Module-0 Training - Lab Manual: Linux Shell Scripting and Development Tools

This lab manual will guide you through the essential tools for Linux development: Shell Scripting, Vim Editor, VS Code, Git, and Makefiles.

# Setting Up Your Development Environment (Choose One)

## Install Linux:

* Use the [following resources](https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview) to guide you through the installation process

## Use Windows Subsystem for Linux (WSL):

* This allows you to run a Linux environment within Windows.
* Follow the [official Microsoft guide](https://learn.microsoft.com/en-us/windows/wsl/install) to set up WSL

**Note:** Throughout the labs, commands might differ slightly between pure Linux and WSL. The Linux commands will be provided in the other resources, WSL users might need to put extra efforts to make those commands run on their machines.

# Linux Shell Scripting

The shell is your command center in Linux. Shell scripting allows you to automate repetitive tasks.

## Learning Resources:

* Microsoft Learning Path: [click here](https://learn.microsoft.com/en-us/training/paths/shell/)
* MIT OpenCourseware: ([Link-1](https://missing.csail.mit.edu/2020/course-shell/), [Link-2](https://missing.csail.mit.edu/2020/shell-tools/)). Focuses on practical exercises
* Test operators in Bash: ([available here](https://linuxhandbook.com/bash-test-operators/))

## Exercises:

* **Basic Shell Commands:**
  + Practice navigating directories (cd), listing files (ls), and creating/removing files/directories (mkdir, rm, touch, rmdir).
* **File Manipulation:**
  + Use cat to view files, head/tail to see the beginning/end, and cp/mv to copy/move files.
* **Working with Text Files:**
  + Learn commands like grep to search for text patterns and cut/paste to manipulate text data.
* **Shell Scripting Basics:**
  + Write a simple script to automate a task, like printing a message or backing up a directory. Learn about variables, loops (for, while), and conditional statements (if, else).
* Explore resources like <https://missing.csail.mit.edu/> for guided exercises.

# Vim Editor

Vim is a powerful text editor popular for its efficiency and customization options. Learning Vim can be challenging but rewarding.

## Learning Resources:

* MIT OpenCourseware: [course link](https://missing.csail.mit.edu/2020/editors/)

## Exercises:

* **Basic Vim Navigation:**
  + Learn to move around the editor using the h, j, k, l keys and understand modes (Insert, Command).
* **Text Editing:**
  + Practice basic editing commands like deleting, copying, and pasting within Vim.
* **Saving and Exiting:**
  + Understand the concepts of saving files (:w) and quitting Vim (:q).
* Explore features like search (:?text), regular expressions, recording macros, and plugins.

**Tip:** Many users prefer starting with a more beginner-friendly editor like VS Code and gradually learning Vim later but learning Vim is rewarding for functional development of softwares.

# VS Code

VS Code is a popular, versatile code editor with excellent support for various programming languages and tools.

## Learning Resources:

* VS Code Installation: [click here](https://code.visualstudio.com/docs/setup/linux)
* VS Code Introduction Videos: [click here](https://code.visualstudio.com/docs/introvideos/basics)

## Exercises:

* **Setting Up VS Code:**
  + Install VS Code for your chosen platform (Linux/WSL) and explore the interface.
* **Shell Integration:**
  + Learn to run shell commands directly from the VS Code terminal.
* **Extensions:**
  + Install extensions for specific languages like Python or Git to enhance functionality.

**Tip:** During the Git section, you'll learn how to integrate Git with VS Code for a seamless workflow.

# Git

Git is a version control system that allows you to track changes in your code, collaborate with others, and revert to previous versions if needed.

## Learning Resources:

* Microsoft Learning Path: [click here](https://learn.microsoft.com/en-us/training/modules/intro-to-git/)
* MIT OpenCourseware: [click here](https://missing.csail.mit.edu/2020/version-control/)
* Setting Up SSH Keys for Secure Communication: [click here](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account)
* Using Git with VS Code: [click here](https://code.visualstudio.com/docs/sourcecontrol/intro-to-git)

## Exercises:

* **Git Basics:**
* Learn core Git commands like init, add, commit, push
* **Branching and Merging:**

1. **Create a Feature Branch:**
   * Create a new branch for a specific feature, make modifications, commit those changes, and then merge the branch back into the main branch.
2. **Resolving Merge Conflicts:**
   * Simulate a merge conflict scenario where changes were made to the same line of code in different branches. Guide them through manually resolving the conflict and committing a successful merge.
3. **Remote Repositories:**
   * **Clone a Public Repository:**
     + Have trainees clone an existing public repository from a platform like GitHub. This helps them understand how to work with remote codebases.
   * **Contributing to a Public Repository:**
     + Fork an existing public repository, make changes locally, push those changes to their forked repository, and then create a pull request for the original maintainers to review. This teaches them the contribution workflow.
4. **Understanding Git Stash:**
   * Introduce the git stash command that allows temporarily saving uncommitted changes. Practice using stash to keep your working directory clean.
5. **Ignoring Files/Directories:**
   * Create a .gitignore file to specify files or directories that should be excluded from version control (e.g., compiled files, configuration files).
6. **Git Tags:**
   * Demonstrate using Git tags to mark specific points in the development history. Trainees can create tags to signify project milestones or releases.

**Additional Tips:**

* **Visualizing Git Workflow:** Create diagrams or animations to illustrate Git concepts like branching and merging. It will help you visualizing the essence of version control.
* **Real-world Scenarios:** Think about practical use cases for Git, like collaborating on a group project or managing different versions of your own code.
* **Interactive Learning:** Utilize online Git tutorials with interactive exercises to solidify their understanding.