Computational Techniques

Lecture: 4

Nitesh Kumar

nitesh.kumar@ddn.upes.ac.in
School of engineering
UPES, Dehradun

September 11, 2024

Agenda

1. Introduction
2. Syllabus
3. Unit 1
4. Linux Terminal and Commands
5. Editors
6. Computer Architecture and Organization

1 Introduction

2 Syllabus

3 Unit 1

4 Linux Terminal and Commands

5 Editors

6 Computer Architecture and Organization

Course Objectives

1. Learning fundamentals of operating system: Linux
2. Writing computer codes using vim editor in Linux
3. Elementary concepts of Python (Computer language)
4. Elementary concepts of technical writing using LaTeX
5. Plotting of graphs using `Gnuplot`
6. Data analysis using `Gnuplot`

1. Introduction
2. Syllabus
3. Unit 1
4. Linux Terminal and Commands
5. Editors
6. Computer Architecture and Organization

Syllabus

Units

1. Unit 1: Introduction to Linux and Scientific Computing : 7 lecture hours
2. Unit 2: Programming with Python : 11 lecture hours
3. Unit 3: Scientific word processing : 6 lecture hours
4. Unit 4: Data analysis and visualization : 6 lecture hours

Unit 1: Introduction to Linux and Scientific computing

1. Basics of Linux operating system
2. Linux commands
3. Vi and vim editors
4. Computer architecture and organization
5. Number system and
6. Floating point representation
7. Underflow and overflow
8. Errors in scientific computation

Unit 2: Programming with Python

1. Introduction to Python
2. Python literals, Operators, Variables
3. Loops and logic operations
4. Lists, functions, scopes in Python
5. Tuples and dictionaries
6. Modules and packages
7. Errors and Exceptions
8. Characters and strings
9. Basic concepts of Object oriented programming (oop)
10. Dealing with files
11. Numpy and Pandas

Unit 3: Scientific word processing

1. Introduction to LaTeX
2. Preparing a basic LaTeX file
3. Document classes and compiling a LaTeX file
4. Formula and equation representation
5. Preparing figures and tables
6. Preparing bibliography and citation

Unit 4: Data Analysis and visualization

1. Introduction to Gnuplot
2. Basic Gnuplot commands
3. Plotting data from a file
4. Saving and exporting data
5. Data analysis with Gnuplot

1 Introduction

2 Syllabus

3 Unit 1

4 Linux Terminal and Commands

5 Editors

6 Computer Architecture and Organization

Introduction to Linux and Scientific computing

Computer

– A machine that ‘process’ information:

1. Store the data
2. Retrieve the data
3. Process the data

Examples: Laptop, Desktop, Servers, Mobiles, Tablets, Smart wearable, etc.

Parts of computers

Hardware

– Physical components of a computer
e.g. CPU, Monitor, Keyboard, Mouse, etc.

Software

– Programs that run on a computer
e.g. Operating system, Application software, etc.

Operating System

Operating System

– A software that manages the hardware and software resources of a computer.

Functions

1. Memory management: RAM, ROM, Cache
2. Processor management: CPU, Cores, Threads
3. Device management: Input, Output, Storage
4. File management: File system, Directories
5. Security: User authentication, Data protection, etc.

Operating Systems

Types of Operating Systems

1. Windows : Microsoft (PC) [Date of release: 1985]
2. Mac OS : Apple (Mac) [Date of release: 1984]
3. Linux : Open source (PC and Servers) [Date of release: 1991]
4. Unix : Bell Labs (Servers) [Date of release: 1969]
5. Android : Google (Mobiles) [Date of release: 2008]
6. iOS : Apple (Mobiles) [Date of release: 2007]
7. Chrome OS : Google (Chromebooks) [Date of release: 2009]

Advantages of Linux

Advantages

1. Free and open source: No license fee
2. Secure: Less prone to virus attacks
3. Stable: Less prone to crashes
4. Customizable: Can be modified as per requirement
5. Supports multiple users: Multiple users can work on a single system
6. Supports multiple file systems: Ext4, NTFS, FAT32, etc.
7. Supports multiple programming languages: C, C++, Python, Java, etc.

