Name: Wan Muhammad Rayhan Arwindra

NPM: 1806241210

This week, we learnt about scripting and signals. As mentioned on the log of the previous week, scripting is the act of programming or automating tasks by writing script. This script is then interpreted, not compiled, meaning that once made it can immediately be run on the console (assuming there exists no errors). Signals are a way to notify a process that a certain event has occurred, similar to how we get notifications on our mobile phones if we get a message.

There are many scripting languages out there, such as python or bash. In this course, we use bash scripting, can be run on the bash shell. The bash script files have an extension of .sh, and can be run using the bash command, for example "bash example.sh". Bash scripts can use while or for loops, have if-else statements, arithmetic and logical operators, and so on. To perform an arithmetic operator, we use the notation $((STATEMENT)). For example, if we wish to compute 1+1, we can write $((1+1)), and can then print the result by calling the echo command. We can also access the OS's settings, such as the current logged in user, the hostname, IP address, and even hardware specifications.

When we run a bash script, it becomes a running process. If we wish to kill that process while it's running, we could simply do CTRL+C, but we can also send a signal from another instance of our shell. This is done by writing "kill [PID]", where PID is the pid of the process we wish to kill. We could also specify what kind of signal we wish to send to the process, by typing "kill –[SIGNAL] [PID]". For example, "kill -USR1 1234" would send a USR1 signal to the process with pid 1234.

Signals can be trapped, caught, or blocked. This means that we can program a handler or subroutine should the program receive a certain signal. However, there exists two signals that cannot be handled, those being SIGKILL and SIGSTOP. SIGKILL would immediately kill the process, without asking for permission or notifying the process beforehand. If we wish to handle a signal which is meant to kill our process, we can send a SIGTERM process instead. SIGTERM would terminate the process on normal conditions, however we can trap the signal to perform a subroutine without actually terminating it.