# Laporan
# Tugas Besar IF2230 Sistem Operasi Milestone 2

# Pembuatan Sistem Operasi Sederhana dengan Hierarchical File System, Shell, Utility Programs


oleh

**Harry Rahmadi M. / 13517033**

**Muhammad Hendry Prasetya / 13517105**

**Ridwan Faturrahman / 13517150**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG


2019

# 1. Langkah-langkah Pengerjaan

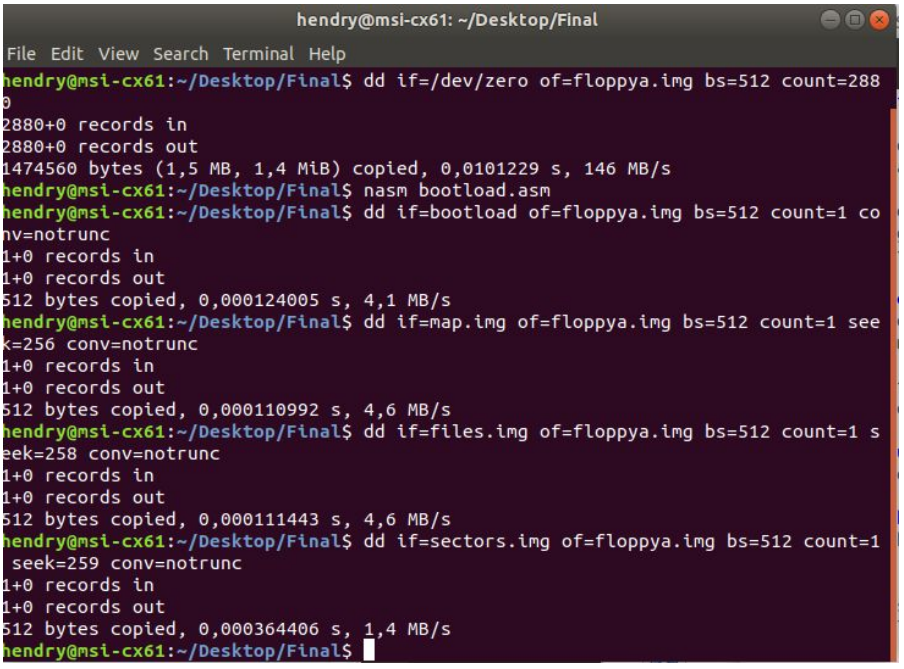
**A.** Mengubah kernel sistem operasi

**1.** Mengubah struktur *file system* menjadi ***hierarchical file system***

Berbeda dengan struktur *file system* pada milestone sebelumnya yang mana setiap berkas berada pada satu direktori root saja, struktur *file system* pada milestone ini bersifat hirarkis. Untuk *file system* yang baru ini, direktori root dapat berisi direktori lain juga, yang mana direktori lain tersebut juga dapat berisi berkas-berkas dan direktori lain.

Struktur dari file system os pada milestone ini adalah :

- **map** dipindahkan ke sektor 0x100. Hal ini dilakukan supaya sektor-sektor utilitas tidak memakan tempat sektor-sektor yang dapat digunakan oleh file system (0x00-0xFF).
- **dir** dihilangkan.
- Sektor baru **dirs** dibuat di sektor 0x101. Sektor ini berfungsi untuk menyatakan sebuah direktori pada file system. Setiap entri memiliki indeks direktori parentnya (0x00-0x1F, 0xFF jika root) dan nama direktorinya sendiri maksimum sepanjang 15 karakter.
- Sektor baru **files** dibuat di sektor 0x102. Sektor ini berfungsi untuk menyatakan sebuah file pada file system. Setiap entri memiliki indeks direktori parentnya (0x00-0x1F, 0xFF jika root) dan nama filenya sendiri maksimum sepanjang 15 karakter.
- Sektor baru **sectors** dibuat di sektor 0x103. Sektor ini berfungsi untuk menyatakan sector-sector setiap file.
- **kernel** (di sektor 0x1-0xA)



2. Mengubah fungsi **`handleInterrupt21`**

   Parameter AX dari fungsi handleInterrupt21 akan dipisah menjadi dua buah bagian yaitu AL dan AH. AH diisi dengan AX yang digeser ke kanan sebanyak 8 bit.

3. Mengubah implementasi *syscall* **`readFile`**

   Syscall readFile akan diubah sehingga menerima path relatif dari file yang akan dibaca seperti "**abc/def/g**", bukan nama filenya saja. Syscall ini dipanggil melalui interrupt 0x21 AL=0x04. Pada program kami, implementasi readFile menggunakan metode rekursif. Hal ini disebabkan input path tidak selalu langsung mengarah ke nama file.

4. Mengubah implementasi *syscall* **`writeFile`**

   Syscall writeFile diubah sehingga menerima path relatif dari file yang akan ditulis seperti "**abc/def/g**", bukan nama filenya saja.*Syscall* ini dipanggil melalui interrupt 0x21 AL=0x05. Pada program kami, implementasi writeFile menggunakan metode rekursif. Hal ini disebabkan input path tidak selalu langsung mengarah ke nama file.

