System-oriented Programming

Portfolio Assignment 05

Prof. Dr. Peter Braun

Technical University of Applied Sciences Würzburg-Schweinfurt

1 Introduction

The fifth portfolio exam in this module is the software development project. You must demonstrate in this project that you can develop a larger software in C in a small group of 3 to 4 students.

2 Dates

The deadline to submit your solution is Saturday, February 15th, 2025, at 6 pm CET.

3 Technical Requirements

1. The source code **must** be written in C. Use meaningful and descriptive file and function names. The project should be organized into multiple files using appropriate headers and source files (`.h` and `.c`). It **should not** be necessary to include excessive comments in the code if it is self-explanatory. Writing documentation is optional and will not be graded.

2. The program **should** implement a limited version of a Linux command (e.g., `ls` or `find`). We expect your solution to have more than 600 and less than 1,500 lines of code. Take these numbers only as a rough estimation.

3. The software **must** be capable of interacting with the filesystem and handling user-provided arguments. The program must use the Linux File API to read from and write to files. Command-line arguments **must** be supported. The program should handle user input provided via `argc` and `argv`, offering configurable options for the functionality of the command (e.g., flags like `-l` or `-r`).

4. The program must utilize dynamic data structures (e.g., linked lists, queues, or hash tables) for managing program data. These structures should be dynamically allocated and freed correctly to avoid memory leaks.

5. The program **should** support `stdin` and `stdout` for integration into pipelines. For example, the program's output should be able to feed into another command, and input could be provided from another program's output.

6. To enhance concurrency, the program **could** utilize threads for parallel processing where applicable. For instance, multiple directories could be traversed simultaneously using threads.

7. Synchronization mechanisms, such as semaphores or mutexes, **should** be implemented to ensure thread safety when accessing shared resources, like the dynamic data structures.

8. The project must be compiled using `gcc`. Use a `Makefile` to manage the build process, ensuring modularity and simplicity. The `Makefile` must include targets for building, cleaning, and running tests.

4 Deliverables

This assignment's deliverables are

1. a **Git repository** (on THWS Bitbucket, THWS Gitlab, or any other Git platform) that contains your solution. The Git repository must contain the source code of a backend, including test cases. The repository's root directory should contain a README file explaining how to start the system and execute all test cases.

2. a **short screencast video** of about 5 minutes (this can be added to the repository or store the video anywhere you like and where we can access it, e.g., in the THWS cloud, on Youtube as a private video, etc.). In the video, you **must** explain some technical aspects of your implementation. The video **must** be recorded by yourself and show the screen of your computer. Your voice must be audible. It is not necessary to show your face. Not all team members need to speak up in the video. The video must be provided in the MP4 format. The video should have at least a screen resolution 720p; it would be better to have 1080p. But not more. It should not be larger than 20 MB. Use tools like `ffmpeg` to compress it. It is not necessary to use slides in addition to the screencast. There is no title page, agenda page, or summary page. No intro or outro video, no music. But it would be nice if you said "Hi" at the beginning :-) The video must be playable using VLC or Quicktime Player. Please check the quality of the video before submitting it.

You must submit a plain text document (i.e., no Word or PDF) to Moodle before the deadline containing information on where to find these two deliverables. The text file must follow a pattern: the first text line contains the repository URL, and the second text line contains the URL to the video. The third line contains a comma-separated list of the immatriculation numbers of all students who worked on this project. Example:

```
https://git.fiw.thws.de/sop/mycoolproject
https://www.youtube.com/watch?v=CLi6YIWXkQI
5123007, 5123008, 5223001
```

Providing the document in this structure helps me to process all data automatically.

5 Rubrics

You can get 22.5 points for this assignment. When we grade your solution, we will execute the following steps:

- We clone the Git repository to our computer. We will check that there was no Git commit after the deadline (2025-02-15 at 6 pm CET). Otherwise, we will stop evaluating your submission.
- We read the instructions in the `README` file on how to start the build process. We will stop evaluating your solution if there is no `README` file and your software cannot be built by `make all`.
- Next, we watch your video. We are interested in learning how you split the problem into functions.
- We read your software system's source code and check the requirements mentioned in Sec. 3.

If any criteria are not completely fulfilled, we will reduce the points.