Tuesday, December 10, 2019,
17:00 – 18:30 written part
19:30 – 21:00 practical part

Practice Questions

Instructions for the written part of the exam:

- This exam is closed book. You may use one page A4 of handwritten notes.
- No electronic tools whatsoever (turn in your mobile phones, smartwatches, etc.)
- Write your answers in the space provided below the problem. Try not to make a mess and clearly indicate your final answer. If you run out of space, ask the TA for additional sheets.
- Write down the logical steps of your computation, not just the final answer.
- You can write your answers either in English or Korean. Make sure that your handwriting is legible.
- This written part of the exam contains *n* questions with a maximum score of *p* points.
- The problems are of varying difficulty. The point value of each problem is indicated. Solve the easy problems quickly and then work on the harder problems.
- If you finish before the time is up, stay seated and raise your hand. The TA will collect your exam and then let you go.

Good luck!

Problem	Points
Total	

Practice questions for the practical part

Questions for the practical part will be delivered in paper and electronic form. Most questions come with skeleton code that you have to fill in. During the exam, we "sell" hints for a point deduction. You will work with a locked-down version (no network access) of the CSAP VM. You will be allowed to use the textbook (in printed form only) for your reference.

Practice question 1

Write a program that shows information about a file/directory. If a filename is given as an argument, print the information about that file. If a directory is given, print information about all files in that directory. If no argument is given, assume ".", i.e., print information about all files in the current directory.

The output of a file should include
- the file size
- the file type. Distinguish between all types that are listed in the manpage of inode(7).
- the file modification time in human-readable format (YYYY-MM-DD HH:MM:SS)
- the filename

The sample solution produces the following output:

```
$ ./fileinfo fileinfo.c
     4461 F 2019-12-03 12:03:43 fileinfo.c
$ ./fileinfo
.:
     4461 F 2019-12-03 12:03:43 fileinfo.c
     4096 D 2019-12-03 21:25:41 .
    12768 F 2019-12-03 12:03:45 fileinfo
    16384 F 2019-12-03 21:25:50 .fileinfo.c.swp
     4096 D 2019-12-03 21:25:41 ..
$ ./fileinfo /dev
/dev:
     4420 D 2019-12-03 16:16:42 .
     4096 D 2019-07-30 06:53:07 ..
        0 C 2019-12-03 16:12:33 hidraw2
       80 D 2019-12-03 16:12:33 usb
        0 C 2019-12-03 16:12:33 hidraw0
        0 C 2019-12-02 18:38:47 vcs2
        0 B 2019-12-02 18:38:47 cdrom
      100 D 2019-12-02 18:38:47 dri
        0 C 2019-12-02 18:38:47 mixer1
...
```

The exact output is not important; the aim of this exercise is to understand file meta-data and directories.

Practice question 2

Write a program that takes as an argument the number of processes/threads to create. For each process/thread, a random amount of work is initialized. Each process/thread then processes that work in a simple loop.

The threaded version should use a single global variable 'sum' that is incremented by each thread. At the end of the program, the value of 'sum' should be equal to the total amount of work created.

The version using processes should use a pipe through which the child processes convey the (partial) sum to the parent. The parent reads all partial sums from the pipe's read end and prints the total sum. Again, this should be identical to the total amount of work created.

The "work" to be performed in each thread/process is a simple loop:

```
for (int i=0; i<iter; i++) {
  sum++;
}
```

where 'iter' is the amount of work passed to the process/thread and 'sum' is the global variable.

The output has been demonstrated in class. It is also not important how and how much work you create – the important part of this problem is to learn about threads, semaphores, processes, and pipes.

Echo server/client

The directory echo/ contains the source codes to the echo server(s) and the client.
If you get errors when compiling the echo client/server(s), make sure to run "make" in the utils/ directory first!

Networking library

During the exam, you will not have access to csapp.h. Instead, you will have access to our own library that contains a few functions that simplify some networking tasks. You will find the sources and the documentation in the directory utils/; for examples, have a look at the sources of the echo server/client that make use of this networking library.

Non-disclosure agreement

The source codes and sample solutions are provided with the intention to help you prepare for the exam. By downloading these sources you agree to not upload them to private or public websites, including gitlab, github, etc. We plan re-use these sources in future classes and homeworks; if the sources are readily available on the Internet, the echo client homework, for example, will become useless. Thank you for your cooperation.