# Operating System Lab: CS341

# LAB 1: Linux Basics


**Course Instructor: Dr Satendra Kumar**

# Basic Linux Commands

- **mkdir:** The **mkdir** command in Unix-like operating system is used to create new directories.

**Syntax:**

- In order to create a single directory the syntax is:

  **mkdir directory_name**

- In order to create multiple directories the syntax is:

  **mkdir directory1_name directory2_name directory3_name**

- In order to create a directory with necessary parent directories, we use -p option.

  **mkdir -p parent_directory/sub_directory**

- **touch:** The touch command creates an empty file if it does not already exist, or updates the access and modification times if it does.

  **Syntax:** **touch filename**

- **mv:** The mv command in Linux is used to move or rename files and directories.

- To move a file to a different directory:

  **Syntax:** **mv file1.txt /path/to/destination/**

- To rename a file:

  **Syntax:** **mv oldname.txt newname.txt**

- To move a file to a different directory and rename it at the same time

  **Syntax:** **mv file1.txt /path/to/destination/newname.txt**

➢ **cp:** The cp command in Linux is used to copy files and directories.

- To copy a file to another location:
  **Syntax:** **cp file1.txt /path/to/destination/**

- To copy a file and rename it at the same time:
  **Syntax:** **cp file1.txt /path/to/destination/newname.txt**

- To copy multiple files to another directory:
  **Syntax:** **cp file1.txt file2.txt /path/to/destination/**

- To copy a directory and its contents, you need to use the -r (recursive) option:
  **Syntax:** **cp -r dir1 /path/to/destination/**

- **ls:** The ls command in Linux is used to list the contents of a directory.

  **Synatx:** **ls –R**

  Lists files in all subdirectories recursively.

- **read:** The read command in Linux is used to take user input and assign it to a variable. It reads a single line of input from standard input (keyboard) or a file.

  **Syntax:** **read name**

  The -p option allows you to provide a prompt message before reading the input.

  **Syntax:** **read -p "Enter your name: " name**

- To check whether a given path is not a directory the following syntax is used:

**Syntax:** **if [ ! -d "path" ]; then**
**# Commands to execute if the path is not a directory**
**fi**

- **echo:** Prints a message to the standard output.
**Syntax:** **echo “Message”**

- **for loop:** Loops through a list of items and executes commands for each item.

**Syntax: for item in list; do**
**# Commands to execute for each item**
**done**

- To append output to a file.
**Syntax: echo "Message" >> filename**

- **rm:** Removes files or directories.

  **Syntax:** **rm file_name or rm directory_name**

- **if-else:** Conditional statement based on the value of a variable.

  **Syntax:** **if $variable; then**

  **# Commands to execute if variable is true**

  **else**

  **# Commands to execute if variable is false**

  **fi**

- **exit:** Exits the script with a specified status code.

  **Syntax:** **exit status_code**

- To check if the number of arguments passed to the script is not equal to 1.

  **Syntax:**
  ```
  if [ "$#" -ne 1 ]; then
      # Commands to execute if the number of arguments is not 1
  fi
  ```

- **ps:** Displays information about active processes.

  **Syntax:** `ps [options]`

- **ps aux:** Displays all processes for all users in a user-oriented format.

  **Syntax:** `ps aux`

- **sort:** Sorts the output of the ps command by a specified field.

  **Syntax:** `ps aux --sort=-%cpu`

- **awk:** The awk command is a powerful text-processing tool in Unix and Unix-like operating systems. It is used for pattern scanning and processing. awk reads input files line by line, searches for patterns, and performs actions on lines that match the patterns.

**Syntax:** **awk '{ print $1, $2 }' input_file**

By default, awk splits each line into fields based on whitespace (spaces or tabs). Fields can be accessed using $1, $2, etc.

**Fileds Specifications:**

1. **$11**: Prints the 11th field of the input line. In the context of ps aux, this is typically the command or process name.
2. **$2**: Prints the 2nd field of the input line. This is the process ID (PID).
3. **$3**: Prints the 3rd field of the input line. This is the CPU usage percentage.
4. **$4**: Prints the 4th field of the input line. This is the memory usage percentage.

- **df:** The df command reports the amount of disk space used and available on file systems.

  **Syntax:** **df -h /**

  Displays the disk usage of the root directory / in a human-readable format.

- **free:** The free command displays information about the system's memory usage.

  **Syntax:** **free –h**

  Shows the free, used, and total memory available on the system in a human-readable format.