

**Laporan Tugas Besar 2 IF 2230 Sistem Operasi**  
**Pembuatan Sistem Operasi Sederhana**  
**Hierarchical File System, Shell, Utility Programs**

Laporan ini disusun untuk memenuhi tugas mata kuliah IF2230 Sistem Operasi

Disusun oleh:

Aditya Putra Santosa	13517013
Ricky Yuliawan	13517025
Hafidh Rendyanto	13517061



PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG

2019

# DAFTAR ISI

	Halaman
Daftar Isi .....	ii
Langkah Pengerjaan .....	1
Pembagian Tugas .....	15
Kesulitan .....	16

# LANGKAH Pengerjaan

## 1. Mengubah Kernel Sistem Operasi

### a. Mengubah struktur *file system* menjadi *hierarchical file system*

Berbeda dengan struktur *file system* OS sebelumnya yang mana setiap berkas berada pada satu direktori root saja, struktur *file system* yang akan Anda implementasikan 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.

Sebelumnya, struktur dari file system anda adalah seperti berikut:

- **map** (di sektor 0x1)
- **dir** (di sektor 0x2)
- **kernel** (di sektor 0x3-0xC)

Struktur dari sektor **dir** sebelumnya:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Nama file 0 (12 karakter)															
Sektor file 0 (20 sektor)															
Nama file 1 (12 karakter)															
Sektor file 1 (20 sektor)															
...															
Nama file 15 (12 karakter)															
Sektor file 15 (20 sektor)															

Setelah diubah, struktur dari file system anda akan seperti berikut:

- **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)

Struktur dari sektor **dirs**:

idx	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	P	Nama direktori 0 (15 karakter)														
01	P	Nama direktori 1 (15 karakter)														

...

1F	P	Nama direktori 31 (15 karakter)														
----	---	---------------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--

P : indeks direktori parent (0x00-0x1F) dari direktori tersebut. 0xFF jika parentnya root.

Struktur dari sektor **files**:

idx	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	P	Nama file 0 (15 karakter)														
01	P	Nama file 1 (15 karakter)														

...

1F	P	Nama file 31 (15 karakter)														
----	---	----------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--

P : indeks direktori parent (0x00-0x1F) dari file tersebut. 0xFF jika parentnya root.

Struktur dari sektor **sectors**:

idx	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Sektor file 0 (16 sektor)															
01	Sektor file 0 (16 sektor)															

...

1F	Sektor file 0 (16 sektor)															
----	---------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Pada kit milestone 2 sudah tersedia **bootload.asm**, **loadFile.c**, **map.img**, **files.img**, dan **sectors.img** yang telah disesuaikan untuk struktur *file system* yang baru. Salinlah file-file tersebut ke direktori sistem operasi kalian.

Selain itu, ubahlah bagian penulisan sektor awal pada `compileOS.sh` kalian sehingga menjadi seperti berikut:

```
dd if=/dev/zero of=floppya.img bs=512 count=2880
dd if=bootload of=floppya.img bs=512 count=1 conv=notrunc
dd if=map.img of=floppya.img bs=512 count=1 seek=256
conv=notrunc
dd if=files.img of=floppya.img bs=512 count=1 seek=258
conv=notrunc
```

```
dd if=sectors.img of=floppya.img bs=512 count=1 seek=259
conv=notrunc
```

b. Mengubah fungsi handleInterrupt21

Karena keperluan beberapa syscall setelah ini, parameter AX dari fungsi handleInterrupt21 akan dipisah menjadi AL dan AH.

```
void handleInterrupt21 (int AX, int BX, int CX, int DX) {
    char AL, AH;
    AL = (char) (AX);
    AH = (char) (AX >> 8);

    switch (AL) {
        case 0x00:
            printString(BX);
            break;
        case 0x01:
            readString(BX);
            break;
        case 0x02:
            readSector(BX, CX);
            break;
        case 0x03:
            writeSector(BX, CX);
            break;
        case 0x04:
            readFile(BX, CX, DX, AH);
            break;
        case 0x05:
            writeFile(BX, CX, DX, AH);
            break;
        case 0x06:
            executeProgram(BX, CX, DX, AH);
            break;
        case 0x07:
            terminateProgram(BX);
            break;
        case 0x08:
            makeDirectory(BX, CX, AH);
            break;
        case 0x09:
            deleteFile(BX, CX, AH);
            break;
        case 0x0A:
            deleteDirectory(BX, CX, AH);
            break;
    }
}
```

```

        case 0x20:
            putArgs(BX, CX);
            break;
        case 0x21:
            getCurdir(BX);
            break;
        case 0x22:
            getArgc(BX);
            break;
        case 0x23:
            getArgv(BX, CX);
            break;
        default:
            printString("Invalid interrupt");
    }
}

```

Untuk pemanggilan interrupt, gunakan format kode berikut:

```

interrupt(0x21, (AH << 8) | AL, BX, CX, DX);

```

#### c. Mengubah implementasi syscall readFile

Ubah syscall readFile 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 dan memiliki function signature sebagai berikut:

```

void readFile(char *buffer, char *path, int *result, char
parentIndex)

```

Untuk membaca file “g” dengan path relatif “abc/def/g” dapat dilakukan langkah-langkah berikut:

- 1) Cari indeks direktori pada sektor dirs yang bernama “abc” dan indeks parent-nya adalah adalah parentIndex (0xFF jika root). Jika tidak ditemukan, syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.
- 2) Cari indeks direktori pada sektor dirs yang bernama “def” dan indeks parent-nya adalah indeks direktori “abc” yang telah didapatkan sebelumnya. Jika tidak ditemukan, syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.
- 3) Cari indeks file pada sektor files yang bernama “g” dan indeks parentnya adalah indeks direktori “def” yang telah didapatkan sebelumnya. Jika tidak ditemukan, syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.

