figure 2. 1: Program Development Lifecycle

Program Development Life Cycle consists of six stages, and these are:

1. Defining problem
2. Designing the solution
3. Coding the program
4. Testing and Debugging
5. Implementation of the solution
6. Review and Maintenance

**Defining the problem**

Analyse and define the problem, check, and understand that the problem is clearly defined. What are the inputs /outputs, process steps, logic, rules and requirements. Common roles of people in this stage includes business analysts.

**Designing the solution**

The first step in program design is to concentrate on the major objective that the program is attempting to accomplish, after which the program is divided into manageable parts, each of which contributes to the overall objective. Top-bottom programming, often known as modular programming, is this method of program design. Different design tools are used in this stage and some of these includes pseudocode, flowcharts, algorithms, Use case diagrams, data flow diagrams, decision tables, Input Processing -Output tables, hierarchy charts and many others. Systems architect and UX/UI designers are the key in this stage of development.

**Coding the program**

The actual programming instructions for the actions specified in the previous phase are now written or implemented using a programming language. In this stage, we write the program. Utilizing programming languages like C, C++, Java, etc., we create the software to address the problem at hand. The technique of programming using only well-defined control structures is known as *Structured programming*. Programmer must follow the language rules, violation of any rule causes *error*. These errors must be eliminated before going to the next step. Developers both front end and back end are the common people at this stage.

**Testing and Debugging**

*Software Testing* is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free.

The application will run following the correction of syntax problems. However, the program's results might not be accurate. This is a result of a logical error in the software. A logical error occurs when the programmer makes a mistake when creating a solution to a problem. Therefore, by carefully reviewing the program output using Test data, the programmer must identify and fix logical problems. Bugs are a collective term for both syntax and logical errors. The process of identifying errors and eliminating them is known as *Debugging*.

**Testing levels**

The following are the main types of software testing methodologies:

|  |  |
| --- | --- |
| **Testing level** | **Activities** |
| Integration testing | This brings together two or more application modules to make sure they work together. Defects in interface, communication, and data flow between modules are also revealed by this kind of testing. |
| Unit testing | Unit testing is typically done throughout the application development process and its goal is to make sure that every single unit or component works as planned. |
| Regression testing | This determines if adding additional features results in a decrease in an application's functionality. |
| Stress testing | This gauges a program's robustness by seeing how much stress it can withstand before failing. |
| White Box Testing | In white-box testing, test cases are created using programming knowledge and an internal viewpoint of the system. Typically, this testing is carried out at the unit level. |
| Black Box Testing | Software testers use the black box testing technique to assess the functionality of the program being tested without examining the internal code layout. |
| Acceptance testing | a testing technique performed to determine whether or not the software system has met the requirement specification |

**Implementation of the solution**

At this testing, all testing has been done and the developers are happy with the product. The program is deployed (installed) at the user’s site. Document the solution using comments with the program, and support documentation for users (i.e., those that won’t be looking at the code).

**Review and Maintenance**

The application is monitored in this instance as well until the user gives it the all-clear. Even after it is finished, the software still needs to be constantly maintained and assessed. The programming team upgrades the software and corrects program flaws during software maintenance.

I hope you have learnt how software is developed, starting with brainstorming to identify the problem up to the last stage of review and maintenance. You must remember that execution of these stages can be done only after completion of the previous phase.

### 2.1.4 Describe the purpose of problem solving leading to solutions

**VOCABULARY**

Problem solving is a mental process that involves discovering and analysing a particular issue, developing strategies, and organizing skills and knowledge to overcome obstacles and find viable solutions that best resolve the problem.

There are 6 steps that you should follow to solve a problem:

1. Understand the Problem

2. Formulate a Model

3. Develop an Algorithm

4. Write the Program

5. Test the Program

6. Evaluate the Solution

As you will see, the above stages are just like those of program development lifecycle.

Consider a simple example of how the input/process/output works on a simple problem:

Example: Calculate the average grade for all students in a class.

1. Input: get all the grades
2. Process: add them all up and compute the average grade.
3. Output: output the answer to either the monitor, to the printer, to the USB flash drive or hard disk … or a combination of any of these devices.

Let us try to make use of the problem-solving steps to solve the above problem.

**Step 1: Understand the problem**

Some of the key questions that should be answered in order to understand the problem are:

* What input data/information is available?
* What does it represent?
* What format is it in?
* Do I have everything that I need?
* What output information am I trying to produce and in what format?

We are aware that the input in our example is a collection of grades. But we must comprehend the structure of the grades. Each grade could be a letter grade from A+ to F or a number from 0 to 100. If the answer is a number, the grade may be a real number, such as 73, or a full integer.

**STEP 2: Formulate a Model**

We will calculate the average of the incoming grades in our example. Therefore, we must be aware of the procedure (or formula) for calculating the average of a collection of numbers.

Assuming that the input data is a bunch of integers or real numbers x1,x2,…,xn representing a grade percentage, we can use the following computational model:

Average1 = (x1 + x2 + x3 + … + xn) / n

where the result will be a number from 0 to 100.

We can also draw up a chart of grade values and symbols.

**STEP 3: Develop an Algorithm**

To develop an algorithm, we need to represent the instructions in some way that is understandable to a person who is trying to figure out the steps involved. To allow even non-computer people to understand the development, pseudocode would be the best to depict the scenario.

*Pseudocode* is a simple and concise sequence of English-like instructions to solve a problem.

Input: Number of terms n

Output: Sum and average of those n terms

Procedure SumAverage

Sum=0

i=0

Repeat for each i <n:

Read a number x

i =i +1

Avg=sum /n

Sum =sum + x

Print sum and average

**STEP 4: Write the Program**

At this stage, the design structure is changed into a programming code of choice. In our case we used C++ programming language.

Here is the sample program

#include <iostream>

using namespace std.

 int main () {

    int i, count, sum, inputArray[500];

    float average.

     cout << "Enter number of elements\n";

    cin >> count.

    cout << "Enter " << count << " elements\n”.

    // Read "count" elements from user

    for (i = 0; i < count; i++) {

        cin >> inputArray[i];

    }

    sum = 0;

    // Find sum of all array elements

    for (i = 0; i < count; i++) {

        sum += inputArray[i];

    }

  average = (float)sum / count;

    cout << "Average = " << average;

     return 0;

}

After coding we will need to compile our program to check for errors. *Compiling* is the process of converting a program into instructions that can be understood by the computer.

**STEP 5: Test the Program**

Once a program has been written and has passed compilation, it must be checked to see if it solves the problem it was designed to and that the answers are accurate. If everything is in order, the output from your application should be correct after running the program. Your program should be bug-free as much as possible. Test your program with many test cases (called a test suite) to find bugs effectively.

