

# **Tugas Besar**

## **IF2230 - Sistem Operasi**

**Milestone 2 of ??**

*"Ho/OS"*

**Pembuatan Sistem Operasi x86  
Interrupt, Driver, dan Filesystem**

Dipersiapkan oleh  
Asisten Lab Sistem Terdistribusi



**Waktu Mulai**

Jum'at, 3 Maret 2023, 18.00 WIB

**Waktu Akhir**

Kamis, 30 Maret 2023, 23.59 WIB

# I. Latar Belakang

Setelah sehari-hari terjebak di dunia bit yang berantakan, kamu mulai berhasil mengendalikan bit-bit yang ada di dunia tersebut. Bersama Docchi, kamu melewati halangan dan rintangan yang ada, bagaikan seorang karakter utama dan seorang sidekick. Bukan hanya itu, kamu juga memperhatikan Docchi sepertinya mulai merasa nyaman di dunia tersebut, ~~bukan nyaman sama kamu ya, jangan geer.~~ Ia sedikit demi sedikit membantu kamu ~~bagaikan kopilot.~~



Meskipun demikian, kamu belum melihat tanda-tanda akhir dari petualanganmu di dunia bit ini. Sebaliknya, kompleksitas dunia tersebut terlihat semakin rumit, begitu pula halangan dan rintangan yang dilewati.

"Duh, kelarnya kapan sih aaaaAaaaaAaa?!?" Kamu mengungkapkan rasa malding kamu dengan teriakan, yang tentunya disambut dengan gema dari suara kamu sendiri. "U-U-Uhhh, aku juga t-t-tidak t-t-t-tahu...", balas Docchi yang juga kebingungan. Bagaimana tidak, kamu sudah terlalu lama di dunia bit ini hingga jebrut (jenuh brutal).

Tidak lama setelah kamu meneriakkan kemaldingan kamu, terdengar suara gadis misterius yang masih kamu ingat, ya tentu saja kamu ingat, kan kamu--"

"Hmph, bisa juga kamu." Sebuah pujian bernada sinis yang bergema di dunia tersebut. Ini adalah yang ke sekian kalinya gadis misterius tersebut memuji kemampuan kamu, dimulai dari konfrontasi kamu dengan Pa Una-

Ya, Pa Una. Gadis satu lagi yang hampir kamu lupakan eksistensinya ~~karena sekarang lagi Docchi Are~~. Kamu teringat kalau seharusnya kamu mulai mencari Pa Una, mengingat kamu sudah cukup mampu bernavigasi di dunia ini... ya, kalau memang Pa Una memang juga ada di dunia ini.

"Kataku, mending neng jelasin semua ini!" Kamu sedikit berteriak, berusaha berkomunikasi dengan gadis misterius tadi. Sayangnya, tidak ada jawaban. Apakah ia tidak di sini, atau ia memang tidak mau menjawabmu, tidak ada yang tahu.

"N-n-neng? N-neng siapa?" Docchi mempertanyakan gadis misterius itu. Kamu menyadari kalau Docchi tidak tahu cerita bagaimana kamu masuk ke dunia ini, meskipun kalian sudah berhari-hari terjebak di dunia itu. "Oh ya, kamu belum tahu ya, jadi gini..." kamu mulai bercerita panjang lebar tentang bagaimana kamu sampai di dunia itu, bagaimana cerita kamu yang awalnya hanya ingin kerja di Uncover Corps untuk ~~bertemu Pekola~~ mengasah skill dan menambah pengalaman, kemudian bertemu Pa Una, terjebak di labirin bertema halowin, dan lain sebagainya. Docchi terlihat manut-manut saja, tapi dari raut wajahnya kamu menyimpulkan Docchi menikmati cerita itu, mungkin sekarang Docchi mulai melihat kamu sebagai sosok yang keren...? Ntahlah, gausah mimpi ya.

