NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4186 - Operating Systems

Exercise 7

File System & Unix I/O

Professor:
Di Liu
Teaching Assistant:
Roman K. Brunner
Student Assistants:
Eik Hvattum Røgeberg
Halvor Linder Henriksen
Ole Ludvig Hozman
Daniel Hansen

Handout: 10.04.2023

Introduction

In this exercise, we are focussing again on the persistent storage in our system. We will explore disk characteristics, filesystems and also quickly tap into Unix-style I/O.

1 Accessing Files on a Harddsik

Assume the numbers given for the Barracuda Disk in Figure 37.5 in chapter 37.4[1] for the following exercises.

1. One way to improve the throughput of a disk and make sure we are using the full capacity of a disk is to run a compaction procedure. The compaction procedure reorders the files on the disk so that all the files are now contiguous and there are no holes between them. Now assume that this means we are required to move 128 8 KB files on the disk. How long does it take to complete the compact step? What is the average throughput of the operation? How does it compare to the max throughput shown in Figure 37.5?

2. Now lets assume we only have to copy a single file of size 1 MB. How long does it take, how is the average throughput and how does it compare to the previous exercise?

3. Even though SSDs do not really benefit from being compacted, lets assume we are copying the same amount of files as in exercise 1 and 2 above. You find the expected speeds for a selected range of SSD models in chapter 44 of the ostep book, page 18. Explain in which case an SSD would be much faster than the HDD and give some background on how the SSD achieves these high throughput numbers.

2 File Systems

In this part we will solve a few exercises on file systems and learn how they organize disk space that is available to us.

1. Which abstractions does a filesystem provide us with?

   - ☐ It unifies all system information in a single namespace
   - ☐ Gives us a logical layer providing directories and files above a persistent storage device
   - ☐ Tracks the metadata of files (e.g. last edited, created at, file type, ...)
   - ☐ Provides an abstract interface to read, write, and execute files that are stored on disk
   - ☐ The filesystem depends on the hardware layout of the persistent storage and abstracts from the hardware-specific mechanisms to a standard file interface

2. Given the following outline of an inode structure, what is the biggest file you could store on such a filesystem? Assume a block size of 4 KB, while having 8 direct pointers, 4 single-indirect pointers and 2 double-indirect pointers, with a disk-address size of 64 bits.

3. In the previous exercise: How much overhead do we have for storing the largest possible file?

4. You are building a system where you mostly expect large files to be stored. Propose another way of organising files on the disk in order to reduce the storage overhead and maximize the amount of storage used for useful data.

5. Describe a technique to make your filesystem described above more resilient to a system crash while updating a huge file. What guarantees are needed by the hardware to achieve higher resiliency?

[1] https://pages.cs.wisc.edu/~remzi/OSTEP/file-disks.pdf, page 7

3 Unix I/O

While file systems are generally independent of the OS, Unix played a crucial role in today's file-system interfaces we are working with. Due to this special role, we will also dive into the interface and abstractions of Unix.

1. Unix has the open syscall. Assume we would remove that syscall. Could one still operate on a file? How could such a system probably look like?

2. Can you always assume that your file descriptor is private to your process? If yes, how does the system ensure that? If no, what would you need to do to prevent race conditions?

3. We talked a lot about handling I/O in this exercise sheet, but the focus resided a lot with the filesystem and persistent storage. How does Unix model other types of I/O devices? How are they accessed and where are they exposed?

4. Practical Exercise: You might got to know the `ls` tool in the labs (e.g. xv6 has one, Linux or MacOS also have a `ls` tool). In this exercise, you will be implementing your own version of the ls tool. The output should correspond to the output of `ls -l --color=never` (sample output below). You might find it useful to read up on the `opendir`, `readdir` and `fstat` functions.

```
$ ls
drwxr-xr-x 2 username groupname    4096 Apr  4 07:24 lab1
-rw-r--r-- 1 username groupname 3355648 Apr  4 07:26 lab1.pdf
drwxr-xr-x 2 username groupname    4096 Apr  4 07:24 lab2
-rw-r--r-- 1 username groupname 2843648 Apr  4 07:26 lab2.pdf
drwxr-xr-x 2 username groupname    4096 Apr  4 07:24 lab3
-rw-r--r-- 1 username groupname 4441088 Apr  4 07:27 lab3.pdf
drwxr-xr-x 2 username groupname    4096 Apr  4 07:24 lab4
-rw-r--r-- 1 username groupname 3738624 Apr  4 07:27 lab4.pdf
```

Listing 1: Sample output for the `ls` utility