**STEP 6: Evaluate the Solution**

You should re-consider the original problem and ensure that your answer is formatted into a proper solution after your program produces a correct result. Cross check with the objective of the problem and the results. Now you can deploy the solution and constantly maintain it. Sometimes, the program may misbehave days or months after deployment due various reasons. So, you will need to regularly test it with the acceptable test data.

So now we have illustrated how you can implement problem-solving steps. You should make sure to follow these steps as a guide to coming up with the correct solution to a given problem.

2.1.5 Explain and apply various problem-solving steps.

Polya’s First Principle: Understand the problem. His techniques of problem solving became very common and are used in the modern days. In 1945 George Polya published the book “How to Solve It” which quickly became his most prized publication. In this book he identifies four basic principles of problem solving.

First Principle: **Understand the problem**

* Some of the questions that can help elucidate the problem could be as follows:
* Do you understand all the words used in stating the problem?
* What are you asked to find or show?
* Can you restate the problem in your own words?
* Can you think of a picture or diagram that might help you understand the problem?
* Is there enough information to enable you to find a solution?

Second Principle: **Devise a plan**

There are numerous rational approaches to resolve issues, says Polya. The greatest way to learn how to choose an effective strategy is to solve lots of issues. Following the stages will make problem solving get easier and easier. Some of the strategies include:

* Guess and check
* Eliminating possibilities
* Use a model
* Use direct reasoning
* Look for a pattern
* Draw a picture

Third Principle: **Carry out the plan**

Typically, this stage is simpler than creating the strategy. Given that you have the essential abilities, all you really need is care and patience. Stick to the strategy you've picked. If it doesn't stop failing, throw it away and pick another. Don't be fooled; this is how math is done, even by experts.

Fourth Principle: **Look back**

The idea behind looking back is to allow the process of reflection on what worked well and what did not work. People learn more from reflections. You will be able to foresee what approach to take to address difficulties in the future by doing this.

### 2.1.6 Use appropriate tools and techniques to present a solution. Range:

**VOCABULARY**

Agile software development refers to software development methodologies centred around the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

User stories are among some of the techniques to present a solution. User stories are mostly used in agile development methodologies.

In agile product management and software development a user story is a succinct, casual, and straightforward summary of software functionality that the system's end users need. Its main objective is to offer software features that will meet the needs of the consumer. User stories are simple, yet extremely powerful constructs: they describe pieces of functionality from a user’s point of view, expressed in a solid, compact way.

***Why User Stories?***

1. User stories provide an excellent way to [define your product with clarity](https://www.theinnovationmode.com/the-innovation-blog/the-mvp-recipe-what-to-include-in-your-product-when-and-why).
2. User stories help to achieve *cross-team clarity* on *what* to build, for *whom*, *why,* and when.
3. User stories encourage participation by non-technical members.
4. User stories help in [defining the entire product](https://www.theinnovationmode.com/the-innovation-blog/the-mvp-recipe-what-to-include-in-your-product-when-and-why) — as a set of solid, wisely-prioritized stories.

User stories are completely from the end-user perspective which follows the Role-Feature-Benefit pattern. The pattern is as follows:

As a [ type of user], I want [ an action], so that [ some reason]

Some points outlined which are taken into consideration during writing user stories like

1. Requirements
2. Tasks and their subtasks
3. Actual user
4. Importance to user words/feedback
5. Breaking user stories for larger requirements

A common principle when writing user stories is to consider 3C’s.

* Card-write stories on cards
* Conversation-conduct conversations with the people involved to get more information.
* Confirmation-meet the acceptance criteria of the software.

**Example of a user story**

As a customer I want ability to book a movie ticket that matches my preferences so that I get to quickly and easily book the movie of my choice.

User Story Description

This feature will involve user selecting a specific city, searching for the movie name, selecting a specific timeslot, and then completing the order booking formalities.

**Acceptance Criteria:**

1. User navigates to the search movie page
2. User selects city
3. User enters movie name
4. System searches for the matching movies and displays results
5. User selects specific movie timing and proceeds to book
6. User enters no. of guests and seats
7. User provides payment information
8. System validates the payment information and confirms the booking
9. System sends email/SMS to the user with booking confirmation details

**Key Terms**

* **user story** – one short sentence in everyday language of the end user that states what a user does as part of his or her work
* **acceptance criteria** – features that must be present in the final system for the user to be satisfied
* **use case** – an activity that the system performs, usually in response to a request by a user
* **user goal technique** – a technique to identify use cases by determining what specific goals or objectives must be completed by a user
* **event** – something that occurs at a specific time and place, can be precisely identified, and must be remembered by the system

## 2.2 Construct an algorithm and present a solution to a given problem

Earlier in this chapter, we spoke about the program development lifecycle and the associated stages. We highlighted the design tools, but we want to go a bit deeper with examples. In this section we will discuss flowcharts, algorithms and pseudocodes.

2.2.1 Define the term algorithm and its purpose in the problem-solving process.

A flowchart is a schematic representation of an algorithm or a stepwise process, showing the steps as boxes of various kinds, and their order by connecting these with arrows.

A process or program can be designed or documented using flowcharts. A flow chart, also known as a flow diagram, is a graphic representation of a system or process that shows the order in which actions must be completed in order to produce an outcome.

**Flowchart symbols**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Name** | **Function** |
|  | Start/End | An oval shape represents a start or end of a program |
|  | Arrows | A line is a connector that shows relationships between the representative shapes |
|  | Input/Output | A parallelogram represents an input or output to the program |
|  | Process | A rectangle represents a process |
|  | Decision | A diamond indicates a decision |
|  | Connector symbol | Indicates that the flow continues where a matching symbol has been placed. |

Table 2. 1:Flowchart symbols

2.2.2 Implement and understand the basic algorithmic constructs used to create a flowchart**.** Range: Input, Output, Processing and Calculations, Selection Iteration

VOCABULARY

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.

**Characteristics of an Algorithm**

* **Unambiguous**
* **Input-must have 0 or more inputs**
* **Output** − should have 1 or more well-defined outputs.
* **Finiteness**-must terminate after several steps
* **Feasibility**-should be feasible with available resources
* **Independent**-must have step by step directions independent from other programs

Example: Design an algorithm to add two numbers and display the result.