- 4) Baca daftar sektor file pada entri file dengan indeks tersebut pada sektor sectors.
- 5) Masukkan sektor-sektor tersebut ke buffer.
- 6) Kembalikan success (0) pada parameter result.

d. Mengubah implementasi syscall writeFile

Ubah syscall writeFile 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 dan memiliki function signature sebagai berikut:

```
void writeFile(char *buffer, char *path, int *sectors,
char parentIndex)
```

Untuk menulis file “g” dengan *path* relatif “abc/def/g” dapat dilakukan langkah-langkah berikut:

- 1) Cari indeks direktori pada sektor **dirs** yang bernama “abc” dan indeks *parent*-nya adalah **parentIndex** (0xFF jika root). Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter result.
- 2) Cek apakah jumlah sektor kosong cukup pada sektor **map**. Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* INSUFFICIENT\_SECTORS (0) pada parameter **sectors**.
- 3) Cek apakah masih tersisa entri kosong pada sektor **files**. Jika tidak ada entri yang kosong, *syscall* akan gagal dan mengembalikan *error* INSUFFICIENT\_ENTRIES (-3) pada parameter **sectors**.  
**Tips:** ciri-ciri entri kosong yaitu byte pertama dari nama filenya adalah karakter NUL ('\0').
- 4) Cari indeks direktori pada sektor **dirs** yang bernama “abc” dan indeks *parent*-nya adalah **parentIndex** (0xFF jika root). Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter **sectors**.
- 5) Cari indeks direktori pada sektor **dirs** yang bernama “def” dan indeks *parent*-nya adalah indeks direktori “abc” yang didapatkan sebelumnya. Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter **sectors**.
- 6) Cari indeks file pada sektor files yang bernama “g” dan indeks *parent*-nya adalah indeks direktori “def” yang didapatkan sebelumnya. Jika ditemukan, *syscall* akan gagal dan mengembalikan *error* ALREADY\_EXISTS (-2) pada parameter **sectors**.

- 7) Pada entri kosong pertama pada sektor files tulis indeks dari direktori “**def**” yang didapatkan sebelumnya pada byte indeks parent dan “**g**” pada byte-byte nama filenya.
- 8) Cari sektor kosong pertama pada sektor map dan tandai byte sektor tersebut menjadi terisi (0xFF).
- 9) Tulis data dari **buffer** ke sektor dengan nomor tersebut.
- 10) Lakukan langkah 7 dan 8 untuk setiap sektor dari file.

e. Mengubah implementasi syscall executeProgram

Ubah syscall executeProgram sehingga menerima path relatif dari program yang akan dieksekusi. Anda tidak perlu mengubah banyak karena perubahan utama sudah ditangani oleh readFile.

Syscall ini dipanggil melalui interrupt 0x21 AX=0x06 dan memiliki function signature sebagai berikut:

```
void executeProgram(char *path, int segment, int *result,
char parentIndex)
```

f. Implementasi syscall terminateProgram

Buat syscall terminateProgram dengan mengeksekusi syscall executeProgram yang mengembalikan eksekusi program ke shell sehingga seakan-akan mengakhiri eksekusi program sebelumnya. Syscall ini dipanggil setiap akhir program. Berikut kode implementasinya:

```
void terminateProgram (int *result) {
    char shell[6];
    shell[0] = 's';
    shell[1] = 'h';
    shell[2] = 'e';
    shell[3] = 'l';
    shell[4] = 'l';
    shell[5] = '\0';
    executeProgram(shell, 0x2000, result, 0xFF);
}
```

Syscall ini dipanggil melalui interrupt 0x21 AL=0x07.

g. Implementasi syscall makeDirectory

Buatlah syscall makeDirectory yang menerima path relatif dari direktori yang akan dibuat, contohnya “abc/def/ghi”.



Syscall ini dipanggil melalui interrupt 0x21 AL=0x08 dan memiliki function signature sebagai berikut:

```
void makeDirectory(char *path, int *result, char
parentIndex)
```

Untuk membuat direktori “ghi” dengan path relatif “abc/def/ghi” dapat dilakukan langkah-langkah berikut:

- 1) Cek apakah masih tersisa entry kosong pada sektor dirs (ciri-ciri entry kosong adalah byte pertama dari nama direktorinya adalah karakter NUL ‘\0’). Jika tidak ada maka syscall akan gagal dan mengembalikan error INSUFFICIENT\_ENTRIES (-3) pada parameter result.
- 2) Cari indeks direktori pada sektor dirs yang bernama “abc” dan indeks parent-nya adalah parentIndex (0xFF jika root). Jika tidak ada maka syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.
- 3) Cari indeks direktori pada sektor dirs yang bernama “def” dan indeks parent-nya adalah indeks direktori “abc” yang didapatkan sebelumnya. Jika tidak ada maka syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.
- 4) Cari indeks file pada sektor files yang bernama “ghi” dan indeks parentnya adalah indeks direktori “def” yang didapatkan sebelumnya. Jika ada maka syscall akan gagal dan mengembalikan error ALREADY\_EXISTS (-2) pada parameter result.
- 5) Pada entri kosong pertama pada sektor dirs, tulis indeks dari direktori “def” yang didapatkan sebelumnya pada byte indeks parent dan “ghi” pada byte-byte nama direktorinya.
- 6) Kembalikan success (0) pada parameter result.