History of Linux Operating System

History

1. Developed by Linus Torvalds in 1991
2. Based on Unix operating system
3. Open source: Free to use and modify
4. Supports multiple platforms: PC, Servers, Embedded systems, etc.
5. Supports multiple file systems: Ext4, NTFS, FAT32, etc.
6. Supports multiple programming languages: C, C++, Python, Java, etc.

Linux Distribution

Distribution

– A collection of software that is based on the Linux kernel.

Examples

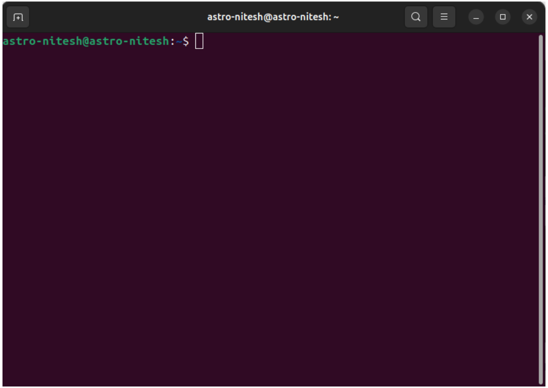
1. Ubuntu : Developed by Canonical Ltd.
2. Fedora : Developed by Red Hat Inc.
3. Debian : Developed by Debian Project
4. CentOS : Developed by CentOS Project
5. Arch Linux : Developed by Arch Linux Project
6. Kali Linux : Developed by Offensive Security

1. Introduction
2. Syllabus
3. Unit 1
4. Linux Terminal and Commands
5. Editors
6. Computer Architecture and Organization

Linux Terminal

Terminal

– A command line interface to interact with the operating system.



Linux Commands

Basic Syntax

```
$ command [options] [arguments]
```

Commands

1. `ls` : List files and directories
2. `cd` : Change directory
3. `pwd` : Present working directory
4. `cp` : Copy files
5. `mv` : Move files
6. `rm` : Remove files
7. `mkdir` : Make directory

Try these commands

Exercise

1. Open the terminal
2. Type the command: `ls`
3. Type the command: `pwd`
4. Type the command: `cd` Downloads
5. Type the command: `mkdir` MyFolder
6. Type the command: `cp` file1 file2
7. Type the command: `mv` file2 MyFolder/
8. Type the command: `rm` file1

Instructions

1. Download the quiz using the QR code.
2. Solve the quiz by typing the commands in the terminal.
3. Use Google to find the answers if you are stuck.
4. Try it on your machines or use online terminals (e.g. JSLinux).
5. 10 Minutes time to solve the quiz.
6. Extra points for attempting Advanced questions.

Linux Quiz

1 Introduction

2 Syllabus

3 Unit 1

4 Linux Terminal and Commands

5 Editors

6 Computer Architecture and Organization

Editors in Linux

Editors

– A software to write and edit text files, codes, and scripts.

Types

1. Command line editors: Vi, Vim, Nano.
2. Graphical editors: Gedit, Kate, Emacs.

History of Vi and Vim Editors

- **Vi Editor:**
  - Developed by Bill Joy in 1976.
  - Part of the Berkeley Software Distribution (BSD).
  - Originally designed for UNIX systems.
- **Vim Editor:**
  - Created by Bram Moolenaar in 1991.
  - Stands for "Vi IMproved".
  - Aimed to add more features to the original Vi editor.

Importance of Vi and Vim Editors

- **Efficiency:**
  - Highly efficient for text editing, especially for programmers.
  - Commands are optimized for speed and minimal keystrokes.
- **Universality:**
  - Available by default on most UNIX/Linux systems.
  - Vi is a standard editor in many environments.
- **Extensibility (Vim):**
  - Vim supports plugins, scripts, and custom configurations.
  - Highly customizable for various use cases.

