DM510 Operating Systems
Spring 2026

Exercise Sheet 8

Complete before tutorial on Thursday, April 16th

Learning goals

Be able to

- explain the central features of a file system (files, meta-data, directories, mounting of file systems)
- explain the role of FCBs (inodes)
- apply the three main allocation strategies (continuous, linked-list, indexed) and compare their efficiency in different settings
- use Linux system calls for file manipulation

Chapters 13-15

Exercise 1. PDF, HTML, and Markdown are examples of file formats. Download the following files, which contain the description of project 3 in the different formats:
https://larsrohwedder.com/teaching/dm510-26/exercises/project3.pdf
https://larsrohwedder.com/teaching/dm510-26/exercises/project3.html
https://larsrohwedder.com/teaching/dm510-26/exercises/project3.md

- Open each file in a texteditor or a hex editor. Discuss differences between these file formats.
- Discuss advantages and disadvantages of text-based files (like HTML and Markdown) compared to binary files (like PDF).

Hint: A hex editor displays the bytes of a binary file using hexadecimal numbers and often in a seperate window the ASCII letter interpretation of the bytes. This can be very useful for debugging binary files. In Linux, you can use `xxd <filename> | less` *for a very simple (readonly) hex editor.*

Exercise 2. Discuss reasons why operating systems have support for more than one file system type.

Exercise 3. Why do operating systems mount the root file system automatically at boot time?

Exercise 4. Could you simulate a tree directory structure with a directory in which arbitrarily long names can be used? If your answer is yes, explain how you can do so, and contrast this scheme with the tree directory scheme. If your answer is no, explain what prevents your simulation's success. How would your answer change if file names were limited to seven characters?

Exercise 5. Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

- How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)
- If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Exercise 6. Consider a file currently consisting of 100 blocks. Assume that the file-control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold.

- The block is added at the beginning.
- The block is added in the middle.
- The block is added at the end.
- The block is removed from the beginning.
- The block is removed from the middle.
- The block is removed from the end.

In the contiguous-allocation case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory.

Exercise 7. Why must the bit map for file allocation be kept on mass storage, rather than in main memory?

Exercise 8. Consider a system that supports the strategies of contiguous, linked, and indexed allocation. What could be criteria for deciding which strategy is best utilized for a particular file?

Exercise 9. Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

C Programming

In the following programming exercises, you need to use the file related system calls. You can consult the Linux man pages for details, for example: https://man7.org/linux/man-pages/man2/open.2.html.

Exercise 10. Discuss the functionality, parameters, and return value of the system calls `open`, `close`, `read`, `write`, `lseek`, `ftruncate`, and `mmap`.

Exercise 11. Write a program that opens a file, creating it if it does not exist. The program then writes $4 \cdot 1024 \cdot 1024$ times the character 'Q' using `write`. Make one implementation where the program writes one character at a time and one where it writes 1024 characters at a time. Compare the running times of both variants and interpret them.

Exercise 12. Write a program with the same functionality as before, but without `write`. Instead, use `mmap` to map the file to your memory space. Measure the running time and compare to the previous variants.
Hint: Use `ftruncate` to set the file size.

Exercise 13. In both of the programs from the previous exercises, after the initial write, replace every 42th character by 'P'. Compare and interpret the running times.
Hint: In the first program use `lseek`.