Operating System Concepts Weekly

Week 1

Lucas van der Laan
s1047485

1

The purpose of an interrupt is to enable multi-task behaviour, in the sense that it tells the OS to stop the current task and start another.
A trap is a software interrupt and can either be caused by an error in software or by software doing a system call.
Yes, traps can be generated by system calls in software, this is done to elevate the privilege to kernel mode so it has access system calls like open and exec.

2

The obvious argument against is to say that if a "user" or something similar has a leak and access can be gained through a program or something like that, the adversary can do whatever they want, because they have complete access.
The argument in favour tho would be that you can secure the hardware quite well against any type of attack, like fault injection attacks, by either solving the problem in hardware or taking some extra precautions in the software. As long as the hardware has some fail-safes and the software has some extra thought put into it, it should be safe.

3

The processor could keep track of the memory range that is associated to every process and limit processes to only their memory ranges. Information about those ranges is in the base and limit registers, which can then be checked whenever a program accesses memory.

4

1. Passing parameters to registers.
2. Parameters can be stored in memory in a block, and only the block is then put on the register.
3. Parameters can be transported using the stack.

5

Advantages:
Operating Systems work by seeing everything as a file, this is good because in this way the OS can handle files and devices the same way. Another benefit is that we, the programmers, can use the same API for files and devices, with some small changes.

Disadvantages:
Because the API is the same for both devices and files, it could be more difficult to handle all the functionality of certain devices, as they may have functionally that cannot be handled easily in a file-like manner. This means that we either have less performance (because of a roundabout way of getting said functionality) or we have less functionality.

6

pwd displays the current directory, which for me was "/home/lucas/assignment1".
When a wrong command is entered, the shell acts like nothing ever happened.
The difference is that foobar is not an actual command, thus the program cannot create a new process with it and instead just sends an error back, while with something like pwd, the program starts another program.
The line on which this happens is line 82: int rc = ::execvp(c_args[0], const_cast<char**>(c_args));
The function on that line is execvp, which runs a different command and as stated by the comment above it, it replaces the current process instead of creating a new one so that the shell can continue.