Name: Wan Muhammad Rayhan Arwindra

NPM: 1806241210

This week, we learnt about physical and virtual memory, and memory allocation. Physical memory is memory used in the main memory, for example the memory on the RAM. As we know, RAM is relatively small, so the number of programs and processes we can run simultaneously cannot be too large, or the system won’t respond correctly. To handle this, we can use Virtual Memory, which takes a portion of memory from the secondary memory (HDD or SDD), to act as temporary storage, this frees up memory storage in the RAM and allows us to run larger programs at the same time.

Virtual memory isn’t perfect though. Programs that are run on virtual memory are generally slower than the ones running on Physical memory. Because of this, it’s a good idea to have as much physical memory as possible, however its quite expensive to do so. Virtual memory also slows down the system as a whole, since it requires the data to be mapped in-between virtual and physical memory, and also takes time to switch between multiple running programs when using virtual memory.

A process has several segments, that being code, data, heap, and stack. The data segment ends at a location called the program break. To change, or increase the program break, we can use several commands, such as brk() and sbrk(). These functions are similar, but ultimately do different things in the end. The brk() function sets the end of data segment to a specific value which is specified in the argument of the function call. On the other hand, sbrk() increments the program break by a specified amount.

To allocate more memory to our program, we can use the malloc() function. What malloc() does is it grabs a block of memory from the OS, and stores it in a linked list called “free list”, which is a list of free, available memory. Afterwards, every call to malloc() would give the called a chunk of the free memory by returning a pointer to that memory. To release the memory given from the free list, we can then use the free() function.

If we forget to use free(), or mismanage the allocation of memory within our program, we may end up with an issue called memory leak. A memory leak is a phenomena which occurs when a programmer forgets to delete unneeded memory from his program. This reduces the performance of the system because the amount of available free memory to use is less than needed. We can solve memory leak by calling the free() function.