Step 1 − START

Step 2 − declare three integers a, b & c

Step 3 − define values of a & b

Step 4 − add values of a & b

Step 5 − store output of step 4 to c

Step 6 − print c

Step 7 − STOP

**Types of Algorithms**

* **Brute Force Algorithm**-
* **Recursive Algorithm**- In this instance, a problem is divided into multiple smaller components and repeatedly called by the same function.
* [**Backtracking Algorithm**](https://www.geeksforgeeks.org/backtracking-algorithms/)**-** Every time a solution fails, we go back to the original problem, build on the new one, and repeat the process until the problem is solved or all potential solutions have been considered.
* **Divide-and-conquer algorithm-**A a problem is repeatedly divided into two or more subproblems of the same or similar type, until these are sufficiently straightforward to be solved by themselves.

**PROBLEM**: Find the sum of 5 numbers. Use an algorithm to solve the problem

Solution

Algorithm (in simple English)

1. Initialize sum = 0 and count = 0    (PROCESS)
2. Enter n    (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
5. if YES go to step 2  
   else  
   Print sum (I/O)

Flowchart for the above problem

A picture containing diagram

Description automatically generated

### 2.2.3 Create a flowchart to present a particular algorithm and its associated tasks

In section 2.2.1 we identified the common shapes for making a flowchart. Let us consider the following challenge.

Problem: Add two numbers from the keyboard and display the output. Design a flowchart to solve the problem.

To solve this problem we will take a variable sum and set it to zero. Then we will take the two numbers number1 and number2 as input. Next we will add both the numbers and save the result in the variable sum i.e., sum = number1+number2. Finally, we will print the value stored in the variable sum.

Here is the algorithm for the above example.

Algorithm (in simple English)

* Initialize sum = 0 (PROCESS)
* Enter the numbers (I/O)
* Add them and store the result in sum (PROCESS)
* Print sum (I/O)

**Flowchart**

Text

Description automatically generated

Figure 2. 2: Adding two numbers

Figure 2.2 is a sequential flowchart since steps are followed one after the other in sequence without a choice of condition followed. If the program requires a choice of events based on specific conditions being met, conditional flowcharts will be used. A conditional flowchart is used when a condition is imposed on a problem. The condition will either be true or false. The course of the problem depends on the answer to the condition. Here is the challenge:

Write a program to check if the given number is a multiple of 3 or not.

Here are the steps:

1. Input the number 'n'.
2. Divide 'n' be 3.
3. If reminder equals 0, print 'n' is a multiple of 3'.
4. If reminder does not equal to 0, print 'n' is not a multiple of 3'.

Use a flowchart to depict the above scenario.

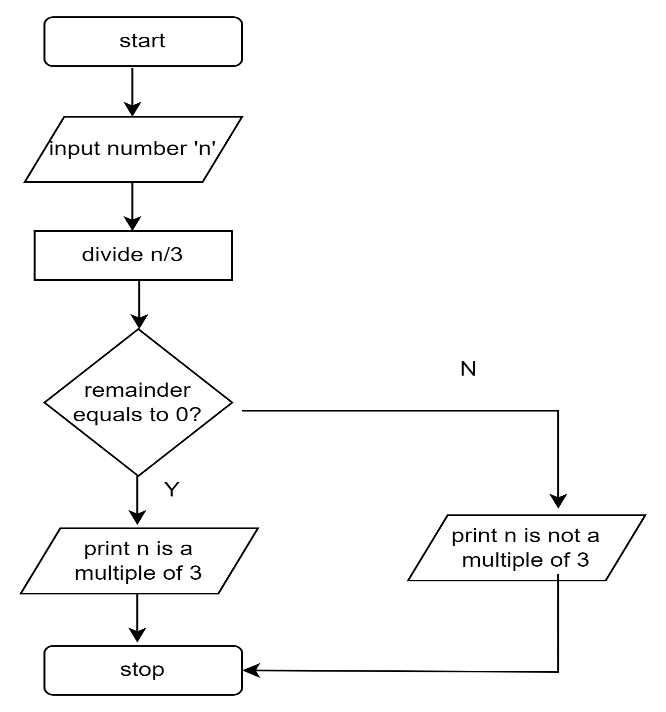


Figure 2. 3: conditional flowchart

**Repetition flowchart** is used when the program requires the act of repeating one or more steps in a process. It involves a **branching***backwards* away from the normal sequence of steps towards an earlier step. The branching decision is based on a **condition** (relationship between the values of known data) at the time that the branching test is performed.

Program: Write a program to print hello five times.

In this case, we take a variable count and set it to zero. Then we print "Hello World" and increment count by 1.

i.e., count = count + 1

Next we check if count is less than 10. If this is TRUE then we again print "Hello World" and increment the variable count. On the other hand, if the condition if FALSE then we will stop.

Here is the flowchart for the problem.

A picture containing diagram

Description automatically generated

Figure 2. 4:Repetition flowchart

Let us consider the following challenge.

Challenge: Draw a flowchart to log in to Facebook account

To log in to Facebook account we first enter the Facebook URL www.facebook.com in our browser. This request is sent to the Facebook server and it responds by sending us the home page of Facebook. Next, we enter our registered Email ID and Password and click the Login button. Then our login credential is checked. If it is correct, we are show our profile. On the other hand, if the login credential is wrong then an error occurs, and we are prompted to re-enter our Email ID and Password.

Diagram

Description automatically generated with medium confidence

PROBLEM: Write an algorithm to determine a student’s final grade from 4 tests scores and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

A picture containing polygon

Description automatically generated

Figure 2. 5:Average of four test scores

### 2.2.4 Interpret a basic flow chart and describe its intended operation / function

How frequently have you found it difficult to comprehend a process after being given a thorough explanation? In these circumstances, flow charts are a helpful tool since they make a process simple to comprehend at a glance. They effectively illustrate what happens at each stage and how this influences other decisions and actions using only a few basic words and symbols.

**When to Use a Flow Chart**

* Defining a process.
* [Standardize](https://www.mindtools.com/pages/article/5s-system.htm)  a process.
* Communicating a process
* Identify bottlenecks or waste in a process

We have already discussed the key shapes symbols used in creating a flowchart. Interpreting your Flowchart will help you to :

* Determine who is involved in the process.
* Form theories about root causes.
* Identify ways to streamline the process.
* Determine how to implement changes to the process.
* Locate cost-added-only steps.
* Provide training on how the process works or should work.

**Summary of design tools**

There are quite a lot of tools which can be used in designing systems. None of them can be regarded as the best but we could say they are situational. There are certain cases where one tool may work better than the other. Also, it depends on whether the developers understand them and provide an easy depiction of the process or problem-solving steps.

References

Polya, G., 2004. *How to solve it: A new aspect of mathematical method* (Vol. 85). Princeton university press.

