The Final Assignment

Author's Name: Taha Biklaryan

Student ID: 401411318

Dr. Malekimajd

Computer Workshop

1 Git and Github

1.1 Repository Initialization and Commits

For creating a repository for this project, first i opened my account and then in the repository section of the github i created a repository and named it "Workshop-Final-proj" and then cloned the repository into a local directory of my pc and synced it with my vscode where i'm editing this assignment right now and going to commit the changes so far right now after putting the dot at the end of this sentence.

1.2 Github Actions for LaTeX Compilation

After setting up the repository in github i created a main.tex that i'm constantly commiting changes there and i created folder .github which is workflow to i build and release my LaTeX files through github Actions. And after each changes in the text editor i commit the changes to the main.tex file via vscode.

2 Exploration Tasks

2.1 Vim Advanced Features

1. Sessions: Vim sessions allow you to save and restore your working environment, including open files, cursor positions, and window layouts. You can create a session with ':mksession' and then later restore it using ':source Session.vim'. This is particularly useful when you are working on a project and want to quickly resume your work with the same setup.
2. Marks And Jumps: Vim allows you to place marks in your file, which act as bookmarks. You can then jump between these marks using various commands. For example, you can use 'ma' to mark a position with the label 'a', and then use "a' to jump back to that position. This is handy for navigating large files and remembering specific locations.
3. Text Objects: Vim provides a variety of text objects that allow you to operate on different parts of text efficiently. For instance, 'iw' represents "inner word," and 'i(' represents "inner parentheses." These text objects can be combined with commands to perform powerful operations on specific chunks of text.

2.2 Memory Profiling

2.2.1 Memory Leak

Memory leaks occur when a computer program allocates memory but fails to release or deallocate it properly, leading to a gradual consumption of system resources. In simpler terms, memory leaks occur when a program reserves space in the computer's memory but forgets to free up that space when it's no longer needed. This can result in a gradual decrease in available memory, which can eventually lead to performance issues or even program crashes.

In these cases this might happen:
1. Unintentional Memory Allocation:
If you dynamically allocate memory using functions like malloc() in C or new in C++, you must remember to free that memory using free() or delete respectively. Forgetting to do so can lead to memory leaks.
2. Not Releasing Resources:
Memory leaks are not limited to just memory allocation. Other resources like file handles, network connections, or database connections also need to be released when no longer needed. Failure to release these resources can lead to memory leak.
3. Circular References
4. Global Variables and Static Data
and there are other reasons that this might happen

2.2.2 Memory Profilers

- Introduction:
  The Valgrind tool suite provides a number of debugging and profiling tools that help you make your programs faster and more correct. The most popular of these tools is called Memcheck. It can detect many memory-related errors that are common in C and C++ programs and that can lead to crashes and unpredictable behaviour.It is mainly an open-source programming tools designedd for memory debugging, memory leak detection, and profiling. Valgrind provides a framework for instrumenting and analysing the memory usage of a program, helping developers identify and fix memory related issues.

- Memory Leaks Detection:
  Valgrind can detect memory leaks by keeping track of memory allocations and deallocations during the execution of a program. If memory is allocated but not deallocated (resulting in a memory leak), Valgrind can identify the source of the leak and provide information to help developers fix the issue.
  steps:
  1. Valgrind instruments the binary code of a program by inserting its own instructions. This allows Valgrind to monitor memory operations during the program's execution.
  2. The instrumented program is executed under Valgrind's supervision. Valgrind keeps track of memory-related activities, such as allocations and deallocations, while the program runs.
  3. Valgrind maintains a detailed record of each memory block that is allocated, including its size, address, and other relevant information. This information is crucial for detecting leaks.
  4. Valgrind monitors whether each allocated block is properly deallocated. It keeps track of freed and allocated memory blocks and their associated addresses.
  5. If Valgrind identifies any memory blocks that were allocated but not freed before the program exits, it flags these as memory leaks.
  6. Valgrind generates a detailed report at the end of the program execution, summarizing the memory leaks detected. The report includes information about the source file, line number, and size of each leaked memory block.

2.3 GNU/Linux Bash Scripting

2.3.1 fzf

- What is fuzzy searching?
  A fuzzy search is a technique that uses search algorithms to find strings that match patterns approximately. It's particularly useful for helping users find webpages without having to know exactly what they're looking for or how a word is spelled. Fuzzy searches are also used for Structured Query Language lookups to help database users find records without having to be sure of the exact spelling of the value they're looking for.A fuzzy search is performed using a fuzzy matching algorithm, which returns a list of results based on likely relevance even though search argument words and spellings may not be an exact match. For web lookups, exact and highly relevant matches appear near the top of the list. Subjective relevance ratings may be given, usually as percentages.

- ls—fzf: