Tugas Besar
IF2230 - Sistem Operasi
Milestone 03 of 03
"April Fools"
Pembuatan dan Eksekusi Program yang Menggunakan System Call

Dipersiapkan oleh :
Asisten Lab Sistem Terdistribusi

Didukung Oleh :

Waktu Mulai :
Kamis, 01 April 2021, 20.21 WIB

Waktu Akhir :
Jumat, 23 April 2021, 20.21 WIB

I. Latar Belakang

Setelah beberapa minggu Anda sebagai seorang System Engineer yang *restless*, Anda menunjukkan skill Anda kepada para hunter kelas S. Anda akhirnya mendapatkan pesan dari sistem. Anda melihat kalender dan ternyata hari ini tanggal 1 April. Mengikuti panggilan sistem, Anda kembali ke dungeon tempat semuanya bermula.

Anda sudah menyiapkan beberapa potions dan mental Anda untuk mengakhiri segalanya pada hari ini. Anda memasuki double dungeon dan bertemu kembali dengan boss dari dungeon tersebut, namun kali ini level Anda sudah jauh lebih tinggi dibanding sebelumnya.

Anda pun menyadari bahwa boss tersebut bukanlah boss yang sebenarnya, melainkan patung yang memegang sebuah batu ajaib. Akhirnya Anda berbincang dengan sistem. Anda pun melawan sistem dan berhasil mengalahkannya dengan trik dan skill yang telah Anda miliki. Namun, tiba-tiba Anda masuk ke dalam sebuah dunia yang berisi jutaan sistem yang diatur oleh seseorang yang memiliki kemampuan *System Engineering* yang melebihi kemampuan Anda.

Sementara Anda masuk ke alam sadar, rekan-rekan hunter lainnya bergegas membantu Anda, namun mereka tidak mampu menyentuh sistem. Puluhan hunter pun tidak berhasil menyelesaikan *d*mo* dengan sistem, namun pengorbanan mereka tidak sia-sia. Anda pun terbangun dari alam sadar Anda hanya untuk melihat puluhan rekan Anda tidak berhasil menyelesaikan *d*mo*.

Anda menginginkan semuanya ini hanyalah sebuah kebohongan atau april fools, namun... setelah Anda membaca berita terbaru yang terpercaya, ternyata april fools dibatalkan.

II. Deskripsi Tugas

Pada praktikum ini, kalian akan melanjutkan sistem operasi sehingga menjadi sistem operasi yang lebih berguna. kalian dengan membuat fitur-fitur berikut ini

- Menambahkan *system call* baru untuk mengeksekusi program.
- Membuat *library* sederhana untuk mengakses fitur-fitur dasar sistem operasi.
- Membuat aplikasi terpisah menggunakan system call yang telah dibuat untuk melakukan suatu fungsionalitas tertentu.
    - Aplikasi utilitas
    - BONUS: Buat program eksternal sederhana, pertahankan state shell, implementasi flag -r pada cp dan rm.

III. Langkah Pengerjaan

3.1. Membuat Library Sederhana

Sebuah sistem operasi baru akan berguna jika dapat membantu program-program lain berjalan di atasnya, sebagai salah satu fungsi utama sistem operasi yaitu abstraksi. Oleh karena itu, agar pembuat aplikasi dapat dengan lebih mudah mengembangkan aplikasi pada sistem operasi Anda, akan dibuat sebuah *library* sederhana. *Library* ini akan terbagi menjadi beberapa modul yaitu:

- Modul teks (input, output, string utility)
- Modul fileIO (read, write, delete)
- Modul folderIO (create, delete)
- Modul matematika (mod, div)

Setiap modul harus dapat di-*include* secara terpisah agar aplikasi tidak menjadi *bloated*. *Signature* fungsi yang digunakan dibebaskan selama memenuhi fungsi yang diperlukan. Jika membutuhkan fungsi lain, diperbolehkan untuk menambahkan.

3.2. Membuat aplikasi utilitas

Sebuah sistem operasi akan jauh lebih berguna jika mempunyai utilitas-utilitas yang berguna (misal utilitas *cat*). Dengan menggunakan *library* yang sudah kalian buat (dilarang memanggil langsung interrupt). Buatlah utilitas berikut:

- **mv** (memindahkan file/folder)
- **cp** (mengcopy file/folder)
- **mkdir** (membuat directory)
- **rm** (menghapus file/folder)
- **cat** (mencetak isi file)
- **ln** (membuat symbolic link)

Perintah-perintah di atas, metode penggunaannya disamakan dengan yang ada pada **sistem operasi linux**, namun **tanpa implementasi *flag*** untuk spesifikasi wajib.