"...jadi gitu." Kamu mengakhiri cerita perjalanan kamu sampai tenggorokan kamu kering. Kamu melihat ke arah Docchi yang sepertinya ingin bercerita juga. "A-A-anu..." "Hm? Kenapa?" Kamu mencoba menggunakan kemampuan komunikasi kamu yang sudah kamu asah ~~setelah bertahun-tahun menonton anime harem dan mencontoh gaya berkomunikasi karakter utamanya~~ untuk membantu Docchi memulai cerita. Perlahan termangap-mangap namun pasti, Docchi mulai menceritakan bagaimana ia sampai ke dunia *bit* ini juga, bagaimana keadaannya sebelumnya, yaitu menjadi salah satu personil "Kresekku Band". Docchi meneruskan ceritanya hingga bagaimana ia bertemu rekan-rekan band-nya, dan bagaimana mereka menjadikan Docchi sosok manusia yang lebih baik dari dirinya sebelum bertemu mereka. Kamu menikmati cerita tersebut, meskipun kamu harus membantu Docchi melanjutkan cerita dengan sekian pertanyaan pancingan.

DOCCHI THE STONE OPENING SONG



# Masa Muda Dan Frame Buffer

Kresekku Band

"...jadi gitu." Docchi mengakhiri cerita perjalanannya sampai tenggorokannya kering. Ya, sama seperti kamu sebelumnya. Docchi juga terlihat memasang senyum lesu saat mengakhiri ceritanya, sepertinya ia menyangkan situasinya saat ini. Tetapi tentu saja, kamu sadar akan situasi kamu. Kamu adalah satu-satunya orang yang dapat Docchi andalkan bila dibutuhkan saat itu. Kamu juga satu-satunya orang-yang kamu tahu, paling tidak-yang telah ditunjuk oleh Gadis Misterius itu untuk dapat menyelesaikan "puzzle" ini. Dan yang terpenting, kamu memiliki kekuatan ~~anime~~ informatika bersamamu, yang akan menjadi penyelamat bagi kamu, Docchi, dan mungkin Pa Una.

Kamu membulatkan tekad. Kamu berdiri di depan Docchi dan menatap wajahnya dengan serius. Kamu menepuk dan memegang kedua bahu Docchi, sembari mencoba meyakinkannya.

"Kita akan keluar dari sini."

"...janji?"

"Janji." Yak, sebuah adegan yang sama sekali tidak terinspirasi dari acara TV atau film manapun.

Ucapan kamu sepertinya membuahkan hasil. Docchi mulai tersenyum dan mengangguk semangat. Di dalam batin kamu merasa seperti satu langkah lagi menjadi karakter utama anime, ngga juga sih ngga tahu penulis lagi-lagi hanya menebak.

Kamu mulai menatap kedepan, eh nggak tau juga depannya kemana, ngga ada kompas, ngga ada penanda apapun, hanya *bit bit*. Tapi kamu yakin, tekad yang kuat akan menunjukkan jalan yang benar untuk kamu. Kamu mulai bersiap-siap untuk melanjutkan perjalanan, yang kali ini kamu sudah memiliki bekal yang jauh lebih banyak. Kamu sudah memahami cara kerja dasar dari dunia *bit* ini. Kamu sudah mampu mengendalikan framebuffer untuk mengatur beberapa *bit*. Selain itu, ilmu pengetahuan juga berdatangan dari luar domain yang membuat kamu semakin paham dengan dunia ini. Bagaimana abstraksi objek-objek *bit* serta metodenya, bagaimana *bit-bit* ini saling berelasi membentuk suatu basis *bit*, bagaimana mengoptimasi segala tindakan dengan berbagai strategi dan bagaimana pula mengukur hasil optimasi tersebut, bagaimana pula perilaku *bit-bit* ini terhadap suatu ketidakpastian, dan yang terpenting, bagaimana keseluruhan pemahaman tersebut saling berkaitan dan dipadukan menjadi suatu proses rekayasa kompleks yang menjadikannya senjata pamungkas kamu untuk menaklukkan dunia ini.