#### h. Implementasi syscall deleteFile

Buat syscall deleteFile yang menerima path relatif dari file yang akan dihapus, contohnya “**abc/def/g**”. Syscall ini dipanggil melalui interrupt 0x21 AL=0x09 dan memiliki function signature sebagai berikut:

```
void deleteFile(char *path, int *result, char
parentIndex)
```

Untuk menghapus file “g” dengan path relatif “**abc/def/g**” dapat dilakukan langkah-langkah berikut:

- 1) Cari indeks direktori pada sektor **dirs** yang bernama “**abc**” dan indeks *parent*-nya adalah **parentIndex** (0xFF jika root). Jika tidak ditemukan,

*syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter *result*.

- 2) Cari indeks direktori pada sektor **dirs** yang bernama “**def**” dan indeks *parent*-nya adalah indeks direktori “**abc**” yang telah diperoleh sebelumnya. Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter *result*.
- 3) Cari indeks file pada sektor **files** yang bernama “**g**” dan indeks *parent*-nya adalah indeks direktori “**def**” yang didapatkan sebelumnya. Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter *result*.
- 4) Pada entri *file* dengan indeks tersebut pada sektor **files**, ubah byte pertama nama *file*-nya menjadi karakter NUL ('\0').
- 5) Baca daftar sektor file pada entri *file* dengan indeks tersebut pada sektor **sectors**.
- 6) Mark byte sektor-sektor tersebut menjadi kosong (0x00) pada sektor **map**.
- 7) Kembalikan success (0) pada parameter **result**.

i. Implementasi *syscall* deleteDirectory

Buat *syscall* deleteDirectory yang 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 dan memiliki function signature sebagai berikut:

```
void deleteDirectory(char *path, int *success, char
parentIndex)
```

Untuk menghapus direktori “**ghi**” dengan path relatif “**abc/def/ghi**” dapat dilakukan langkah-langkah berikut:

- 1) Cari indeks direktori pada sektor **dirs** yang bernama “**abc**” dan indeks *parent*-nya adalah **parentIndex** (0xFF jika root). Jika tidak ditemukan, maka *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter *result*.
- 2) Cari indeks direktori pada sektor **dirs** yang bernama “**def**” dan indeks *parent*-nya adalah indeks direktori “**abc**” yang didapatkan sebelumnya. Jika tidak ditemukan, *syscall* akan gagal dan mengembalikan *error* NOT\_FOUND (-1) pada parameter *result*.
- 3) Cari indeks direktori pada sektor **dirs** yang bernama “**ghi**” dan indeks *parent*-nya adalah indeks direktori “**def**” yang didapatkan sebelumnya.

Jika tidak ditemukan, maka syscall akan gagal dan mengembalikan error NOT\_FOUND (-1) pada parameter result.

- 4) Pada entri direktori dengan indeks tersebut pada sektor **dirs**, ubah byte pertama nama direktorinya menjadi karakter NUL ('\0').
- 5) Cari indeks dari semua *file* yang indeks *parent*-nya adalah indeks direktori “**def**” pada sektor files. Hapus *file-file* tersebut menggunakan langkah 4-6 *syscall* deleteFile. Anda dapat memisahkan langkah-langkah tersebut menjadi fungsi deleteFile terpisah yang menerima indeks file langsung.
- 6) Cari indeks dari semua direktori yang indeks parentnya adalah indeks direktori “**def**” pada sektor **dirs**. Hapus direktori-direktori tersebut menggunakan langkah 4-6 dari *syscall* deleteDirectory (termasuk langkah ini). Anda dapat memisahkan langkah-langkah tersebut menjadi fungsi deleteDirectory terpisah yang menerima indeks direktori langsung.
- 7) Kembalikan success (0) pada parameter result.

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

Gunakan kode di bawah untuk implementasinya:

```
#define ARGS_SECTOR 512

void putArgs (char curdir, char argc, char **argv) {
    char args[SECTOR_SIZE];
    int i, j, p;
    clear(args, SECTOR_SIZE);

    args[0] = curdir;
    args[1] = argc;
    i = 0;
    j = 0;
    for (p = 1; p < ARGS_SECTOR && i < argc; ++p) {
        args[p] = argv[i][j];
        if (argv[i][j] == '\0') {
            ++i;
            j = 0;
        }
    }
}
```

```

        else {
            ++j;
        }
    }

    writeSector(args, ARGS_SECTOR);
}

void getCurdir (char *curdir) {
    char args[SECTOR_SIZE];
    readSector(args, ARGS_SECTOR);
    *curdir = args[0];
}

void getArgc (char *argc) {
    char args[SECTOR_SIZE];
    readSector(args, ARGS_SECTOR);
    *argc = args[1];
}

void getArgv (char index, char *argv) {
    char args[SECTOR_SIZE];
    int i, j, p;
    readSector(args, ARGS_SECTOR);

    i = 0;
    j = 0;
    for (p = 1; p < ARGS_SECTOR; ++p) {
        if (i == index) {
            argv[j] = args[p];
            ++j;
        }

        if (args[p] == '\\0') {
            if (i == index) {
                break;
            }
            else {
                ++i;
            }
        }
    }
}