# 4.0 Building and running C/C++ applications

**4.1 Introduction**

In chapter 1.2.1, we discussed about the different types of language translators. Translators convert programs written in high-level languages into machine code that a computer understands. To bring you up to speed, a language translator in software programming terms, is a generic term that could refer to a compiler, assembler, or interpreter. In this chapter, we are going to explain more about the C or C++ compiler.

### 4.1.1 Define *the* term compiler

**VOCABULARY**

A compiler is a language processor that reads a whole source program written in high-level language in one go and converts it into an equivalent program written in machine code.

Because they must instantly transform higher-level code into lower-level machine language and save the executable object code to memory, compilers can be slow. A compiler generates machine code for a particular Instruction Set Architecture (ISA), which is unique to each processor. For instance, without a specific compiler, it is impossible to compile code for the x86 architecture and run it on the Million instructions per second (MIPS) architecture.

If the source code in a compiler is error-free, it will translate correctly to object code. If there are any mistakes in the source code, the compiler specifies the errors with line numbers at the end of the compilation. Before the compiler can correctly recompile the source code, the errors must be fixed. Examples of languages which uses compilers are C, C++, C#, Java,  Erlang, Haskell, Rust, and Go.

### Define the term source code

In general, programming statements written by a programmer using a text editor or visual programming tool and then saved in a file are referred to as source code. When the source code is compiled using a C compiler, the output, a compiled file, is referred to as object code.

It's common to refer to a computer program's source code and object code as its "before" and "after" versions respectively. Since there is only one form of the code, the words source code and object code do not apply to script (noncompiled or interpreted) programming languages like JavaScript. Figure 4.1 illustrates the conversion from source code to object code.

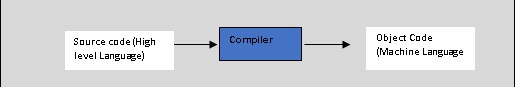


Figure 4. 1:Source code to object code

To write a source code, all you really need is a simple text editor like Notepad++, notepad or any other. This way, source code can be saved as plain text with the correct file name ending for the programming language. So, if you find a file with the ending “.cpp” on your hard drive, then it’s actually a text file, containing code in the C++ programming language.

Here is an example of C++ language source code:

**EXAMPLE 4.1**

#include <iostream>  
using namespace std;  
  
int main() {  
 cout << "Hello, World! \n" ;  
 cout << "This is a test program!" ;  
 return 0;  
}

It is obvious to everyone who isn't a computer programmer that the code on Example 4.1 has anything to do with the text being printed. "Hello World” and “This is a test program!" . When executed the output is as follows:

C:\ CLionProjects\untitled\cmake-build-debug\test.exe

Hello, World!

This is a test program!

Process finished with exit code 0

We are not going to explain the code for now, but we wanted to show what will be printed.

The object code file contains a sequence of machine-readable instructions that is processed by the CPU in a computer. Operating system or application software is usually in the form of compiled object code.

Here is a list of some popular C++ compilers/IDE

* [C++ Builder](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#1_C_Builder)
* [Microsoft Visual C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#2_Microsoft_Visual_C)
* [Eclipse IDE](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#3_Eclipse_IDE)
* [Codeblocks](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#4_Codeblocks)
* [Dev-C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#5_Dev-C)
* [NetBeans IDE](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#6_NetBeans_IDE)
* [Cygwin](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#7_Cygwin)
* [GCC](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#8_GCC)
* [MinGW](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#11_MinGW)
* [CodeLite](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#12_CodeLite)
* [Clang C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#14_Clang_C)
* [Clion](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#15_Clion)

### 4.1.3 Explain what a decompiler is used for

In software programming, a decompiler converts executable programs or machine language into a form programmers can understand.

It performs the operations of a compiler, which translates source code into an executable format, but in reverse. A decompiler’s recipient is a human user, whereas the compiler’s is the machine.  Decompilation is a type of reverse-engineering that performs the opposite operations of a compiler.

**NOTE**

To convert a programming source code into a working program, it must first be compiled -- i.e., converted into a series of binary bits or digits (that is, 1s and 0s) that can be understood by the computer. This operation can be reversed by decompiling the final program (which is why decompiling is described as a type of **reverse-engineering)**.

Decompilation can be used unethically to copy source code for reuse or adaption without the owner's consent. A decompiler can be useful in some cases for the following purposes:

* Recovery of lost source code to archive or maintain the code
* Debugging programs
* Antivirus capability to find vulnerabilities in the program

Some common examples of decompilers includes:

* IDA Pro
* Hex-Rays Decompiler
* CFF Explorer
* Hiew

### 4.1.4 Define the term interpreter

**VOCABULARY**

An interpreter program executes other programs directly, running through program code and executing it line-by-line.

Interpreters translates only one statement of the program at a time and as well are often are smaller than compilers.

### Difference between a compiler and an interpreter

Table 4. 1: Comparison of a compiler vs an interpreter

| **Compiler** | **Interpreter** |
| --- | --- |
| A compiler translates the entire source code in a single run. | An interpreter translates the entire source code line by line. |
| Compilers are faster than an interpreter. | Interpreters are slower than the compiler. |
| CPU utilization is more. | CPU utilization is less as compared to the compiler. |
| Both syntactic and semantic errors can be checked simultaneously. | Only syntactic errors are checked. |
| The compiler is larger. | Interpreters are often smaller than compilers. |
| The localization of errors is difficult. | The localization of error is easier than the compiler. |
| The compiler is used by the language such as C, C++. | An interpreter is used by languages such as Java and Python. |

### 4.1.6 Explain what a binary is and when it is produced

A binary file is one that contains data in a format known as binary, which consists of a series of bytes that are sequential and each eight bits long. The material needs to be interpreted by a software program or hardware processor that is aware of how it is formatted and how to read the data in advance. Among the many different file kinds that can be stored as binary files are executables, libraries, images, databases, archives, and many more.

Binary files often include a header indicating the type of file. Application developers and other software developers commonly use binary files. An executable program is generally identified as a binary file with an extension such as .bin or .exe. Binary files are often described as executables or as compiled applications. Files of this type are also known as object code.

### 4.1.7 Key differences between C and C++ compiler

Applications, games, databases, operating systems, and games are commonly developed in C and C++. Although C and C++ sound similar, their features and usage differ. The C programming language supports objects and classes. C++, on the other hand, is an enhanced version of C that supports object-oriented programming. C is a procedural-oriented language and C++ is an object-oriented language, which is the most important difference between them. Table 4.2 demonstrates the key differences between C and C++ programming languages.

