Weekly Log 5 System Programming

Nama : Mahardika Krisna Ihsani

NPM : 1806141284

Kelas : A

Yang Saya Pelajari di Minggu ini

Pada minggu ini Saya mempelajari bagaimana struktur memory dari process. Struktur memory dari suatu process tersusun atas beberapa bagian dan masing-masing bagiannya mempunyai peran masing-masing.

- Stack: Terletak pada bagian atas address dan bergerak ke address yang lebih rendah. Bagian ini berperan untuk menyimpan variabel local.
- Heap: Bagian ini berperan untuk dynamic memory allocation dan bergerak dari address rendah ke address yang lebih tinggi. Apabila heap dan stack bertemu atau bertabrakan, maka akan menyebabkan stack overflow error.
- Data: Bagian ini berfungsi untuk menyimpan global atau static variable sebelum program dijalankan
- Text: Bagian ini berfungsi untuk menyimpan baris kode yang sudah di-compiled.

Selain itu, Saya juga mempelajari mengenai virtual memory. Virtual memory ini muncul karena keterbatasan physical memory (RAM) untuk menampung process-process yang berjalan. Virtual memory akan menangani penampungan process-process yang diperlukan dalam runtime suatu program. Virtual memory ini memanfaatkan sebagian space dari secondary storage(hard disk). Dengan adanya virtual memory, maka penampungan data hanya bisa dilakukan on-demand sehingga cukup hemat dan mengurangi beban dari physical memory. Selain itu juga, komputer dapat mengeksekusi banyak program dan menerapkan multi-threading environment. Namun, di sisi lain virtual memory mempunyai drawback, yaitu karena virtual memory sendiri berada di secondary storage, maka performa pengeksekusian prosesnya tidak sebagus performa dari physical memory. Selain itu juga virtual memory memakan tempat dari secondary storage. Dalam virtual memory juga terdapat istilah spatial locality dan temporal locality. Temporal locality adalah suatu instruksi yang baru saja dieksekusi mempunyai peluang besar untuk dieksekusi kembali, sedangkan spatial locality adalah semakin dekat address dari instruksi dengan address instruksi yang baru dieksekusi, maka semakin besar peluang instruksi tersebut akan dieksekusi.

Selain virtual memory, Saya juga mempelajari issue yang berkaitan dengan process dan memory, yaitu memory leak. Memory leak dapat dikatakan adalah suatu masalah yang menyebabkan suatu

objek/process sudah menjalankan instruksi atau sudah dikatakan tidak dibutuhkan lagi, namun tidak bisa untuk dibersihkan atau dihapus. Akibatnya memory consumption akan terus meningkat sehingga lama kelamaan sistem akan crash. Beberapa penyebab issue ini terjadi adalah zombie process, daemon yang tidak bisa melaksanakan tugasnya, dan variabel yang di-define secara static.

Selain itu, Saya mempelajari command yang berkaitan dengan memory, yaitu malloc, sbrk, brk, dan free. Sbrk dan brk sama-sama mempunyai fungsi untuk mengatur address dari program break(akhir dari suatu program). Letak perbedaannya adalah, brk berfungsi untuk mengatur address berdasarkan parameter input, sedangkan sbrk menambah space segment berdasarkan input parameter. Yang dilakukan malloc mengalokasi heap berdasarkan ukuran yang didefinsikan (bila memory yang tersedia di heap masih mencukupi dan terdapat blok yang mampu menampung ukuran input). Setelah mengalokasi space pada heap, maka pointer untuk address yang dibuat di malloc bisa digunakan untuk memasukkan data. Bila data dimasukkan address tersebut namun tidak ada heap space yang tersedia, maka pointer akan mengembalikan NULL. Malloc sendiri mencari blok yang tersedia melalui free block list. free() berguna untuk mendealokasi block dari memory dengan cara menambahkan free block ke linked list daripada menurunkan address program break. Merupakan suatu good practice untuk menggunakan free() secara eksplisit ketika ingin mendealokasi suatu blok, karena bila tidak sangat sulit untuk melacak terjadinya memory leak.

Selain itu Saya juga mempelajari mengenai command setjmp dan longjmp. Kedua command ini pada dasarnya adalah melakukan proses goto. Ketika setjmp pertama kali digunakan, maka sudah ditentukan ke mana flow program setelah longjmp dipanggil (yaitu menuju program counter di mana setjmp berada). Setjmp akan mengeluarkan nilai 0 ketika pertama kali dipanggil dan mengeluarkan nilai bukan nol bila bukan pertama kalinya dipanggil (nilai ini bergantung pada env yang disuplai oleh longjmp dimana untuk 0 nilainya adalah 1 dan sisanya adalah nilai itu sendiri_. Kedua command ini dapat menyebabkan masalah segmenation fault ketika setjmp diset di dalam suatu fungsi dengan parameter karena ketika longjmp dieksekusi, memang benar bahwa process akan pindah ke program counter di mana setjmp berada, namun function tersebut membutuhkan nilai argumennya pada stack, sedangkan ketika melakukan jmp, nilai argument tersebut undefined sehingga hal ini menyebabkan masalah. Bila dikaitkan dengan pok, setjmp dan longjmp ini menerapkan stack.