```

```

void putArgs(char curdir, char argc, char argv[64][128])
{
    char args[SECTOR_SIZE];
    int i, j, p;
    clear(args, SECTOR_SIZE);

    args[0] = curdir;
    args[1] = argc;
    i = 0;
    j = 0;

    for (p = 2; p < ARGS_SECTOR && i < argc; ++p)
    {
        args[p] = argv[i][j];
        if (argv[i][j] == '\\0')
        {
            ++i;
            j = 0;
        }
        else
        {
            ++j;
        }
    }
    writeSector(args, ARGS_SECTOR);
}

```

(Perubahan pada fungsi putArgs agar berjalan dengan benar (nilai awal p & tipe parameter argv))

```

void getArgv(char index, char *argv)
{
    char args[SECTOR_SIZE];
    int i, j, p;
    readSector(args, ARGS_SECTOR);
    i = 0;
    j = 0;
    for (p = 2; p < ARGS_SECTOR; ++p)
    {
        if (i == index)
        {
            argv[j] = args[p];
            ++j;
        }

        if (args[p] == '\\0')
        {
            if (i == index)
            {
                break;
            }
            else
            {
                ++i;
            }
        }
    }
}

```

(Perubahan pada fungsi getArgv agar berjalan dengan benar (nilai awal p))

## 2. Membuat Shell Sistem Operasi

### a. Persiapan program shell

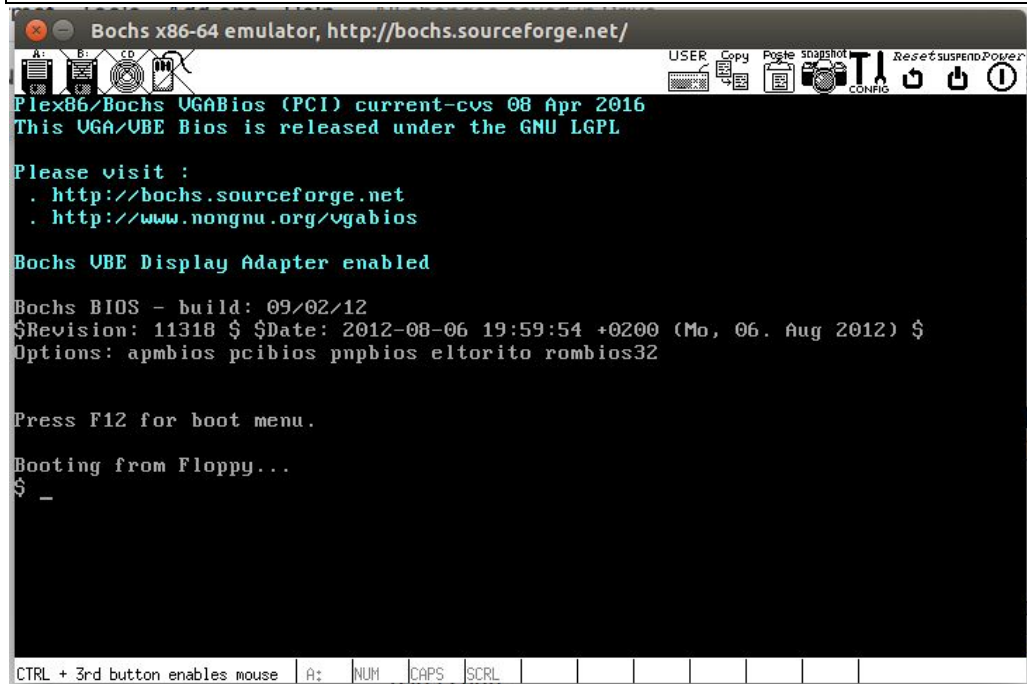
Anda sekarang diminta untuk membuat program shell untuk sistem operasi anda. Buatlah sebuah file source code C bernama shell.c. Pada fungsi main kernel.c, panggil

```
interrupt(0x21, 0xFF << 8 | 0x6, "shell", 0x2000,
&success);
```

untuk menjalankan shell setiap kali kernel dijalankan.

Program shell akan menampilkan \$ pada layar dan menunggu input dari pengguna seperti di bawah:

\$



(Tampilan Shell saat baru mulai)

### b. 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. Tambahan, cd jika dilakukan tanpa parameter akan langsung membawa pengguna ke root folder (seperti linux), jika dilakukan dengan parameter “..” akan membawa user ke parent folder dari directory sekarang



```
$ cd test
Go to : test
$ ls
Isi :
  Folder :
    a
    ab
  File :
    f1
$ CD
CD not found
$ cd
$ ls
Isi :
  Folder :
    test
  File :
    KERNEL
    shell
    echoUtil
    mkdir
    ls
    cat
    keyproc2
$
```

(Contoh penggunaan cd)

c. Implementasi perintah menjalankan program

Hal yang harus ditangani oleh shell adalah perintah dari pengguna untuk menjalankan program. Misal, jika pengguna memasukkan hal seperti di bawah pada shell:

```
$ myprog
```