Table 4. 2: Differences between C and C++ programming

|  |  |  |
| --- | --- | --- |
| **Key** | **C** | **C++** |
| Developer | C developed by Dennis Richie between 1969-1973 at AT&T Bell Labs. | C++ developed by Bjarne Stroustrup in 1979. |
| Object Oriented Programming | C does not support OOPS concepts like polymorphism, encapsulation, and inheritance. C is procedural. | C++ is a object oriented programming and supports OOPS concepts like polymorphism, encapsulation, and inheritance. C++ supports both procedural and OOP. |
| Function | C is function driven language. | C++ is Object driven language. |
| File extension | The file extension of a program in C language is .c. | The file extension of a C++ program is .cpp. |
| Header | C uses <stdio.h> header file for input/output operations. | uses <iostream> header file for input/output operations. |

### 4.2.8 Basic three stage compiler design (Front Middle Back end)

A compiler typically has six states, and the high-level language source code that it receives is processed by each state individually to produce machine-understandable code or object code as an output. The three main stages of the compiler are backend, middle end and front end.

**Front End Phase**- Phases or parts of phases in the front end depend on the source language and are independent of the target machine. The main activity involves analysis of source code into intermediate code. These generally consist of lexical analysis, semantic analysis, syntactic analysis, symbol table creation, and intermediate code generation.

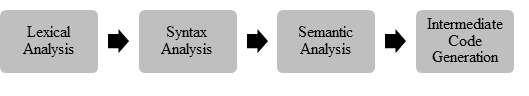


Figure 4. 2: Front End Phase

**Lexical Analysis** -Convert a program into sequence of tokens.

**Syntax Analysis** - Recover the structure portrayed by utilizing arrangement of tokens from past scanner.

**Semantic Analysis-**Ensures program has a well-defined meaning.

**IR Generation-**Compilers create an explicit low-level or machine-like intermediate representation that should be straightforward to deliver and straightforward for the target machine to understand.

**Code Generation-**The compiler converts the optimized intermediate code to the machine code dedicated to the target machine.

**Middle End**

Independent of the desired CPU architecture, the middle end optimizes the intermediate representation (IR). A source code/machine code independence is intended to make it possible to share generic optimizations between versions of the compiler that support different languages and target processors. Middle end performs target-independent optimizations on the IR (e.g., removal of useless or unreachable code, constant propagation, hoisting code from out of a loop, loop unrolling, etc.)

**Back End**

The back end of compilers includes the parts that are independent of the source language and dependent only on the target computer. In the back end, code generation and essential aspects of code optimization phases, along with error handling and symbol table activities are also provided.

Figure 4. 3: Back End compiler phase

In summary, we have learnt about compilers and their duty being to change source code in low level machine code at once. Compliers consists of front, middle and back end. The middle takes the output of the front end, transforms it somehow, then feeds that to the back-end. The [front end converts the source code to an intermediate representation](https://opensource.com/article/18/4/introduction-python-bytecode) which is not language specific. The backend converts its input to machine instructions for the target actual or virtual machine.

## 4.2 C/C++ on the Desktop

**Introduction**

In section 4.2.2, we listed some of the C++ compilers and some come bundled with the IDE. For windows platforms, the common two options are CygWin and MingW.

The [CygWin](http://www.cygwin.com/) package provides a Posix layer in Windows, allowing much Unix software to be compiled and run without modification. When you install CygWin, you get a good command shell (bash), which may be useful. The g++ compiler can also be selected from the CygWin setup utility.

A disadvantage of CygWin is that the compiler produces code that relies on the Posix layer, which in turn is provided by a CygWin DLL. Binary executables produced by the CygWin port of g++ can only be run on other machines that have a copy of that Dynamic-link library (DLL).

MinGW, a “spin-off” project from CygWin, provides an alternate port of the g++ compiler that works without a special DLL. You may have used this already, as this is the g++ typically shipped with Code::Blocks and other basic Windows Integrated Development Environment (IDEs).

### 4.2.1 Install and configure C/C++ compiler on desktop PC

1. To install C++ compiler, search for Code::Blocks in your browser.
2. On the downloads page, select download the binary release
3. Move to the Microsoft downloads section. In our case we will select the Code::Blocks IDE and the compiler. At the time of writing this book, the latest version was codeblocks-20.03mingw-setup.exe. MingW is our C++ compiler.
4. Open and run the downloaded file and it will install the IDE plus the compiler
5. You will see the following screen.

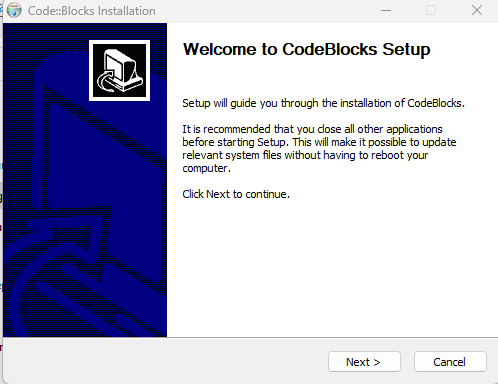


Figure 4. 4: CodeBlocks installation screen

Agree to the licence. It will also indicate the default compiler as GNU GCC compiler.

1. You will need to be very carefully with the path of MingW otherwise in some instances you run the program and it doesn’t pick up the compiler.
2. If this happens open Code::Blocks, click on compiler and select the Toolchain executables. Identify the path of your MingW and you will be good to go. Congratulations your C++ compiler is now installed

**Verify if compiler exists**

Go to the folder with MingW and type the following command: g++ --version

The output will be as follows:

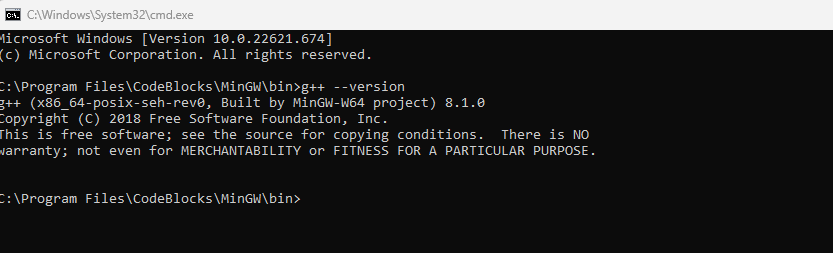


Figure 4. 5: Verifying the compiler

Type the following commands also:

* gdb --version
* make –version

### 4.2.2 Define the term IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.

### 4.2.3 Use of IDE

A source code editor, build automation tools, and a debugger are the typical components of an IDE. Most contemporary IDEs offer intelligent code completion. Compiler, interpreter, or both may be included in some IDEs.  There are occasions when a version control system or different technologies to make creating a graphical user interface (GUI) easier are included. For usage in object-oriented software development, many contemporary IDEs additionally include a class browser, an object browser, and a class hierarchy diagram. Designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools a developer needs, including:

* **Code editor:** Designed for writing and editing source code, these editors are distinguished from text editors because work to either simplify or enhance the process of writing and editing of code for developers
* **Compiler:** Compilers transform source code that is written in a human readable/writable language in a form that computers can execute.
* **Debugger:** Debuggers are used during testing and can help developers debug their application programs.
* **Build automation tools:**These can help automate developer tasks that are more common to save time.