5. Mengubah implementasi *syscall* **`executeProgram`**

   *syscall* executeProgram diubah sehingga menerima path relatif dari program yang akan dieksekusi. tidak terlalu banyak perubahan karena perubahan utama sudah ditangani oleh readFile.*Syscall* ini dipanggil melalui interrupt 0x21 AX=0x06

6. Implementasi *syscall* **`terminateProgram`**

   *syscal* terminateProgram dibuat dengan mengeksekusi *syscall* executeProgram yang mengembalikan eksekusi program ke shell sehingga seakan-akan mengakhiri eksekusi program sebelumnya. *Syscall* ini dipanggil setiap akhir program. Syscall ini dipanggil melalui interrupt 0x21 AL=0x07.

7. Implementasi *syscall* **`makeDirectory`**

   syscall makeDirectory diimplementasikan dengan menerima path relatif dari direktori yang akan dibuat, contohnya "**abc/def/ghi**". Syscall ini dipanggil melalui interrupt 0x21 AL=0x08. Pada program kami, implementasi makeDirectory menggunakan metode rekursif. Hal ini disebabkan input path tidak selalu langsung berada pada directory yang diinginkan.

**8.** Implementasi *syscall* **`deleteFile`**

*syscall* deleteFile diimplementasikan dengan menerima path relatif dari file yang akan dihapus, contohnya "**abc/def/g**". *Syscall* ini dipanggil melalui interrupt 0x21 AL=0x09. Syscall ini menggunakan metode rekursif.

**9.** Implementasi *syscall* **`deleteDirectory`**

syscall deleteDirectory diimpementasikan menerima path relatif dari direktori yang akan dihapus, contohnya "**abc/def/ghi**". *Syscall* ini juga akan menghapus secara rekursif isi dari direktori tersebut, termasuk isi dari direktori lain di dalam direktori tersebut. *Syscall* ini dipanggil melalui interrupt 0x21 AL=0x0A.

**10.** Implementasi *syscall* **`putArgs`**, **`getArgc`**, **`getArgv`**

Syscall putArgs (interrupt 0x21 AL=0x20) digunakan untuk menyimpan current directory, jumlah parameter program, dan isi parameter program sebelum program dieksekusi.

Syscall getCurdir (interrupt 0x21 AL=0x21) digunakan untuk mendapatkan indeks current directory. Syscall getArgc (interrupt 0x21 AL=0x22) digunakan untuk membaca jumlah parameter program.

Syscall getArgc (interrupt 0x22 AL=0x23) digunakan untuk membaca isi parameter ke-n program.

**11.** Implementasi fungsi pembantu

Kami menggunakan beberapa fungsi pembantu seperti writeln, strcmp, strcpy agar lebih memudahkan proses pembuatan program. Selain itu, kami menggunakan file header berisi kumpulan fungsi pembantu agar program-program yang terpisah tidak perlu menuliskan berulang kali implementasi fungsi pembantu

func.h

```
30  void readStringFull(char *string);
31
32  void writeln(char *string);
33
34  int strcmp(char *s1, char *s2, int length);
35
36  void printString(char *string);
37
38  void readString(char *string);
39
40  void readSector(char *buffer, int sector);
41
42  void writeSector(char *buffer, int sector);
43
44  void readFile(char *buffer, char *filename, int *success, char paren
45
46  void writeFile(char *buffer, char *filename, int *sectors, char pare
47
48  void deleteFile(char *path, int *result, char parentIndex);
49
50  void deleteDirectory(char *path, int *result, char parentIndex);
```

**B.** Membuat *shell* sistem operasi

**1.** Implementasi perintah **cd**

Program shell juga menyimpan nilai indeks dari *current directory* sebagai sebuah variabel berukuran byte. Current directory dapat diubah menggunakan command cd, dengan parameter pertamanya adalah *path* relatif ke direktori baru yang diinginkan pengguna.

**2.** Implementasi perintah **menjalankan program**

Pada contoh dibawah ini syscall executeProgram untuk menjalankan program dengan nama "**myprog**" pada root directory

```
$ myprog
```

Untuk mencari program pada *current directory* sintaks '**./**' menandakan shell

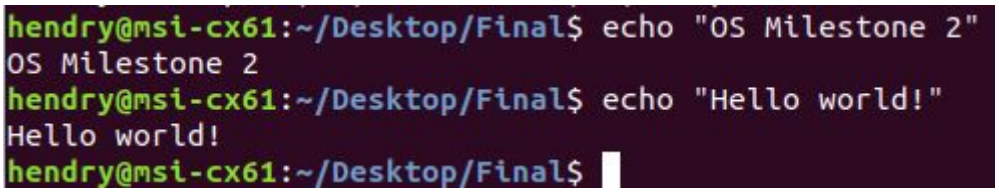
```
$ ./myprog
```

Sebelum executeProgram dijalankan, gunakan terlebih dahulu setArgs untuk menyimpan direktori sekarang (curdir), jumlah argumen (argc), dan nilai-nilai argumen (argv). Untuk melakukan *get* argumen program, digunakan *syscall* getCurdir, getArgc dan getArgv