Gunakan syscall `executeProgram` untuk menjalankan program dengan nama “**myprog**” pada root directory. Untuk menjalankan program pada current directory, pengguna menggunakan perintah seperti di bawah:

```
$ ./myprog
```

Sintaks ‘./’ menandakan shell untuk mencari program pada *current directory*. Pengguna juga dapat memasukkan argumen-argumen (dipisah dengan spasi) untuk program, misal:

```
$ ./myprog abc 123
```

Sebelum `executeProgram` dijalankan, gunakan terlebih dahulu `setArgs` untuk menyimpan direktori sekarang (`curdir`), jumlah argumen (`argc`), dan nilai-nilai argumen (`argv`). Contoh penggunaan dapat dilihat di bawah:

```
char curdir;
char argc;
char *argv[2];
```

```

curdir = 0xFF; //root
argc = 2;
argv[0] = "abc";
argv[1] = "123";
interrupt(0x21, 0x20, curdir, argc, argv);

```

Untuk melakukan *get* argumen program, gunakan *syscall* *getCurdir*, *getArgc* dan *getArgv*. Contoh penggunaannya dapat dilihat sebagai berikut:

```

int main() {
    int i;
    char curdir;
    char argc;
    char argv[4][16];

    interrupt(0x21, 0x21, &curdir, 0, 0);
    interrupt(0x21, 0x22, &argc, 0, 0);
    for (i = 0; i < argc; ++i) {
        interrupt(0x21, 0x23, i, argv[i], 0);
    }
}

```

```

Bochs x86-64 emulator, http://bochs.sourceforge.net/
Press F12 for boot menu.
Booting from Floppy...
$ ls
Isi :
  Folder :
  File :
    KERNEL
    shell
    echoUtil
    mkdir
    ls
    cat
    keyproc2
$ keyproc2
keyproc2 requires 2 parameters
keyproc2 <kelas> <kelompok>
kelas: Nomor kelas anda (01-03)
kelompok: Nomor kelompok anda (01-20)
$ ./keyproc2
keyproc2 requires 2 parameters
keyproc2 <kelas> <kelompok>
kelas: Nomor kelas anda (01-03)
kelompok: Nomor kelompok anda (01-20)
$
IPS: 63,666M

```

(Contoh penggunaan run program pada current directory & root directory)



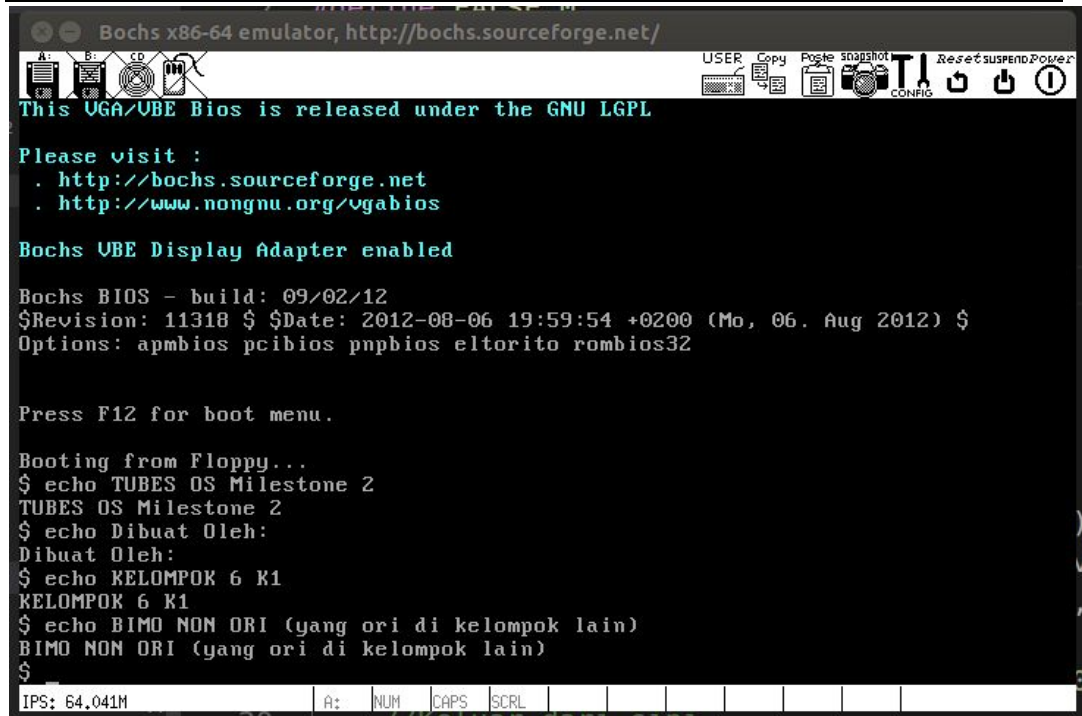
### 3. Membuat Program Utilitas

#### a. Membuat program echo

Program echo mencetak parameter pertama dan seterusnya yang diberikan user ke layar.