Modes of Operation in Vi and Vim Editors

There are Three modes of operation in Vi and Vim editors:

Modes

1. **Command Mode:** Default mode for navigation and editing.
2. **Insert Mode:** For inserting text into the file.
3. **Last Line Mode:** For executing commands and saving files.

1. Command Mode

1. vi starts in Command Mode.
2. Interprets characters as commands (not displayed).
3. Allows navigation, deletion, copying, and pasting of text.
4. Press [Esc] to enter Command Mode from any other mode.
5. Pressing [Esc] in Command Mode will beep or flash the screen.

2. Insert Mode

1. Enables insertion of text into the file.
2. Everything typed in this mode is interpreted as input.
3. vi always starts in Command Mode.
4. To enter text, you must be in Insert Mode.
5. To enter Insert Mode, type `i`.
6. Press [Esc] to exit Insert Mode and return to Command Mode.

3. Last Line Mode

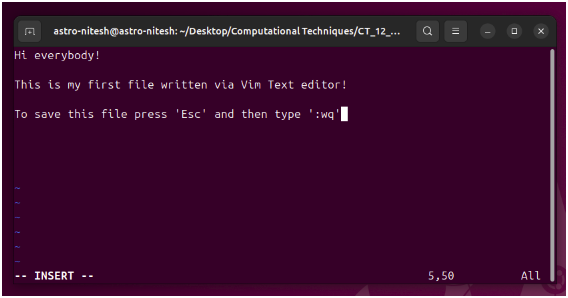
1. Invoked by typing a colon [:].
2. Cursor jumps to the last line of the screen.
3. vi waits for a command in this mode.
4. Enables tasks like saving files and executing commands.
5. To save a file, type `:w` and press [Enter].
6. To quit vi, type `:q` and press [Enter].

Vim Editor

Vim

– A text editor based on Vi editor. To open a file in vim editor:

```
$ vim my_first_file.txt
```



Usage of Vi and Vim Editors

- **Basic Commands:**
  - `:w` - Save file
  - `:q` - Quit editor
  - `dd` - Delete a line
  - `yy` - Yank (copy) a line
  - `p` - Paste text
- **Advanced Features (Vim):**
  - `:split` - Split window
  - `:vsplit` - Vertical split
  - `:!command` - Execute shell commands
  - Plugin support for additional functionalities.

Exercise

1. Open a file in Vim editor: `$ vim my_first_file.txt`
2. Type some text in the file.
3. Save the file: Press [Esc], then type `:w` and press [Enter].
4. Quit the editor: Press [Esc], then type `:q` and press [Enter].
5. Open the file again and try other commands.
6. Get the Vim cheat sheet for more commands and try them.

1 Introduction

2 Syllabus

3 Unit 1

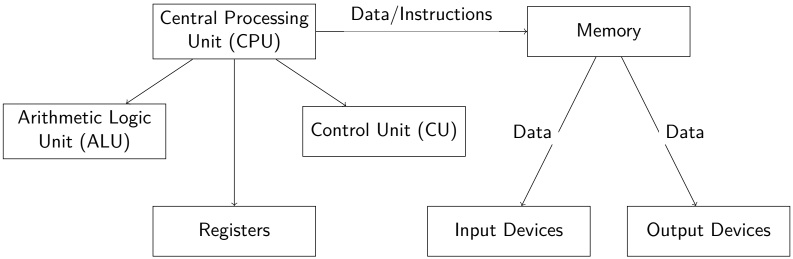
4 Linux Terminal and Commands

5 Editors

6 Computer Architecture and Organization

Computer Architecture and Organization

Computer Architecture Block Diagram



How Computers Store Information

- **Binary System:**
  - Computers use the binary number system (0s and 1s) to represent all types of data.
  - A bit (binary digit) is the smallest unit of data in a computer.
  - 8 bits form a byte, which can represent 256 different values.
- **Memory Hierarchy:**
  - **Registers:** Small, fast storage areas in the CPU used for temporary data.
  - **Cache:** Intermediate storage between registers and main memory, faster than RAM.
  - **Main Memory (RAM):** Stores data and instructions that the CPU needs in real-time.
  - **Secondary Storage:** Non-volatile storage like hard drives, SSDs, where data is stored long-term.
- **Data Representation:**
  - **Characters:** Stored using encoding schemes like ASCII or Unicode.
  - **Numbers:** Stored as integers or floating-point representations.
  - **Images and Sound:** Stored as a series of bits representing pixel values or sound waveforms.
- **File Systems:**