Pada sistem operasi ini, aplikasi tersebut akan diletakkan pada direktori **/bin**, yang merupakan *child directory* dari *root*. Aplikasi apapun yang disimpan pada folder spesial ini dapat dipanggil dari titik mana saja dalam shell, sehingga aplikasi utilitas tersebut wajib kalian letakan dalam folder tersebut, agar dapat dieksekusi dari mana saja.

Semua parameter dari perintah di atas **harus didokumentasikan pada README**.

3.3. Mengeksekusi Program

Program yang telah anda buat akan dieksekusi dengan cara memanggil nama program tersebut dari dalam shell. Fungsionalitas untuk mengeksekusi program dilakukan dengan menambahkan potongan kode berikut.

kernel.asm

```
_launchProgram:
	mov bp,sp
	mov bx,[bp+2]

	mov ax,cs
	mov ds,ax
	mov si,jump
	mov [si+3],bx
```

```
        mov ds,bx
        mov ss,bx
        mov es,bx

        mov sp,0xfff0
        mov bp,0xfff0

jump:   jmp 0x0000:0x0000
```

Tambahkan juga potongan kode yang ditandai kuning pada **handleInterrupt21**, dan tambahkan fungsi baru **executeProgram**.

kernel.c

```
void handleInterrupt21 (int AX, int BX, int CX, int DX){
    char AL, AH;
    AL = (char) AX;
    AH = (char) (AX >> 8);
  switch (AL) {
    case 0x0:
      printString(BX);
      break;
        .
        .
        .
    case 0x6:
      executeProgram(BX, CX, DX, AH);
      break;
    default:
      printString("Invalid interrupt");
  }
}
        .
        .
void executeProgram(char *filename, int segment, int *success, char parentIndex) {
    // Buat buffer
    int isSuccess;
    char fileBuffer[512 * 16];
    // Buka file dengan readFile
    readFile(&fileBuffer, filename, &isSuccess, parentIndex);
    // If success, salin dengan putInMemory
    if (isSuccess) {
        // launchProgram
        int i = 0;
        for (i = 0; i < 512*16; i++) {
            putInMemory(segment, i, fileBuffer[i]);
```

```
        }
        launchProgram(segment);
    } else {
        interrupt(0x21, 0, "File not found!", 0,0);
    }
}
```

Eksekusi program harus bisa langsung menerima argumen yang diberikan dari shell. Sebagai contoh, pemberian perintah **“cp a.txt b.txt”** pada shell akan mengeksekusi program **“cp”**, lalu saat sudah berada dalam program tersebut, argumen yang telah diberikan sebelumnya dapat langsung digunakan tanpa memerlukan input tambahan lagi.

TIPS: Anda dapat melakukan ini dengan cara melakukan *message passing* dari shell ke program yang dipanggil melalui *temporary file*.

Eksekusi program dibagi menjadi dua mode: lokal dan global. Eksekusi program secara global dapat dilakukan apabila program tersebut berada dalam folder **/bin**, sedangkan eksekusi program secara lokal dapat dilakukan apabila program tersebut berada pada direktori yang sama dengan posisi shell sekarang. Pemanggilan program secara global dapat langsung dilakukan dengan menulis nama programnya, sedangkan pemanggilan program secara lokal dilakukan dengan menambahkan **“./”** sebagai prefix (contoh: **“./program1”**)

3.4. Batasan-batasan

Adapun batasan-batasan yang diberikan pada milestone ini adalah:

- Besar kernel maksimal hanyalah 16 sektor (8192 bytes). Oleh karena itu, implementasi shell dan aplikasi (serta logo) harus dilakukan sebagai program terpisah dari kernel. Kernel hanya mengandung *system call* yang telah dibuat sebelumnya. Pastikan nilai **KSIZE** pada **bootloader.asm** telah sesuai dengan batasan (16). Jangan lupa untuk menyesuaikan penggunaan sektor pada **sectors.img**
- Setelah eksekusi program utilitas (atau program lainnya) selesai, maka sistem operasi akan kembali ke shell.
- Karena setiap program akan menjadi satu file, maka ukuran maksimal program anda adalah 16 sektor (8192 bytes)

3.5. BONUS

- Shell akan tetap mempertahankan *state*-nya sebelum dan setelah mengeksekusi program. *State* yang dimaksud misalnya *directory* saat ini.
- Membuat program eksternal yang dapat kalian eksekusi di dalam sistem operasi yang dibuat. Program minimal dapat menerima input dan mengeluarkan output. Program berupa sebuah permainan sederhana adalah nilai tambah untuk bonus ini.
- Dapat mengeksekusi suatu program yang disimpan di *filesystem* dari mana saja (panggil dengan path program tersebut, misalkan **./a/b/program1**)
- Mengimplementasikan ***flag* -r** untuk command **rm** dan **cp**. *Flag* **-r** menunjukkan bahwa *command* dieksekusi secara rekursif. Silakan coba di sistem operasi Linux Anda/bash untuk melihat contohnya.

IV. Pengumpulan dan Deliverables

1. Untuk tugas ini Anda diwajibkan menggunakan *version control system* **git** dengan menggunakan *repository* ***private*** yang telah Anda buat sebelumnya di Github Classroom "**informatika19**".
2. Walaupun *commit* tidak dinilai, namun diharapkan melakukan *commit* yang wajar (tidak semua kode satu *commit*)
3. Anda **boleh menggunakan** kode milestone 2 dari teman kalian apabila sebelumnya kalian belum sempat menyelesaikan milestone 2 secara penuh. Namun perlu ditekankan bahwa Anda **hanya boleh menggunakan kode milestone 2 (rilis tag v.2.0.0) teman Anda**, bukan **pekerjaan teman Anda yang telah diubah untuk milestone 3 atau revisi dari demo**. Cantumkan kode teman mana yang kalian pakai (kelas dan nama kelompoknya) di README.md untuk milestone ini. **Apabila tidak mencantumkan, akan dianggap plagiarisme dan dapat dikenakan sanksi yang berlaku.**
4. File yang harus terdapat pada *repository* adalah file-file *source code* dan *script* (jika ada) sedemikian rupa sehingga jika diunduh dari github dapat dijalankan. Dihimbau untuk tidak memasukkan *binary* atau *image* hasil kompilasi ke *repository*
5. Kelompok tetap sama dan dapat dilihat pada link berikut s.id/kelompok-os19.
6. **Mulai** Kamis, 01 April 2021, 20.21 WIB waktu server.

Deadline Jumat, 23 April 2021, 20.21 WIB waktu server.

Setelah lewat waktu *deadline*, perubahan kode akan dikenakan pengurangan nilai

7. Pengumpulan dilakukan dengan membuat ***release*** dengan tag **"v.3.0.0"** (tanpa tanda kutip) pada repository yang telah kelompok Anda buat sebelum deadline. **Repository team yang tidak memiliki tag ini akan dianggap tidak mengumpulkan Milestone 3. Kesalahan dalam penulisan tag akan berakibat pada pengurangan nilai.**
8. Teknis pengumpulan adalah via kode yang terdapat di *repository* saat *deadline.* Kami akan menindaklanjuti **segala bentuk kecurangan**
9. Diharapkan untuk mengerjakan sendiri terlebih dahulu sebelum mencari sumber inspirasi lain (Google, maupun teman anda yang sudah bisa). Percayalah jika menemukan sendiri jawabannya akan merasa bangga dan senang.
10. Dilarang melakukan kecurangan lain yang merugikan peserta mata kuliah IF2230.
11. Jika ada pertanyaan atau masalah pengerjaan harap segera menggunakan sheets QnA mata kuliah IF2230 Sistem Operasi pada link berikut s.id/qna-prak.

V. Tips

1. Bcc tidak menyediakan *check* sebanyak gcc sehingga ada kemungkinan kode yang Anda buat berhasil *compile* tapi *error*. Untuk mengecek bisa mengcompile dahulu dengan gcc dan melihat apakah *error.*
2. Untuk melihat isi dari *disk* bisa digunakan utilitas **hexedit**.
3. Walaupun kerapihan tidak dinilai langsung, **kode yang rapi akan sangat membantu saat *debugging***.
4. Fungsi-fungsi dari *stdc* yang biasa Anda gunakan seperti mod, div, strlen, dan lainnya tidak tersedia di sini. Anda harus membuatnya sendiri, terutama mod dan div yang akan sangat berguna.