Contoh :

```
$ echo Hellooooo Worldddd
Hellooooo Worldddd
```

The image shows a screenshot of a Bochs x86-64 emulator window. The title bar reads "Bochs x86-64 emulator, http://bochs.sourceforge.net/". The window contains a BIOS boot screen with the following text: "This UGA/UBE Bios is released under the GNU LGPL", "Please visit :", ". http://bochs.sourceforge.net", ". http://www.nongnu.org/ugabios", "Bochs UBE Display Adapter enabled", "Bochs BIOS - build: 09/02/12", "\$Revision: 11318 \$ \$Date: 2012-08-06 19:59:54 +0200 (Mo, 06. Aug 2012) \$", "Options: apmbios pcibios pnpbios eltorito rombios32", "Press F12 for boot menu.", "Booting from Floppy...", "\$ echo TUBES OS Milestone 2", "TUBES OS Milestone 2", "\$ echo Dibuat Oleh:", "Dibuat Oleh:", "\$ echo KELOMPOK 6 K1", "KELOMPOK 6 K1", "\$ echo BIMO NON ORI (yang ori di kelompok lain)", "BIMO NON ORI (yang ori di kelompok lain)", "\$ ". At the bottom, there is a status bar showing "IPS: 64,041M" and a keyboard layout indicator with "A:", "NUM", "CAPS", "SCRL", and several empty slots.

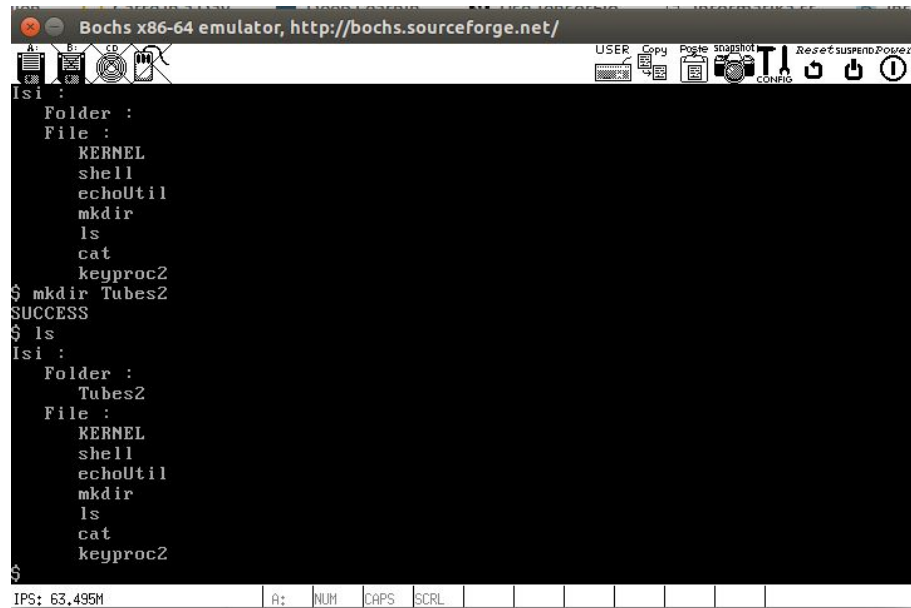
(Contoh penggunaan echo)

#### b. Membuat program mkdir

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

#### c. Membuat program ls

Program ls mendaftarkan nama-nama semua file dan direktori yang berada pada current directory. Format penampilan dibebaskan.

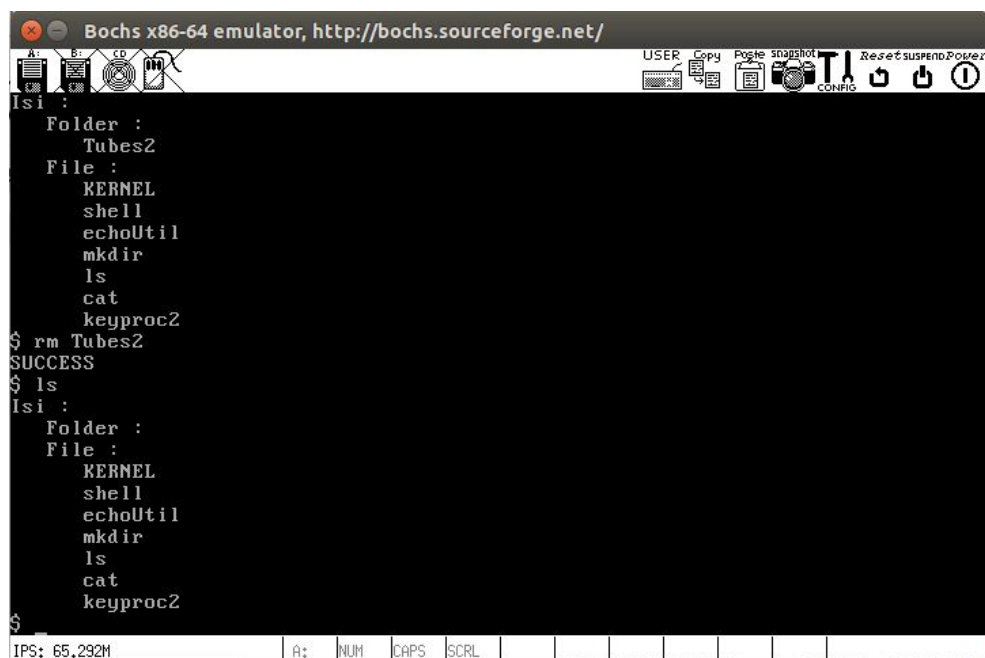


```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Isi :
Folder :
File :
  KERNEL
  shell
  echoUtil
  mkdir
  ls
  cat
  keyproc2
$ mkdir Tubes2
SUCCESS
$ ls
Isi :
Folder :
  Tubes2
File :
  KERNEL
  shell
  echoUtil
  mkdir
  ls
  cat
  keyproc2
$
```