### 4.2.4 Install and configure IDE on desktop PC

We have already demonstrated how we can install Code::Blocks which is an example of an IDE. As mentioned, another IDE which can also be use is VSCode, Eclipse, Netbeans, Geany, Komodo IDE and CLion. There are several IDEs out there and usage depends on user or organisation preferences. These IDE allow you to run languages such as C, C++, Java, Fontran and many others.

Benefits of using IDEs

* Faster set up
* Faster development tasks
* Continual learning
* Standardisation

We are going to demonstrate how to create a Hello World using Code::Blocks IDE.

1. To start a project, select Code::Blocks icon from the desktop or just search for Code.

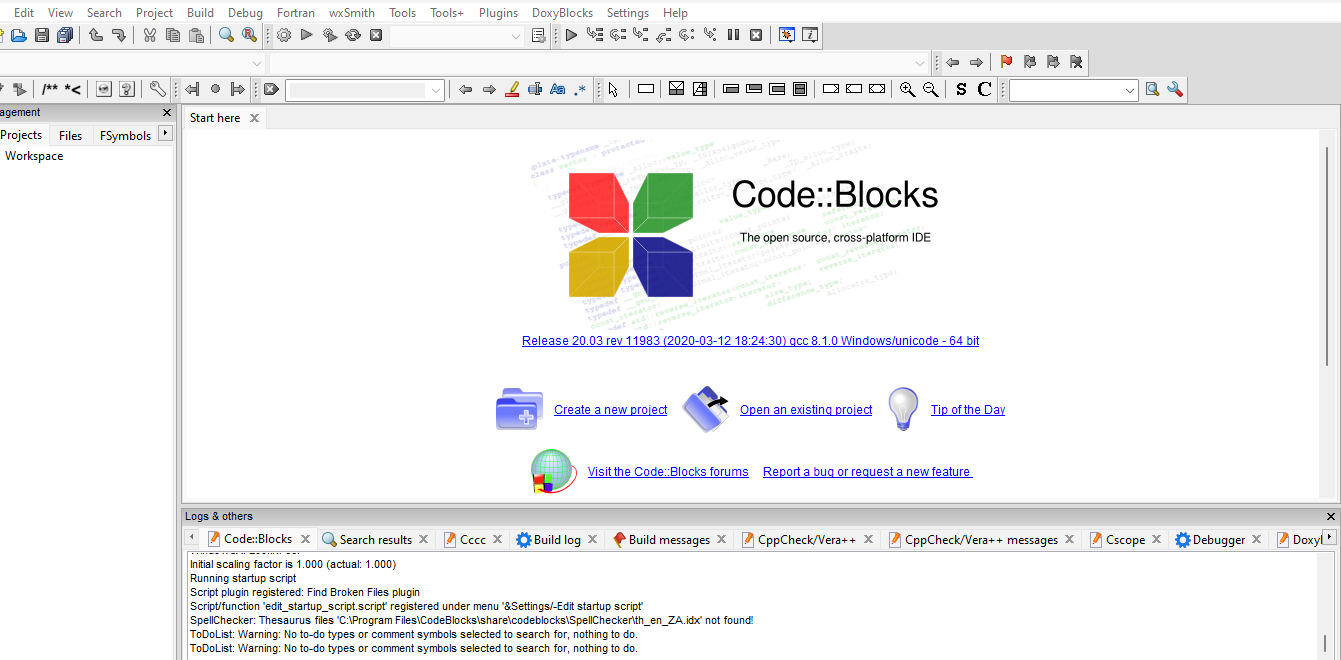


Figure 4. 6 : Code::Blocks Screen

1. Select create a new project and you will get a screen like the one on Figure 4.7 and select console application.

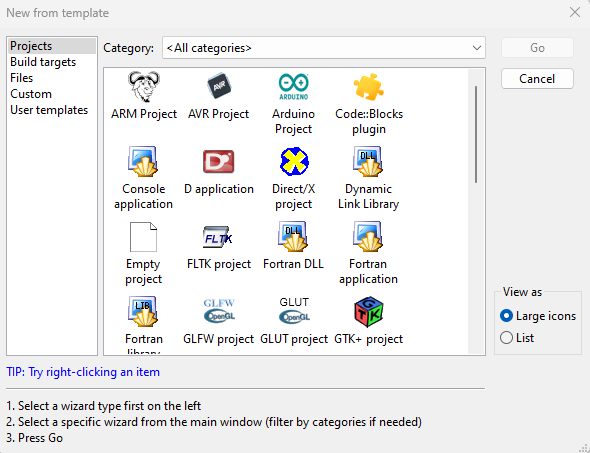


Figure 4. 7: Code::Blocks Template Window

1. Select the C++ programming language. Select the folder where the project will be saved and name of the project.
2. Navigate to the Projects workspace under the Management window and click the + sign on the Sources folder. This will open the main.cpp

