

Please read Chapter 1 and Sections 2 – 2.2 of the textbook and then answer the following, trying not to look at your notes or at the textbook. Quiz #1, on Thursday 7th September, will consist exclusively of questions taken from the Part 1 of this homework.

Part I — Short Questions

Question 1

An Operating System is sometimes called a "resources allocator", give two examples of resources the OS allocates.

Question 2

How do we call the program that manages the execution of user programs?

Question 3

Define what a kernel is.

Question 4

How do we call the initial program, generally stored in the firmware? What is its main purpose?

Question 5

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

Question 6

What is the name of an interrupt sent by a software?

Question 7

What is a volatile storage device? Name an example of volatile memory.

Question 8

What is direct memory access?

Question 9

What is the difference between a simple-purpose and a general-purpose processor?

Question 10

In a cluster, if a machine is in hot-standby mode, then is the cluster symmetric or asymmetric?

Question 11

If I double the number of processors in my computer, will my computer run exactly two times faster?

Question 12

Explain what graceful degradation is.

Question 13

What is the purpose of the mode bit?

Question 14

A multi-processor system where every processor can execute all tasks is called symmetric or asymmetric multiprocessing?

Question 15

Are all multicore system multiprocessor?

Question 16

A program in execution is called a ...?

Question 17

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

Question 18

What is the storage device that the CPU is able to address and access directly?

Question 19

Which one is usually the fastest, the cache or the main memory?

Question 20

In a multiprocessor environment where each CPU has a local cache, what is cache coherency and why does it matter?

Question 21

What is the difference between protection and security?

Question 22

Expand the acronyms **FIFO** and **LIFO**. Which schema is implemented by a queue?

Question 23

What is the particularity of the operating system of an embedded device?

Question 24

What is reverse engineering?

Question 25

Expand the acronym **CLI** and **GUI**.



Part II — Problems

This week's problems require a computer, an Internet connection, and some time. I'll assume that you will have successfully completed them by the time Homework #2 is released (Thursday 7th September), so don't wait and let me know if you had any trouble doing them. Problems 1 and 2 are of particular importance, since they help you to set your computer for the rest of the semester.

Problem 1

In this exercise, we will install a software that will allow us to run virtual machines on your computer, and run our first virtual machine. You will need administrator privileges, an Internet connection, and some time.

1. We first install VirtualBox, which is an (almost) open-source and completely free software that allows to run virtual machines. Go to <https://www.virtualbox.org/wiki/Downloads> and click on your operating system below "VirtualBox 5.1.26 platform packages". The steps to install this software from here should be immediate.
2. Download the following file: <http://people.westminstercollege.edu/faculty/ggagne/osc/vm/OSC-2016.ova>
3. Launch VirtualBox, click on "File", and then on "Import Appliance" (or, preferably, do "Ctrl" + "I"). Select the previously downloaded ova file, and click on "Next" and then on "Import".
4. Now you should come back to the first view you had, except that "OSC-2016 – Powered Off" should appear on the left menu on your screen. Click on it, and then on "start".
5. Your virtual machine should start in a new window, VirtualBox will probably inform you that your keyboard and mouse will be "captured" by the virtual machine every time this window is active.
6. Your machine should start and ask for the password of the user "oscreader": enter "osc" (without the quotes).

You're done!

Problem 2

In this exercise, we will make sure that your virtual machine can compile and execute programs from three different programming languages: C, C++ and java.