(Contoh penggunaan command ls dan mkdir)

d. Membuat program rm

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



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Isi :
Folder :
  Tubes2
File :
  KERNEL
  shell
  echoUtil
  mkdir
  ls
  cat
  keyproc2
$ rm Tubes2
SUCCESS
$ ls
Isi :
Folder :
File :
  KERNEL
  shell
  echoUtil
  mkdir
  ls
  cat
  keyproc2
$
```

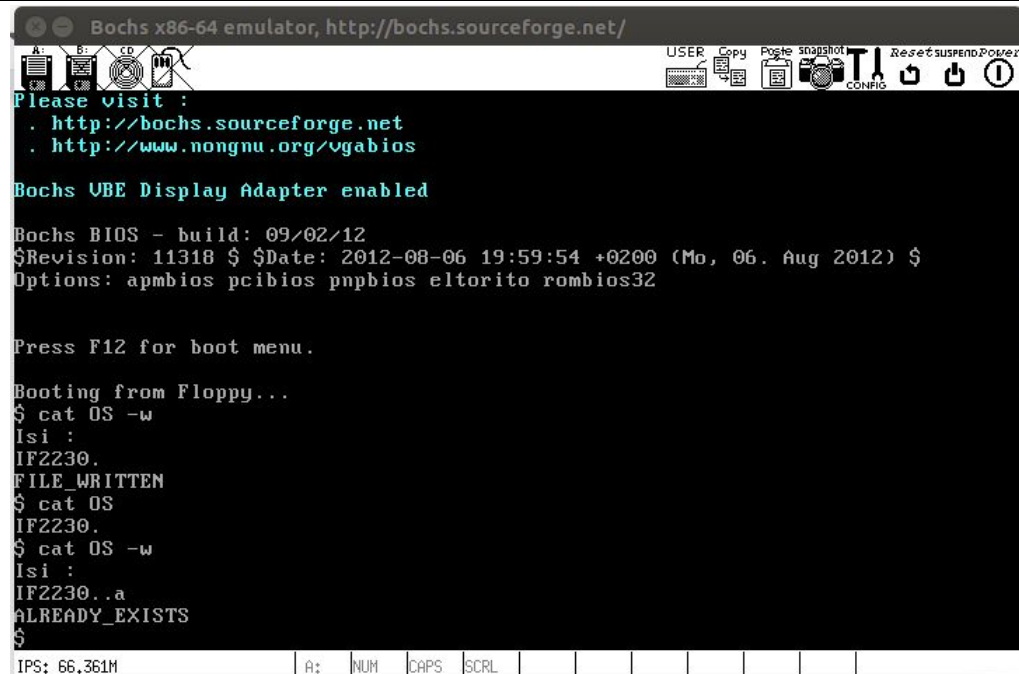
(Contoh penggunaan command rm)

- e. Membuat program 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. Detail implementasi dibebaskan.

Contoh :

```
$ cat newfolder2/aha -w
Folder tidak ditemukan
$ cat newfolder/aha -w
Masukkan teks: Helloooo Worldddddd
$ cat newfolder/aha
Helloooo Worldddddd
```



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios
Bochs VBE Display Adapter enabled
Bochs BIOS - build: 09/02/12
$Revision: 11318 $ $Date: 2012-08-06 19:59:54 +0200 (Mo, 06. Aug 2012) $
Options: apmbios pcibios pnpbios eltorito rombios32
Press F12 for boot menu.
Booting from Floppy...
$ cat OS -w
Isi :
IF2230.
FILE WRITTEN
$ cat OS
IF2230.
$ cat OS -w
Isi :
IF2230..a
ALREADY_EXISTS
$
IPS: 66,361M  A: NUM CAPS SCRL
```

(Contoh penggunaan cat)

#### 4. Menjalankan Program Pengujian

Untuk menjalankan program pengujian, diperlukan aspek-aspek berikut dari sistem operasi anda sudah berfungsi dengan benar sesuai spesifikasi tugas:

- readFile (interrupt 0x21 AL=0x04)
- writeFile (interrupt 0x21 AL=0x05)
- executeProgram (interrupt 0x21 AL=0x06)
- terminateProgram (interrupt 0x21 AL=0x07)
- makeDirectory (interrupt 0x21 AL=0x08)
- Shell sistem operasi

