Week 5 Logbook

Aktivitas Minggu ini
Pada minggu ini, saya melakukan hal-
hal di bawah ini untuk meningkatkan pengetahuan dan pengalaman saya da
lam System Programming:
1. Mengerjakan Pre Test 5
2. Mengerjakan WS5
3. Menonton kembali video materi di Youtube yang sudah diunggah seming
gu yang lalu
4. Menghadiri kelas Sysprog pada hari Jum'at tanggal 16 Oktober 2020
5. Mencari sumber referensi tentang locality of reference, struktur me
mori, implementasi free dan malloc secara detail, virtual memory, dan
memory leak
6. Membaca manual command pada Linux, khususnya longjmp dan setjmp
7. Berdiskusi serta menyamakan pengetahuan dengan teman mengenai mater
i minggu ini
8. Membaca kembali slide yang sudah disiapkan di Scele
9. Trial and error untuk bisa mengcompile file free_and_sbrk.c di Linu
x
10. Trial and error untuk bisa menampilkan value dari program break ya
ng ada di soal no.5b

Spatial Locality
Spatial Locality adalah penggunaan fungsi atau instruksi yang terletak
di dekat sebuah instruksi yang sudah pernah dieksekusi sehingga instr
uksi tersebut berpeluang tinggi untuk akan dieksekusi oleh program. Bi
asanya spatial locality merujuk pada data element yang relatif dekat d
engan storage (dalam case ini adalah program code).

Temporal Locality
Temporal Locality adalah penggunaan kembali secara cepat sebuah fungsi
atau instruksi yang sebelumnya sudah pernah dieksekusi. Fungsi atau i
nstruksi-
instruksi yang akan digunakan kembali ini biasanya disimpan dalam cach
e memory agar mempermudah pemanggilannya dan mempercepat waktu untuk m
encari instruksi yang relevan pada memory yang lain.

Struktur Memori

1. Text-segment - sebuah segmen yang berisi teks, biasanya teks-teks ini berupa instruksi yang siap untuk dieksekusi.
2. Initialized Data segment - biasa disebut sebagai data segment yang menyimpan global variable dan static variable yang dibuat oleh programmer dalam suatu program.
3. Uninitialized Data segment - biasa disebut dengan nama bss, berfungsi untuk menyimpan global variable dan static variable yang valuenya diinisiasi ke nilai 0 atau tidak secara eksplisit diinisiasi oleh programmer dalam suatu source code.
4. Heap - segmen yang adjoin dengan stack dimana biasanya terjadi alokasi memori secara dinamis pada memory. Biasanya segmen ini diatur oleh beberapa system call seperti free(), malloc(), realloc() dan lain-lain.
5. Stack - merupakan segmen yang adjoin dengan heap, berisi program stack, dan LIFO. Pada stack ini, automatic variable dan sebuah pointer yang merujuk pada line dari eksekusi suatu program biasanya disimpan.

Memory Leak
Memory leak adalah suatu isu yang muncul ketika suatu program tidak dengan benar mengimplementasi alokasi memori sehingga pada heap yang ada di memory masih ada resource yang belum dibuang. Hal ini menyebabkan terbuangnya resource yang ada di suatu memori.

Memory leak biasanya terjadi akibat seorang programmer tidak sadar bahwa program yang ia sudah buat menyebabkan memory leak.
Biasanya kesalahan oleh programmer ini juga disebabkan akibat tidak adanya penanganan alokasi memori setelah digunakan sehingga resource yang digunakan oleh suatu program yang mengalami memory leak akan terus digunakan yang pada akhirnya membuat sistem operasi menjadi lambat dan kekurangan resource. Untuk suatu program (contohnya program yang dibuat dari bahasa pemrograman Java), memory leak juga bisa terjadi ketika objek yang diletakkan pada heap yang ada di memory tidak bisa diambil oleh garbage collector milik Java sehingga objek tersebut akan terus memakan resource yang ada pada komputer.

Virtual Memory
Virtual memory adalah suatu fitur pada sistem operasi dimana sistem operasi memungkinkan untuk menggunakan secondary memory atau storage sebagai suatu 'RAM' sehingga mempermudah paging dan mempercepat performa

komputer. Kita membutuhkan virtual memory sebab komputer terkadang mengalami kejadian dimana space yang ada di physical memory sudah penuh sehingga nantinya program akan sulit untuk dijalankan. Oleh karena itu, sistem operasi mengubah/membuat sebagian kecil dari hardisk dijadikan virtual memory sehingga program-program lainnya bisa berjalan.

brk()
brk() berfungsi untuk mengubah suatu akhir dari segmen data menjadi nilai yang berdasarkan pada address. Selain itu, brk() jika berhasil mengembalikan nilai 0, dan jika gagal akan mengembalikan nilai -1.

sbrk()
sbrk() berfungsi untuk meng-
increment space dari program data dengan format byte yang ditambah. sbrk() jika berhasil mengeksekusi fungsi ini akan mengembalikan ke pointer dimana program tersebut memanggil sbrk() dan jika gagal akan mengembalikan nilai -1.

Mekanisme malloc()
Ketika malloc() dieksekusi, malloc() akan membaca parameter berapa banyak memori yang harus dialokasikan (dalam byte). Setelah itu, malloc() akan memanggil sbrk() untuk mencari sebuah pointer ke heap yang kosong pada memori. Lalu, sebuah block yang berisi free memory akan diambil dari free list (sebuah linked-
list yang berisi blok memori bebas yang tersedia pada memori). Setelah itu, pointer dari block tersebut akan dikembalikan ke program. Namun jika tidak ditemukan adanya block yang tersedia, maka malloc() akan mengembalikan value berupa null.

Mengapa free() harus secara eksplisit dipanggil?
Karena dengan kita memanggil free() secara eksplisit untuk men-
deallocate memori setelah suatu proses selesai, kita dapat menghindari memory leaks yang mungkin terjadi. Selain itu kita juga akan terlatih untuk terbiasa untuk mengetahui penggunaan memori yang sedang kita gunakan sekarang sehingga kita tahu kapan kita harus membuangnya atau harus membiarkannya.