Quiz #2, on Wednesday, September 19th, will consist of questions taken from or inspired Parts I and II of this homework.

Part I — Questions

1. The operating system manages resources (resource allocation) and provides an interface to resources for application programs (resource abstraction). Discuss how those two aspect both ultimately provides a service to the user.

2. If I want to develop a new application program, should I rather learn how are the system calls implemented on my computer or how to use the API? Explain your answer, and give arguments in favor of both approaches.

3. What is the role of the system call interface?

4. Give three examples of system calls that deal with file management.

5. Define what a kernel is.

6. When I turn on my computer, which one is loaded first: the kernel or the system processes?

7. Why are embedded device often using real-time operating systems?

8. Can an operating system be written in more than one programming language? Give a reason why one may want to use a high-level language, and a reason why one may want to use low-level (i.e., assembly) language.

9. Name an advantage of peer-to-peer system over client-server system.

10. Mechanisms implement *what* to do, or *how* to do it?

11. Name an advantage provided by a monolithic structure. Name a disadvantage.

12. Name an example of modular approach.

13. In a layered operating system, does a layer needs to know all the details of the implementation in the lower layers?

14. What is the purpose of a log file?

15. What is system generation?

16. Explain in a few word what a process is.

17. What is the state of the process being executed on the CPU?

18. Describe the actions taken by the kernel to context-switch between processes.

19. Give a reason why a process could make a transition from the waiting (a.k.a. blocking) state to the ready state.

20. What happen to a process when it is swapped out? What is the benefit of swapping out processes?

21. What is the name of the part of the O.S. that picks, among the processes in the ready queue, the one to execute on the CPU?

Part II – Problems

I'll assume that you will have successfully completed those tasks by Wednesday, September 19th, so don't wait and let me know if you had difficulties solving them.

Problem 1

Let us start by organizing our folders and files. The instructions below are for the recommended debian image, but it should be fairly easy to adapt them to other UNIX operating systems.

1. Create a new folder on your desktop (right-click, and then "New Folder"), name it "hw".
2. Open it (double click on it) and create a new folder in it, that you will name "02".
3. Right-click, and select "Open in terminal". You should see a terminal (a command-line interpreter) opening, with

   ```
   oscreader@OSC:~/Desktop/hw$
   ```

   displayed at the top of the window. If you are in any other directory (i.e., if, for instance,

   ```
   oscreader@OSC:~/Desktop/hw/02$
   ```

   is displayed), then go the right folder by ***c**hanging **d**irectory*:

   ```
   cd ~/Desktop/hw/
   ```

4. First, let's download (*through the **w**eb **get***) this homework, using

   ```
   wget -P 02
    ↪  http://spots.augusta.edu/caubert/teaching/2018/fall/csci3271/hw/02.pdf
   ```

   How can you find what this `-P` flag is doing?

5. Now, let's ***move*** what we did last time (in the `hw1` folder located two folders "above") to the current folder (denoted with `.`) using the `mv` command:

   ```
   mv ../../hw1 .
   ```

6. Now, let us ***list*** the files and folder in the `hw` folder:

   ```
   ls
   ```

   You should see

   ```
   oscreader@OSC:~/Desktop/hw$ ls
   02        hw1
   ```

7. Finally, let's rename the `hw1` folder by using the `mv` command (indeed, *moving* is taken to be the same as *renaming*):

   ```
   mv hw1 01
   ```

8. If you use the `ls` command again, you should see

```
oscreader@OSC:~/Desktop/hw$ ls
01        02
```

Much better! Starting now, I will leave to you to organize the files in a consistent, ordered way. This walk-through was mainly a pretext to show you how to navigate with a command-line interpreter. As always, use `man <cmd>` where `<cmd>` is the name of a command, to open its manual.

Problem 2

In this exercise, we will play with one of the implementation of hash functions, the `sha256sum` program.

1. Give an informal description of what a hash function does. Name one of the usage of such a function.
2. Create a folder named `hash` in your `~/Desktop/hw/02/` folder, and enter it, using

   ```
   mkdir ~/Desktop/hw/02/hash && cd ~/Desktop/hw/02/hash
   ```

   Then, create a file named `test1.txt`

   ```
   gedit test1.txt
   ```

   enter "test1" in it, save it, and close gedit. Now, print its SHA256 message digest (its *signature*), using

   ```
   sha256sum test1.txt
   ```

   Can you make any connexion between the content of the file and the signature printed?
3. Make a ***copy*** of that file, that you'll name `test2.txt`:

   ```
   cp test1.txt test2.txt
   ```

   Display `test2.txt`'s signature using the `sha256sum` command. Is the signature the same? Can you conclude something regarding the correlation between the name of a file, and its signature?
4. Edit the content of `test2.txt` by changing just one character, and then display its signature. Can you relate that signature to the signature of `test1.txt`? Does changing one character in your file changes only one character in its signature?
5. Define what a *collision attack* is (wikipedia is your friend).

Problem 3

Have a look at the discussion "How can I copy a file on Unix using C?" at `https://stackoverflow.com/q/2180079` (cached version at `https://web.archive.org/web/20161124214739/https://stackoverflow.com/questions/2180079/how-can-i-copy-a-file-on-unix-using-c`, if needed), and answer the following:

1. What is "sparseness" of a file, what is "btrfs cow"?
2. What is a `goto` statement?
3. What is "Bobby Tables attack", and why does "sanitizing" prevent them?

Problem 4

1. Start by reminding yourself what file descriptors are. You can have a brief description at `https://stackoverflow.com/q/5256599/` (cached version, if needed, at `https://web.archive.org/web/20180829013300/https://stackoverflow.com/questions/5256599/what-are-file-descriptors-explained-in-simple-terms`).

2. Open your virtual machine and install `strace`: open a terminal, type `su` to become administator (the password is "osc"), and then use `apt-get install strace` to download and install the `strace` program. Open the manual, and read (at least) the first two paragraphs of the DESCRIPTION.

3. Download and open `http://pages.cs.wisc.edu/~remzi/OSTEP/file-intro.pdf` (cached version at `https://web.archive.org/web/20180519225131/http://pages.cs.wisc.edu/~remzi/OSTEP/file-intro.pdf`), read and reproduce on your machine Section 39.4, *Reading And Writing Files*. To execute `strace cat foo`, you need to still be administrator. To exit the administrator mode, once you are done, type `exit`.