**C.** Membuat program-program utilitas

**1.** Membuat **echo**

Program echo mencetak parameter pertama dan seterusnya yang diberikan user ke layar. Berikut adalah echo pada linux:

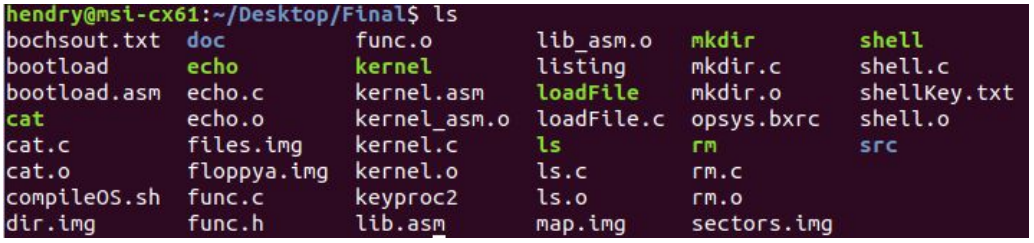



**2.** Membuat program utilitas **`mkdir`**

Program mkdir membuat sebuah direktori baru pada suatu direktori. Pengguna memberikan satu argumen yaitu nama direktori yang ingin dibuat.

**3.** Membuat program utilitas **`ls`**

Program ls mendaftarkan nama-nama semua file dan direktori yang berada pada *current directory*. Berikut adalah ls pada linux:




**4.** Membuat program utilitas **`rm`**

Program rm menghapus suatu *file* atau direktori. Pengguna memberikan satu argumen yaitu *path* relatif *file* atau direktori yang ingin dihapus.

**5.** Membuat program utilitas **`cat`**

Program cat mencetak isi suatu file ke layar. Pengguna memberikan satu argumen yaitu nama *file* yang ingin dicetak ke layar. Jika pengguna memberikan argumen kedua '-w', maka program cat akan berubah ke write mode, yang mana akan dibuat file baru dengan isi input user. Format lengkapnya "cat <path> <command>"

**D.** Kompilasi dan Loading File

Setelah implementasi selesai dibuat, keseluruhan program akan dikompilasi. Kemudian, hasil kompilasi akan diload ke Sistem Operasi yang telah dibuat. Sehingga, program-program dapat dijalankan pada shell.

```
# "Kompilasi kernel..."
bcc -ansi -c -o kernel.o kernel.c
as86 kernel.asm -o kernel_asm.o
ld86 -o kernel -d kernel.o kernel_asm.o
dd if=kernel of=floppya.img bs=512 conv=notrunc seek=1 iflag=fullblock
as86 lib.asm -o lib_asm.o

# "Kompilasi fungsi-fungsi pembantu..."
bcc -ansi -c -o func.o func.c

# "Kompilasi shell dan fitur utilitas..."
bcc -ansi -c shell.o shell.c
ld86 -o shell -d shell.o lib_asm.o func.o

bcc -ansi -c ls.o ls.c
ld86 -o ls -d ls.o lib_asm.o func.o

bcc -ansi -c echo.o echo.c
ld86 -o echo -d echo.o lib_asm.o func.o

bcc -ansi -c mkdir.o mkdir.c
ld86 -o mkdir -d mkdir.o lib_asm.o func.o

bcc -ansi -c rm.o rm.c
ld86 -o rm -d rm.o lib_asm.o func.o

bcc -ansi -c cat.o cat.c
ld86 -o cat -d cat.o lib_asm.o func.o

#loadFile
./loadFile shell
./loadFile ls
./loadFile echo
./loadFile mkdir
./loadFile rm
./loadFile cat
./loadFile keyproc2
```

Jalankan Sistem Operasi dengan emulator bochs.

## 2. Pembagian Tugas

| NIM | Nama | Pekerjaan | Persentase |
|---|---|---|---|
| 13517033 | Harry Rahmadi Munly | Laporan,mkdir,echo, rm | 25% |
| 13517105 | M. Hendry Prasetya | Kernel.c,laporan,debugging,cat | 50% |
| 13517150 | Ridwan Faturrahman | Laporan,ls,shell | 25% |

## 3. Kesulitan Pengerjaan

Dalam mengerjakan tugas ini, kami mendapatkan cukup banyak kesulitan. Bagi kami, membuat sebuah sistem operasi merupakan hal baru. Tidak hanya baru, hal tersebut juga sulit untuk kami pelajari. Berbeda dengan sistem operasi milestone pertama, milestone kali ini menantang kami untuk membuat program-program utilitas dan filesystem yang berbeda.