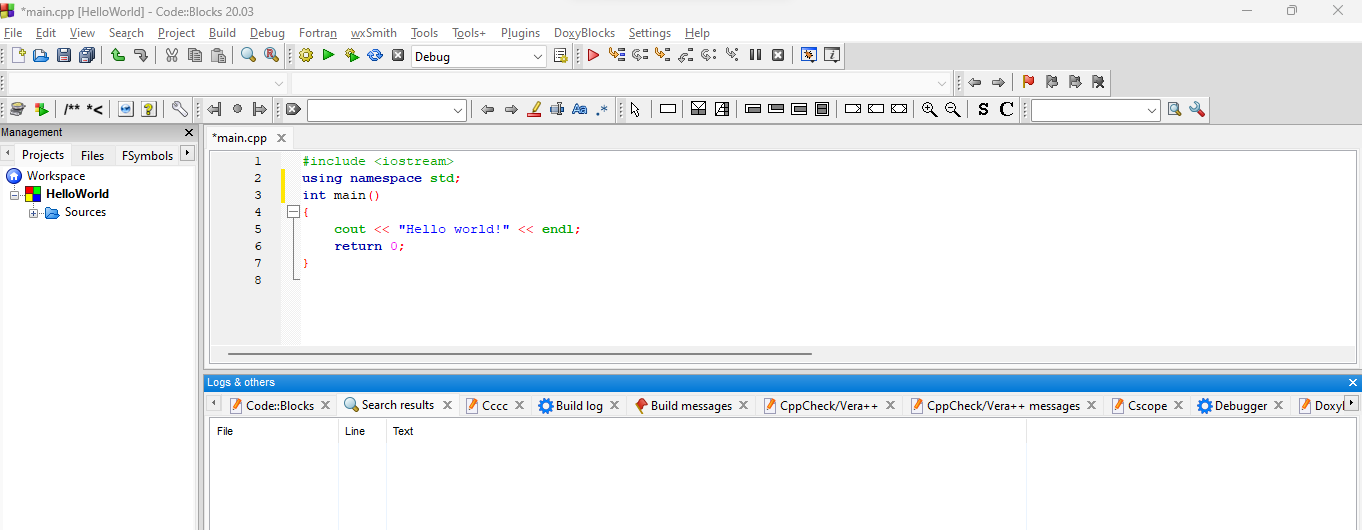


Figure 4. 8: Hello World sample project

To run the project, select build and click build and run. The sample output will be as shown in Figure 4.9.

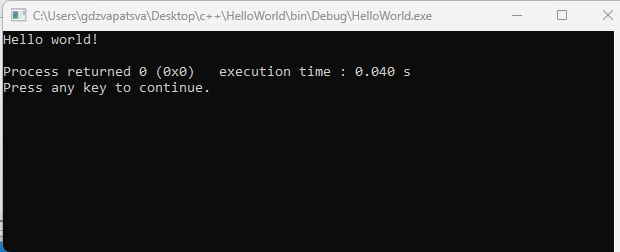


Figure 4. 9: Sample output of a C++ program

## 4.3 C/C++ on the Raspberry Pi

Content:

* The GCC C/C++ compiler
* Geany on Raspberry Pi

*Learning Outcomes:*

*Students should be able to:*

* + 1. Explain what the acronym GCC stands for
    2. Explain what the GCC collection contains
    3. List compilers included in the GCC
    4. Explain the difference between GCC and MinGW
    5. Install and configure GCC C/C++ compiler on Raspberry Pi
    6. Install and configure Geany on Raspberry Pi
  1. **Debugging C/C++ applications**

Content:

* Debugging
* GDB

*Learning Outcomes:*

*Students should be able to:*

* + 1. Define the term debugging
    2. Explain why an application needs to be debugged
    3. List common debugging techniques (Interactive, Print, Remote)
    4. Define the term breakpoint as it relates to debugging
    5. Define the term stepping as it relates to debugging
    6. Explain what the GDB tool is used for
    7. Install and configure MinGW on desktop PC
    8. Locate the GDB application in the MINGW installation
    9. Add the MINGW bin directory to the operating system path environment variable
    10. Compile source code wit the -g flag in IDE
  1. **Debugging C/C++ with GDBgui**

Content:

* GDBgui
* C/C++ debugging

*Learning Outcomes:*

*Students should be able to*

* + 1. Explain the concept of a debugger frontend
    2. Install and configure gdbgiu on Desktop PC
    3. Start gdbgui application
    4. Load a compiled C/C++ binary (.exe)
    5. Run the loaded binary
    6. Step through the running binary line by line
    7. Investigate (watch) the values of local variables
  1. **The Git Version Control System**

### 4.6.1 Introduction

There is often confusion between the terms *source control*, *revision control* and *version control*. The figure below provides a quick overview of source-code management. In level 2, we explained the differences by use of the following diagram shown in Figure 4:10.

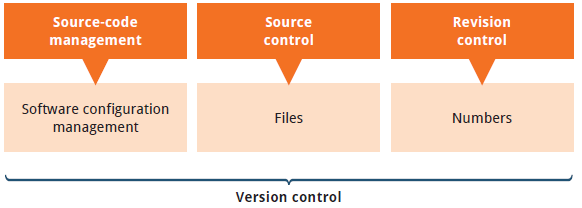
****

Figure 4. 10: Version control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.  For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

**Types of Version Control Systems:**

* Local Version Control Systems
* Centralized Version Control Systems
* Distributed Version Control Systems

**Local Version Control Systems- It is one of the most straightforward types and includes a database that maintains revision control over all file modifications.**

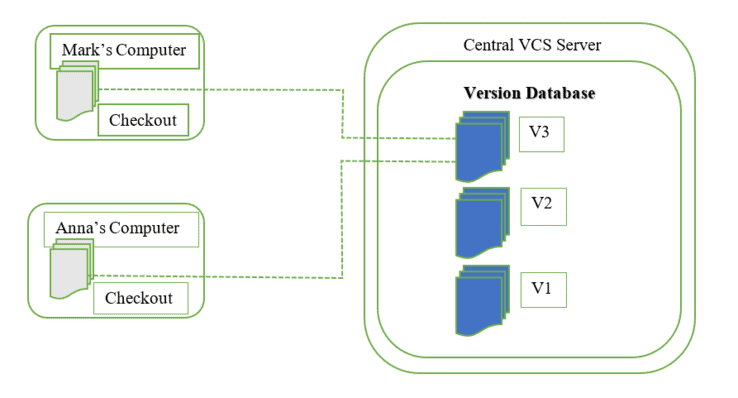
Diagram

Description automatically generated with medium confidence

Figure 4. 11: Local Version Control Systems

**In LVCS, every patch set contains only the changes made to the file since its last version. To see what the file looked like at any given moment; it is necessary to add up all the relevant patches to the file in order until that given moment.**

**Centralized Version Control Systems- These only have a single repository globally, and each user must commit for their modifications to be reflected in the repository. With CVCS, this way, everyone usually knows what everyone else on the project is doing. Administrators have control over who can do what which might be a downfall of CVCS. Figure 4.12 shows how CVCS looks like.**



**Distributed Version Control Systems (DVCS)-** Multiple repositories are present in distributed version control systems, and each user has a separate working copy and repository. You cannot share your modifications with others by simply committing them. This is due to the fact that commit just makes those changes accessible in your local repository; to make them visible on the main repository, you must push them. Git is an example of DVCS. Linus Torvalds created Git in 2005 for the development of the Linux kernel. Git allows multiple developers to work together and supports non-linear development through its thousands of parallel branches.