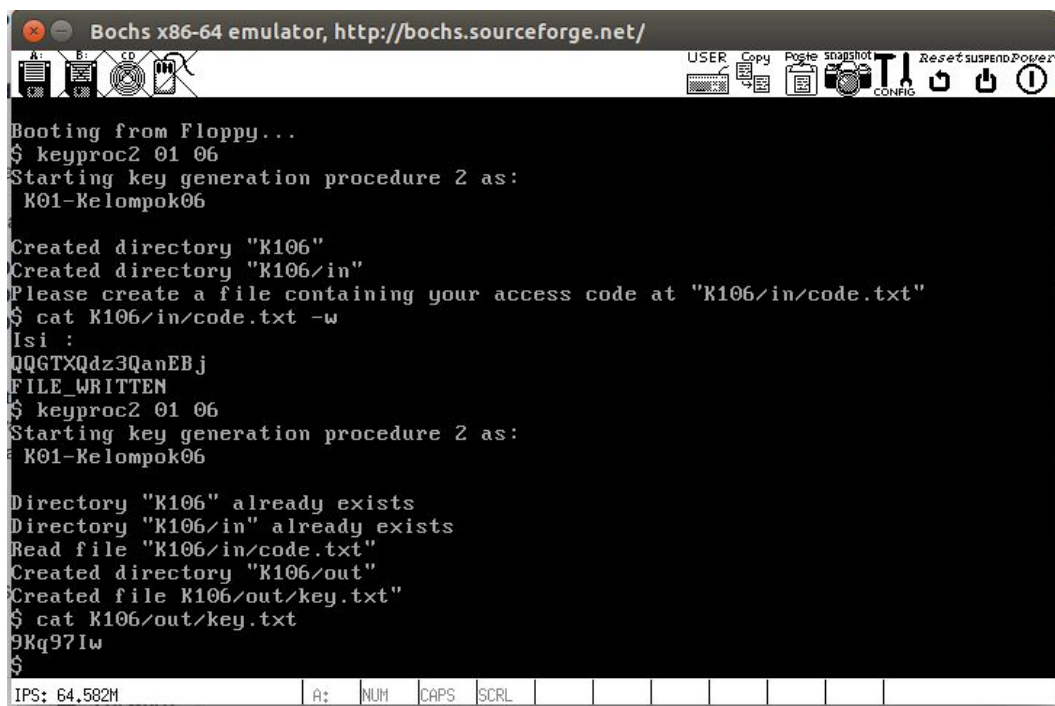
- Program utilitas ls
- Program utilitas cat

Langkah-langkah untuk menjalankan program pengujian adalah sebagai berikut:

- 1) Download archive .zip yang berisi program pengujian yang telah disebar di milis.
- 2) Masukkan file program “keyproc2” yang di dalam archive tersebut ke root directory sistem operasi anda dengan perintah.

```
./loadFile keyproc2
```

- 3) Pastikan bahwa program “**keyproc2**” sudah ada di root directory sistem operasi anda dengan mengeksekusikan program “**ls**” yang anda sudah buat pada shell anda.
- 4) Jalankan program “**keyproc2**”. Program “keyproc2” membutuhkan 2 parameter, yaitu kelas dan kelompok anda. Jalankan kembali program “keyproc2” dengan parameter-parameter tersebut.
- 5) Lakukan apa yang ditampilkan pada layar oleh program dengan menggunakan program “**cat**” yang anda sudah buat dengan parameter pertama path yang dispesifikasikan dan parameter kedua flag “-w” untuk masuk ke write mode. Isi file tersebut dengan access code kelompok anda.
- 6) Jalankan kembali program “**keyproc**” dengan parameter sebelumnya.
- 7) Gunakan program “cat” kembali untuk membaca file hasil pada path yang dispesifikasikan.



```
Bochs x86-64 emulator, http://bochs.sourceforge.net/
Bootling from Floppy...
$ keyproc2 01 06
Starting key generation procedure 2 as:
K01-Kelompok06

Created directory "K106"
Created directory "K106/in"
Please create a file containing your access code at "K106/in/code.txt"
$ cat K106/in/code.txt -w
Isi :
QQGTXXdz3QanEBj
FILE_WRITTEN
$ keyproc2 01 06
Starting key generation procedure 2 as:
K01-Kelompok06

Directory "K106" already exists
Directory "K106/in" already exists
Read file "K106/in/code.txt"
Created directory "K106/out"
Created file K106/out/key.txt
$ cat K106/out/key.txt
9Kq97Iw
$
IPS: 64.582M  A: NUM CAPS SCRL
```

(Hasil jalan program pengujian keyproc2)

## 5. Bonus

a. Membuat program mv

Program mv memindahkan file atau direktori argumen pertama menjadi file atau direktori argumen kedua. Fungsionalitas seperti mv pada Linux.

b. Membuat program cp

Program mv menyalin file atau direktori argumen pertama ke file atau direktori argumen kedua. Fungsionalitas seperti cp pada Linux.

c. Lain-lain

Anda diperbolehkan untuk mengimplementasikan fitur-fitur lain pada sistem operasi kalian. Kreativitas Anda akan dihargai dengan nilai bonus.

## PEMBAGIAN TUGAS

NIM	Nama	Bagian Kerja	Persentase Kontribusi
13517013	Aditya Putra S.	Kernel, Shell	33.333%
13517025	Ricky Yuliawan	Laporan	33.333%
13517061	Hafidh Rendyanto	Kernel, Shell	33.333%

## KESULITAN

- Bosc-x tidak disupport untuk distro fedora
- Ukuran sektor per file yang kecil mengharuskan kami untuk membuat kernel yang jika di compile menghasilkan instruksi assembly sesedikit mungkin. Sehingga tidak bisa memuat terlalu banyak string di kernel karena hasil byte code nya jadi besar
- Terkadang terdapat memory leak, dimana suatu string menimpa bagian program / file lain sehingga terjadi hal yang tidak diinginkan seperti isi file beda, shell tidak jalan lagi karena corrupted.

## FEEDBACK

Secara keseluruhan, tugasnya asik dan membantu kami mengerti tentang hierarchical file system linux. Namun fungsionalitas yang banyak membuat tugas ini lama dan susah dikerjakan.