Module-1 Training - Lab Manual: C Programming, Makefiles, and TCL Scripting

This lab manual delves into C programming, Makefiles for automating compilation, and TCL scripting for scripting tasks. This lab manual provides a foundational understanding of C programming, Makefiles, and TCL scripting. Remember, consistent practice and exploration of additional resources are key to mastering these skills.

# C Programming

C is a powerful and fundamental programming language known for its efficiency and control over hardware interactions.

## Learning Resources:

* The C Programming Language by Kernighan & Ritchie ([available here](https://pern-my.sharepoint.com/:b:/g/personal/meds_uet_edu_pk/EWeK7XdzgrVFrVWS0ieNZlkB6LoOyWm5QGFpAv6MG8zb6Q?e=cMmL5f))
* C Tutorial – Tutorialspoint ([available here](https://www.tutorialspoint.com/cprogramming/index.htm))

## Exercises:

1. **Basic C Syntax:**
   * Practice writing simple C programs that involve variables, data types (int, float, char), operators (arithmetic, assignment, comparison), and control flow statements (if-else, for, while).
2. **Functions and Arrays:**
   * Learn to define functions for modularity and reuse code. Practice working with arrays to store collections of data.
3. **Pointers and Memory Management:**
   * Understand the concept of pointers and how they reference memory locations. This is a crucial but sometimes challenging aspect of C.
4. **Input/Output (I/O):**
   * Learn to use standard input/output functions like printf and scanf to interact with the user.
5. **Compiling C Programs:**
   * Use the gcc compiler to compile your C code. Explore options like -Wall for warnings and -o to specify the output filename.

**Tip:** Start with basic concepts and gradually progress to more complex topics like pointers and memory management. Utilize online resources with interactive exercises to solidify your learning.

# Makefiles

Makefiles are configuration files that automate the compilation process for software projects. They specify dependencies between source files and how to generate the final executable.

## Learning Resources:

* Hand Book: ([available here](https://linuxhandbook.com/using-make/))
* The GNU Make Manual: ([available here](https://www.gnu.org/s/make/manual/make.html))
* Makefile Tutorial: ([available here](https://makefiletutorial.com/))
* MIT Make File Guide: ([available here)](https://web.mit.edu/gnu/doc/html/make_2.html)

## Exercises:

1. **Basic Makefile Structure:**
   * Understand the syntax of Makefiles, including target definitions, prerequisites, and recipes (commands to be executed).
2. **Compiling C Programs with Makefiles:**
   * Create a Makefile that specifies how to compile your C code using gcc. Define targets for compiling individual source files and building the final executable.
3. **Dependencies and Automatic Builds:**
   * Learn how to leverage dependencies in Makefiles to ensure only modified files are recompiled during subsequent builds.
4. **Advanced Makefile Features:**
   * Explore features like wildcards, variables, and conditional statements for more complex build scenarios.

**Tip:** Start with simple Makefiles for single-file C programs and gradually add complexity as you understand the concepts. Online tutorials with examples can be helpful for practicing Makefile creation.

# TCL Scripting

TCL (Tool Command Language) is a scripting language used for automating tasks, building user interfaces, and extending applications.

## Learning Resources:

* TCL Programming Guide: ([click here](https://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html))
* Learn TCL Scripting - Tutorialspoint: ([click here](https://www.tutorialspoint.com/tcl-tk/index.htm))

## Exercises:

1. **Basic TCL Syntax:**
   * Learn the basic syntax of TCL, including variables, data types (strings, lists), commands, and control flow statements (if, while, for).
2. **Procedures and Packages:**
   * Understand how to define procedures (functions) for code reuse and organize code into packages for modularity.
3. **Working with Strings:**
   * Practice manipulating strings in TCL using built-in commands for searching, concatenation, and formatting.
4. **TCL for Automation:**
   * Explore how to use TCL for automating tasks like file manipulation, system calls, and interacting with other applications.

**Tip:** TCL has a gentle learning curve. Start with basic commands and procedures, then progress to more advanced concepts like packages and automation. Many online resources with interactive examples can help you practice.