final project

Erfan Tajik

January 29, 2024

Contents

1 Git and Github

1.1 Repository Initialization and Commits

i make new repo in github, then i go to actions section and add new action and copy action file from milli repo and paste it, then i clone the project on my system, create new doc.tex file and push it

1.2 GitHub Actions for LaTeX Compilation

To create a lightweight tag (just a pointer to a commit), use:
$git tag [tagname]
To push a single tag to GitHub, use:
$git push origin [tagname]
by the action that we add in last section when we push in github with a new tag, github automaticaly compile our file and release it.

2 Exploration Tasks

2.1 Vim Advanced Features

1. Macros - Vim allows you to record and replay sequences of commands as macros. This allows you to automate repetitive tasks. To record a macro, press q followed by a register (a-z), perform your commands, then press q again to stop recording. To replay the macro, press @ followed by the register.

2. Vimscript - Vim has its own scripting language called Vimscript that allows you to customize and extend the editor. You can write Vimscripts to create custom commands, mappings, autocommands, and more. Vimscript files have a .vim extension and are loaded automatically on startup.

3. Splits and Tabs - Vim allows you to view and edit multiple files or views of the same file in a single session. You can split the window horizontally or vertically to show different split panes using :split or :vsplit. Within each split you can open a file. Vim also supports tabs, allowing you to open files or splits in separate tab pages accessed through :tabnew or :tabedit. You can navigate between splits and tabs to easily work across multiple files/views.

2.2 Memory profiling

2.2.1 Memory Leak

A memory leak occurs when memory is allocated dynamically but is not freed when it is no longer needed. This often happens when pointers to allocated memory are lost or overwritten before the memory is freed. Common causes in C include:

- Forgetting to free memory that was previously allocated with malloc()/calloc() /realloc(). Any memory not explicitly freed will remain allocated until the program exits.
- Failing to free memory before overwriting a pointer that points to it. If the only pointer to a block of memory gets overwritten, that memory can no longer be accessed to free it.
- Continuously allocating new memory without freeing old memory that is no longer needed. Over time this can cause the program to run out of available memory.
- Freeing the same memory block twice. This can corrupt the heap memory management structures, making future allocations fail.

2.2.2 Memory profilers

Valgrind is an extremely useful tool for detecting memory leaks and debugging other memory-related issues in programs written in languages like C and C++. Here are some key points about Valgrind:
Purpose: Valgrind is designed to help find memory management and threading bugs, including memory leaks, in programs. It can detect issues that are difficult to uncover with traditional debugging.
How it works: Valgrind instruments and monitors a program's execution to track how it uses memory. It inserts its own memory management code and checks for invalid operations like leaks.
Detecting leaks: When a program is run through Valgrind, it will track memory allocations and identify leaks by detecting any memory blocks that are allocated but never freed before program termination.
Other issues found: In addition to leaks, Valgrind can find uses of uninitialized memory, accesses of invalid memory, duplicate frees, memory overlap errors, and thread synchronization problems.
Minimal overhead: A major advantage of Valgrind is that it imposes fairly low overhead on execution time compared to other memory debugging approaches.
Usage: To use Valgrind, you run your program by executing the Valgrind tool with your program as the argument. It will analyze and report any issues as the program runs.

2.3 GNU/Linux Bash Scripting

2.3.1 fzf

Fuzzy searching is a search technique that finds matching items even if the search query doesn't exactly match the item. It looks for close or approximate matches based on similarity in characters, patterns, and semantics. This allows searches to be more flexible and tolerant of typos.
For example, a fuzzy search would match "appel" when searching for "apple",

despite the missing character. This makes it more powerful than strict literal searches.
Description of command: ls | fzf
This pipes the output of the ls command (listing files and directories) into fzf.
fzf is a command line fuzzy finder. It takes input, matches patterns in fuzzy/approximate way, and allows selecting an item using keyboard or mouse.
So ls | fzf will display an interactive list of files and folders, ordered by similarity to user keystrokes. You can keep typing to filter down matches, select the desired item with arrow keys/Enter, and it will output the selected path.
This makes it easy to visually select files/folders from the terminal in an efficient fuzzy manner, even in directories with many items.

2.3.2 Using fzf to find your favorite PDF

List all PDF files:
$fd .pdf
This will use fd (a faster alternative to find) to recursively search the current directory for all files ending in .pdf and list them.
Pipe the output to fzf to select a PDF:
$fd .pdf | fzf
This takes the listing of PDFs and pipes it into fzf, which will display an interactive fuzzy finder prompt. You can type characters to filter the list, use arrow keys to preview and select the desired PDF file. The selected PDF path will be output by fzf.