Introduction to Number Systems in Computing

- **Number Systems:**
- **Binary System:**
  - Base-2 number system.
  - Uses digits 0 and 1.
  - Essential for representing data in digital computers.
- **Other Systems:**
  - **Decimal (Base-10):** Commonly used by humans.
  - **Hexadecimal (Base-16):** Often used in programming and debugging.
  - **Octal (Base-8):** Sometimes used in computing as a shorthand for binary.

Binary Number System

- **Base-2 System:** Each bit represents a power of 2.
- **Binary Digits (Bits):**
  - 0 and 1.
  - Example: $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$.
- **Significance:**
  - Basis for all modern digital systems.
  - Used in memory storage, data processing, and communication.

Hexadecimal and Octal Number Systems

- **Hexadecimal (Base-16):**
  - Digits: 0-9 and A-F (where A=10, B=11, ..., F=15).
  - Example: $A3_{16} = 10 \times 16^1 + 3 \times 16^0 = 163_{10}$.
  - Used to represent large binary numbers succinctly.
- **Octal (Base-8):**
  - Digits: 0-7.
  - Example: $57_8 = 5 \times 8^1 + 7 \times 8^0 = 47_{10}$.
  - Often used in digital systems as a shorthand for binary.

Conversion Between Number Systems

- **Binary to Decimal:**
  - Example: $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$.
- **Decimal to Binary:**
  - Example: $13_{10} = 1101_2$ (Divide by 2 and track remainders).
- **Hexadecimal to Binary:**
  - Example: $A3_{16} = 1010_2 0011_2$.
- **Octal to Binary:**
  - Example: $57_8 = 101_2 111_2$.

Importance of Number Systems in Computer Architecture

- **Data Representation:**
  - Numbers, characters, and instructions are represented in binary.
- **Memory Addressing:**
  - Memory locations are often referred to using hexadecimal numbers.
- **Instruction Sets:**
  - Instructions are encoded in binary to be processed by the CPU.
- **Efficient Computation:**
  - Binary arithmetic is faster and simpler to implement in hardware.

Introduction to Floating-Point Representation

- **Floating-Point Numbers:** Used to represent real numbers in a form that can support a wide range of values.
- **Components:**
  - **Sign (S):** Indicates positive or negative number.
  - **Exponent (E):** Determines the scale (magnitude) of the number.
  - **Mantissa (M) or Significand:** Represents the precision of the number.
- **IEEE 754 Standard:**
  - Commonly used format for floating-point representation.
  - Single precision (32-bit) and double precision (64-bit) formats.

Example of Floating-Point Representation

- **Example:** Representing the decimal number -6.25 in IEEE 754 single precision.
- **Step 1: Convert to Binary**
  - $6.25_{10} = 110.01_2$.
  - Sign bit $S = 1$ (negative number).
- **Step 2: Normalize the Binary Number**
  - $110.01_2 = 1.1001_2 \times 2^2$.
- **Step 3: Determine Exponent**
  - Exponent $E = 127 + 2 = 129$ (in bias-127 form).
  - Binary of $129 = 10000001_2$.
- **Step 4: Form the Mantissa**
  - Mantissa $= 10010000000000000000000_2$ (23 bits).
- **Final IEEE 754 Representation:**
  - $1\ 10000001\ 10010000000000000000000_2$.
  - In hexadecimal: $C1C80000_{16}$.

Example of Floating-Point Representation (Double Precision)

- **Example:** Representing the decimal number -6.25 in IEEE 754 double precision.
- **Step 1: Convert to Binary**
  - $6.25_{10} = 110.01_2$.
  - Sign bit $S = 1$ (negative number).
- **Step 2: Normalize the Binary Number**
  - $110.01_2 = 1.1001_2 \times 2^2$.
- **Step 3: Determine Exponent**
  - Exponent $E = 1023 + 2 = 1025$ (in bias-1023 form).
  - Binary of $1025 = 10000000001_2$.
- **Step 4: Form the Mantissa**
  - Mantissa $= 1001000000000000000000000000000000000000000000000000_2$ (52 bits).
- **Final IEEE 754 Representation:**
  - $1\ 10000000001\ 1001000000000000000000000000000000000000000000000000_2$.

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases I

- **IEEE 754 Single Precision Format:**
  - 1 bit for sign (S)
  - 8 bits for exponent (E) with bias 127
  - 23 bits for mantissa (M) + implicit leading 1

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases II

- **Maximum Number:**
  - Exponent: $E = 254$
  - Actual exponent: $254 - 127 = 127$
  - Largest mantissa: $M = 1 + (1 - 2^{-23})$
  - Maximum value:
  $$(1.99999988) \times 2^{127} \approx 3.4028235 \times 10^{38}$$
  - Hex representation: $7F7FFFFF_{16}$

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases III

- **Minimum Normalized Positive Number:**
  - Exponent: $E = 1$
  - Actual exponent: $1 - 127 = -126$
  - Smallest normalized value:

$$1.0 \times 2^{-126} \approx 1.17549435 \times 10^{-38}$$

  - Hex representation: 00800000_{16}

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases IV

- **Smallest negative number:**
  - Exponent: $E = 0$
  - Mantissa: $M = 0$
  - Smallest negative value:

$$-0.0 \times 2^{-126} \approx -1.17549435 \times 10^{-38}$$

  - Hex representation: 80800000_{16}

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases V

- **Special Values:**
  - **Zero:**
    - **Positive Zero:**

      $00000000000000000000000000000000_2$
    - **Negative Zero:**

      $10000000000000000000000000000000_2$
  - **Infinity:**
    - **Positive Infinity:**

      $01111111100000000000000000000000_2$
    - **Negative Infinity:**

      $11111111100000000000000000000000_2$

IEEE 754 Single Precision: Maximum, Minimum, and Special Cases VI

- **NaN (Not a Number):**
  - Exponent $= 255$ (all 1's) and mantissa $\neq 0$
  - **Example:**

$$01111111110000000000000000000000_2 \text{ (quiet NaN)}$$

Overflow, Underflow, and Precision in IEEE 754 Single Precision I

- **Overflow:**
  - Occurs when a calculation produces a number larger than the maximum representable value.
  - Example:

$$\text{Max representable} = (1.99999988) \times 2^{127}$$

  - If you try to represent 2^{128} or higher, it exceeds the maximum value and results in infinity:

$$2^{128} \text{ is represented as } \infty$$

Overflow, Underflow, and Precision in IEEE 754 Single Precision II

- **Underflow:**
  - Occurs when a calculation produces a number smaller than the smallest normalized positive value.
  - Example:

$$\text{Smallest normalized value} = 1.17549435 \times 10^{-38}$$

  - If you try to represent a number smaller than 1.4×10^{-45}, it becomes a subnormal number or zero:

$$1.4 \times 10^{-45} \text{ is the smallest subnormal positive number}$$

Overflow, Underflow, and Precision in IEEE 754 Single Precision III

- **Precision:**
  - Precision refers to the number of significant bits used to represent a number.
  - In single precision, 23 bits are used for the mantissa, providing about 7 decimal digits of precision.
  - Example:

$$\text{Number } 0.1_{10} \text{ in binary } = 0.00011001100110011001101_2$$

  - The closest representable value in single precision may be slightly different from the exact decimal representation.

Thank you for listening !

Nitesh Kumar

nitesh.kumar@ddn.upes.ac.in