"...skuy lanjut."

"skuy."

## II. Deskripsi Tugas

Tugas ini merupakan lanjutan dari pembuatan sebuah program *mistis*. Sistem operasi yang akan dibuat akan berjalan pada arsitektur x86 32-bit yang akan dijalankan dengan *emulator* QEMU. Jika dalam milestone 1 merupakan setup dan pemanasan, milestone ini akan mengasumsikan milestone 1 sudah berjalan dan dipahami dengan baik; milestone ini bukan sepenuhnya pemanasan.

### *Here be dragons*


For No Duh

Milestone ini akan berfokus kepada interrupt, simple hardware device driver untuk keyboard & disk, dan file system **"FAT32 - IF2230 edition"**

- Interrupt & IDT
- Keyboard driver
- Disk driver
- File System

File-file template dan contoh milestone 2 dapat diakses pada repository berikut [Sister20/kit-OS-milestone-2-2023](#)

Guidebook tugas besar untuk milestone 2 dapat diakses pada link berikut

 IF2230 - Guidebook - Milestone 2

**Catatan penting:** Tugas besar sistem operasi didesain secara sekuensial dan bergantung dengan pengerjaan sebelumnya. Permasalahan pada milestone sebelumnya akan berdampak langsung pada milestone selanjutnya

# III. Spesifikasi Tugas

## 3.0. Outline Pengerjaan

Sedikit berbeda dengan sebelumnya, dokumen ini akan berfokus kepada target yang ada pada milestone 2. Penjelasan yang lebih lengkap akan dipisah pada dokumen guidebook.

Pengerjaan dapat dibagi menjadi dua bagian, bagian yang relatif sederhana dan pembuatan file system **"FAT32 - IF2230 edition"**

### - 3.1. Interrupt

- ☐ IDT Data Structures
- ☐ PIC Remapping
- ☐ Interrupt Handler / Interrupt Service Routine
- ☐ Load IDT
- ☐ Testing Interrupt

### - 3.2. Keyboard Device Driver

- ☐ IRQ1 - Keyboard Controller
- ☐ Keyboard Driver Interfaces
- ☐ Keyboard ISR
- ☐ Testing Keyboard Driver

Pembuatan file system **"FAT32 - IF2230 edition"**

### - 3.3. Filesystem

- ☐ Disk Driver & Image - ATA PIO
- ☐ FAT32 Data Structures
- ☐ File System Initializer
- ☐ File system CRUD Operation



## 3.1. Interrupt

Sebelum membuat *driver device* eksternal, Interrupt perlu dipersiapkan terlebih dahulu sebagai cara komunikasi CPU dan *device* eksternal tersebut. Bagian ini akan berfokus dengan penyiapan Interrupt Handler

### 3.1.1. Interrupt Descriptor Table

IDT akan menyimpan informasi terkait interrupt handler pada sistem operasi. Buatlah deklarasi struktur data IDT pada file **idt.h** dan **idt.c** seperti GDT:

- **IDTGate**
- **InterruptDescriptorTable**
- **IDTR**

Setelah deklarasi struktur data tersebut dibuat, definisikan satu variabel global **InterruptDescriptorTable** dan **IDTR** pada **idt.c**.

Detail dapat dicek pada **Intel x86 Vol 3a - Chapter 6 - Interrupt and Exception Handling**.

### 3.1.2. PIC remapping

Tugas besar ini akan menggunakan [8259 PIC](#) yang ada pada CPU x86, karena **APIC** relatif kompleks (dapat dicek pada **Intel x86 Vol 3a - Ch11 - APIC**). Sebelum melanjutkan untuk menyiapkan ISR, **8259 PIC** tersebut perlu digeser terlebih dahulu.

Buatlah sebuah fungsi **pic\_remap()** yang menggeser interrupt yang dibuat **8259 PIC** ke tempat lain seperti **0x20** dan **0x28**.