Graphical user interface, diagram, application

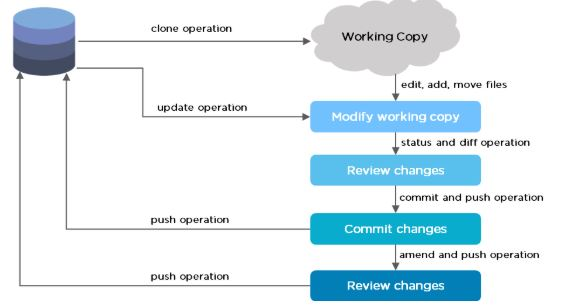
Description automatically generated

### 4.6.2 Defining Git is and what it is used for

According to Git website, Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easier and simpler to learn. Other similar software for version control includes Subversion, CVS, Perforce, and ClearCase. At the time of writing this book, Git was the commonly used VCS.

Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. Developers used to submit their codes to the central server without having copies of their own. Any changes made to the source code were unknown to the other developers. There was no communication between any of the developers. However, with Git, every developer has an entire copy of the code on their local systems. Any changes made to the source code can be tracked by others. There is regular communication between the developers. For Git to flows to be completed- there must be a hosting platform for the remote repositories. Git uses GitHub.

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Github is exclusively cloud based hosting service which enable programmers /users to manage Git repositories.



### Three main goals of the Git VCS

Git has three main goals that it attempts to achieve, and these are speed, data integrity and distributed.

**Speed**

Git handles all projects with speed irrespective of sizes. The speed might only be controlled by the network. Git servers are very efficient.

**Data integrity**

Before being stored, each item in Git is checksummed, and that checksum is subsequently used to identify the item. This means that no file or directory can have its contents changed without Git being aware of it. Git is fundamental to its ethos and has this functionality built in from the ground up.

**Distributed**

One of the nicest features of Git is that it's distributed. This means that instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository.

### 4.6.4 Relationship between local Git directory and directory located on server

Let us explain in simpler terms. When programmer(s) are working on a project, they save their file on the local machine. Once they initialise Git in that folder- the folder is known as the local repository. Any changes will need to be committed so that they are in the stagging area. I hope this explanation very succinct. Here is a what you need to understand with Git. Git has three main states that your files can reside in: modified, staged, and committed:

* Modified- means that you have changed the file but have not committed it to your database yet.
* Staged- means that you have marked a modified file in its current version to go into your next commit snapshot.
* Committed -means that the data is safely stored in your local database.



Figure 4. 12: Git workflow

### Discuss the software license used for Git

Open-source software is frequently shared via the public repositories on GitHub. You must grant people the right to freely use, modify, and distribute the software for your repository to be considered fully open source. There is no pressure on you to select a license. However, without a license, copyright rules that are in effect by default apply, which means that you maintain ownership of your source code and that no one else is permitted to copy, distribute, or create derivative works of your work. Most authors save the text of their license in a file with the name LICENSE.txt. There is a variety of licenses to choose from.

* + 1. **Software license used by Git**

Git’s open-source licensing has several advantages and disadvantages.

**Advantages of Git licensing**

1. **Flexibility and agility**- Technology agility is made possible by open-source licensing, which often provides several solutions to a given issue.
2. **Speed**- The flexibility to use the community versions, get started, determine whether they can solve your business problem, and start delivering value right away is a huge benefit of open-source licensing.
3. **Cost-effectiveness**- Open-source licensing is generally much more cost-effective.
4. **Attract better talent**- Most working engineers are familiar with open source, and many of them think it will be the future of their field. Attracts large community.
5. **Collaboration culture**

**Disadvantages of Git licensing**

1. Some individuals may find it less user-friendly because we are indirectly copying the source code.
2. Support issues can also go wrong from time to time.
3. The open-source license itself is free, however there are costs related to its plugin modules and a host of other things.
   * 1. **Explain why Git was created**

Git was created to resolve weaknesses of other version control systems for instance the centralised version control system. Git's branching features are among its greatest benefits. Git branches are inexpensive and simple to combine, in contrast to centralized version control solutions. This makes it easier to employ the feature branch workflow, which is well-liked by Git users.

* + 1. **Relationship between Git and hosting providers like GitHub**

Most people tend to confuse Git and GitHub.The key difference between Git and GitHub is that Git is an open-source tool developers install locally to manage source code, while GitHub is an online service to which developers who use Git can connect and upload or download resources. Git is a software and GitHub is a service. To make sure that the same copies are available with the remote repository, which is found on the GitHub server, the programmer will need to push the changes. You install Git locally on your machine while GitHub is hosted on the web.

* + 1. **List of major open-source projects using Git**

Since its birth, open software has fundamentally altered the digital ecosystem. On GitHub, there are several projects that developers are working on at the same time. Here is a list of open-source projects using Git.

* freeCodeCamp
* Flutter
* First Contributors
* TensorFlow
  + 1. **Install and configure Git on desktop**

The first important thing to do before intsalling git will be to check if its not installed already on your machine. Git uses commands. The command to run when checking for installed versions is git version or git - -version. These two commands produce the same result. Open command prompt and run the command.

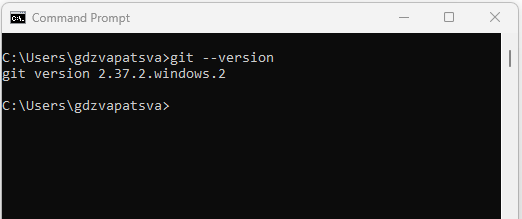


Figure 4. 13:Git checking

If Git is already installed, you will be able to see the version installed. If Git is not installed, you will get the following error message on your command prompt:

“[git' is not recognized as an internal or external command”](https://stackoverflow.com/questions/4492979/git-is-not-recognized-as-an-internal-or-external-command)

* + 1. **Install and configure Git on Raspberry Pi**
    2. **List and explain common terminology associated with distributed VCS’s including:**
       - Branch
       - Checkout
       - Clone
       - Stage
       - Commit
       - Conflict
       - Head
       - Repository
       - Initialise

1. To start this discussion, there are some key terms which we will need to understand. Some of the common git terms include:
2. Repository, (remote, local), push, pull request, clone, merge, origin, master, branch, reset, ignore and checkout.

Git has four main states in which files can be saved: working directory, staging area, local repository and central repository.

1. Working directory: where projects are created (code written) and changes are made
2. Staging area: where code can be reviews before making a commitment
3. Local repository: where you may commit changes to the project before pushing them to the central GitHub repository
4. Central repository: where the primary project is stored (a copy is kept locally by each team member).