Operating Systems
Assignment 3.1

Henri Heyden, Nike Pulow
stu240825, stu239549

Exceptions

1. A CPU or a virtual CPU unit like the OS
2. Interrupts, traps and syscalls are handled by the OS and signals are handled by a process
3. Interrupts, traps and syscalls really should be handled by the OS since else it would result in complete undefined behaviour, signals should also be handled since if not then there could not be more than one program running with more or less defined behaviour. If there would only be one process with no syntactical errors, it should be fine, but then there would not be any kind of OS behind, so it is not recommended.
4. Interrupts, traps and syscalls are handled on L2 and signals are handled on L3

Signals

5. Synchronous signals on a process are triggered by itself and asynchronous signals on a process are triggered by something else, like for example another process or an IO-Operation.
6.

Name	Type	Meaning
`SIGSEGV`	Synchronous	Segment violation
`SIGINT`	Asynchronous	Interrupt for an IO operation for example
`SIGTERM`	Asynchronous	Terminate the process (ignorable)
`SIGKILL`	Asynchronous	Kill the process (unignorable)
`SIGSTOP`	Asynchronous	Stop the process (unignorable)
`SIGCONT`	Asynchronous	Continue process

7., 8. and 9.:
From the Linux Programmer's Manual:
"The sigaction() system call is used to change the action taken by a process on receipt of a specific signal."
"signal() sets the disposition of the signal signum to handler, which is either

SIGIGN, SIGDFL, or the address of a programmer-defined function (a 'signal handler')."
Adding to that `signal()` is implemented differently across different OS where `sigaction()` is always implemented the same.
Also `signal()` resets to `SIGDFL` after each call and is not recommended by the Linux Programmer's Manual.

IPC

10. IPC is an abbreviation for: "inter-process-communication"
11. For example the `killall` (User-)command in Linux can kill processes that have a certain name. This could be implemented by sending `SIGKILL` signals to the processes after which the parent reports via `SIGCHLD` to the OS or the `killall` process, that the child has been killed.
Also another fun example would be a program, that kills their parent when they tell them to be killed. This would also be an example of IPC.

Parallelism and Concurrency

12. Parallelism is when multiple threads run on different real cores while Concurrency is when multiple threads run on the same core but share computing time (evenly) by being interrupted and continued by the OS.