Name: Wan Muhammad Rayhan Arwindra

NPM: 1806241210

For this week, we learnt about buffering, shells, and scripting. Buffering is a method of temporarily storing data as it is being moved from one place to another, usually the data is moved from the user space to the kernel space. Shells are the outer most layer to access the operating system's services, in the form of a command line (CLI) or a graphic user interface (GUI). Scripting is the act of programming or automating tasks by writing a script. Scripts are interpreted, not compiled like programming languages such as C or Java.

The issue of buffering comes from writing into the secondary memory, such as a hard disk or a solid state disk. This is because writing to those devices are quite slow, so if we write a program which repeatedly writes to it, the running time of the program would be inefficient. To solve this issue, we can use a buffer, such that all the data we write would enter a temporary storage first, before finally writing it to the device. The buffer acts as an interface for the data to be written, between the user space to the kernel space. When the buffer is full or flushed, the data is forwarded to the kernel space and is then written to the secondary memory.

Shells provide a way for users to access the operating system's services. It's called a shell because it's the outer most layer of the operating system's services, just like shells are the outer most layer of a snail or a crab. There are a couple of shells to note, which are bash, tcsh, ksh, zsh, and fish. The shells mentioned above have their respective advantages and disadvantages, with bash being the default shell for linux.

In the bash shell, we can write bash scripts, denoted by the file type of .sh. Bash scripts can do most of the things that programming languages can, such as if-else blocks, for loops and while loops, variables, logical operators, arithmetic operators and so on. Within the bash script, we can also access the operating system's services, such as accessing a file, reading a file, changing ownership of a file, and so on. In a bash script, we can also pass in an argument to it, just like how we pass arguments to a function in python or java. To do this, we can add a text after running the script, for example: bash example.sh argument. The arguments can be processed in the script by typing $n, where n is the order of the arguments passed, for example $1 is the first argument passed to the script.