1. First, start your virtual machine: launch VirtualBox, start OSC-2016 and log-in.
2. (Optional: you may want to open firefox and load this homework, located at <http://spots.augusta.edu/caubert/teaching/2017/fall/csci3271/hw/01.pdf>, to relieve you from a part of the typing burden. Be careful when you copy and past, though.)
3. Open a terminal (right-click on your desktop, and click on "Open Terminal", or, alternatively, click on "Activities", and then on "Terminal"). The rest of this exercise will take place in this terminal.
4. Create a folder with

```
mkdir hw1
```

and enter it with

```
cd hw1
```

5. We will first compile and execute a C program.

(a) Create a folder and enter it:

```
mkdir C && cd C
```

(b) Launch the default editor and create a new file with the command

```
gedit hello.c
```

In this new window, type the following

```
#include <stdio.h>
int main()
{
    printf("Hello!\n");
    return 0;
}
```

save and quit. You can see that there was an error message printed in the console:

```
** (gedit:23310): WARNING **: Error when getting information for
→ file '/home/oscreader/hw1/c/hello.c': No such file or
→ directory
```

you can safely ignore it.

- (c) Compile your program with

```
gcc -o hello hello.c
```

and execute it with

```
./hello
```

- (d) Your program should print "Hello!" and a new line in your terminal. Return to the parent folder with

```
cd ../
```

6. We will now compile and execute a C++ program.

- (a) Create a folder and enter it:

```
mkdir Cplusplus && cd Cplusplus
```

- (b) Launch the default editor and create a new file with the command

```
gedit bonjour.cpp
```

In this new window, type the following

```
#include <iostream>
int main()
{
    std::cout << "Bonjour!\n";
    return 0;
}
```

save and quit. You can see that there was an error message printed in the console, you can safely ignore it.

- (c) Compile your program with

```
g++ -o bonjour boujour.cpp
```

and execute it with

```
./bonjour
```

- (d) Your program should print "Bonjour!" and a new line in your terminal. Return to the parent folder with

```
cd ../
```

7. To conclude, we will compile and execute a java program.

- (a) Create a folder and enter it:

```
mkdir java && cd java
```

- (b) Launch the default editor and create a new file with the command

```
gedit Hallo.java
```

In this new window, type the following

```
public class Hallo
{
    public static void main(String[] args)
    {
        System.out.print("Hallo!\n");
    }
}
```

save and quit. You can see that there was an error message printed in the console, you can safely ignore it.

- (c) Compile your program with

```
javac Hallo.java
```

and execute it with

```
java Hallo
```

- (d) Your program should print "Hallo!" and a new line in your terminal. Return to the parent folder with

```
cd ../
```

Problem 3

In this problem, we will create a kernel module, load it and unload it in the Linux kernel. We will then edit it.

- (a) 1. Start your virtual machine, and double-click on "osc9e-src" on the desktop. This will open your file explorer. Double-click on "ch2", and make a right-click anywhere but on a file in this window, and select "open in terminal" from the menu that appeared. All the commands below are to be typed in this terminal.

2. List all the modules that are currently loaded by entering

```
lsmod
```

Have a look at this list.

3. Compile the simple module using

```
make
```

4. List information about the simple module we are about to load using

```
/sbin/modinfo simple.ko
```

5. Load the module using

```
sudo insmod simple.ko
```

You will be asked to enter the password for oscreader, remember that it is "osc".

6. Make sure the module was loaded: the last line returned by the command

```
dmesg
```

should be something like

```
[7450.676808] Loading Module
```

and

```
lsmod
```

should list simple as one of the module loaded.

7. Now, unload the simple module using

```
sudo rmmod simple
```

Make sure it was unloaded by looking at what is now returned by the commands

```
dmesg
```

and

```
lsmod
```

(b) Now, you will be asked to edit this simple module. Start by removing the compiled files with

```
make clean
```

make a copy of the file with

```
cp simple.c simple_backup.c
```

and open `simple.c` with

```
gedit simple.c
```

Have a look at the code, and try to edit this file so that

1. You will be listed as its author,
2. Its name will be "My Module",
3. Instead of printing "Loading Module" (respectively, "Removing Module") when the module is loaded (resp., removed), it prints "Now my module is loaded into the kernel" (resp., "Now my module is removed from the kernel").

You can compile your edited module with

```
make
```

but how can you make sure that your editing actually changed the module? Describe what steps could be done to verify that your edited module actually reflect the change you made to the C file.