### 3.1.3. Interrupt Handler / Interrupt Service Routine (ISR)

Setelah interrupt dari PIC digeser, dapat dilanjutkan untuk menyiapkan generic interrupt handler yang akan memanggil fungsi C yang sesuai. Sebagian besar kode untuk interrupt harus diimplementasikan dengan assembly karena membutuhkan akses langsung ke register.

Fungsi assembly **call\_generic\_handler()** akan memanggil fungsi C **main\_interrupt\_handler()**. Fungsi ini merupakan kelanjutan dari generic interrupt handler, tetapi diimplementasikan pada bahasa C.



Buatlah:

- **interrupt\_handler\_i()** yang merupakan macro asm untuk melakukan interrupt handling sederhana dan memanggil **call\_generic\_handler()**
- **call\_generic\_handler()** yang akan dipanggil setelah **interrupt\_handler\_i()**, dan yang memiliki tugas untuk menyiapkan call stack dan memanggil fungsi C **main\_interrupt\_handler()**
- **main\_interrupt\_handler()**, fungsi C yang akan memanggil ISR yang sesuai untuk interrupt vector
- **isr\_stub\_table**, sebuah array pada assembly yang merupakan function pointer ke **interrupt\_handler\_i**

### 3.1.4. Load IDT

Setelah semua struktur data IDT dan fungsi-fungsi pendukung interrupt telah didefinisikan dengan baik, IDT dapat dihubungkan dengan interrupt\_handler (yang telah dibuat di-assembly) yang sesuai.

Buatlah:

- **set\_idt\_gate()** mengatur nilai IDT dengan IDTGate yang sesuai
- **initialize\_idt()** yang mengatur IDT menggunakan **set\_idt\_gate()** dan melakukan **lidt**

### 3.1.5. Testing Interrupt

Pada kernel, tambahkan kode berikut

**kernel.c**

```
void kernel_setup(void) {
    enter_protected_mode(&_gdt_gdtr);
    pic_remap();
    initialize_idt();
    framebuffer_clear();
    framebuffer_set_cursor(0, 0);
    __asm__("int $0x4");
    while (TRUE);
}
```

Jika berhasil, sistem operasi tidak akan melakukan apapun kecuali blinking cursor pada pojok kiri atas. Jika masih terdapat masalah pada GDT dan IDT, sistem operasi dapat mengalami **general protection fault**, **double fault**, atau **triple fault** ketika memanggil interrupt.

## 3.2. Keyboard Device Driver

Driver akan dibuat merupakan driver keyboard yang sederhana. Driver hanya akan berfokus mengelola satu set scancode keyboard.

Desain yang akan digunakan untuk driver adalah menyimpan input ke keyboard buffer ketika **keyboard\_state.keyboard\_input\_on** bernilai true dan berhenti ketika menerima tombol enter.

### 3.2.1. Enable IRQ1 - Keyboard controller

Sebelum memulai untuk membuat driver keyboard, pastikan interrupt sudah berjalan dengan baik. Hal ini dikarenakan keyboard controller menggunakan interrupt PIC dan port I/O untuk berkomunikasi dengan CPU. Jika tidak menggunakan IRQ0, matikan IRQ dengan mask.

Buatlah:

- **activate\_keyboard\_interrupt()** yang mengirimkan mask untuk IRQ1 ke PIC

### 3.2.2. Keyboard Driver Interface

Keyboard driver yang akan dibuat bersifat stateful. State dari driver akan disimpan pada static variable pada **keyboard.c**.

Buatlah:

- **KeyboardDriverState**, struktur data yang menyimpan state keyboard, berisi state dari pembacaan driver (**keyboard\_input\_on**), buffer driver (**keyboard\_buffer**), dan state lainnya sekiranya diperlukan.
- **keyboard\_state\_activate()**, mengaktifkan pembacaan driver
- **keyboard\_state\_deactivate()**, mematikan pembacaan driver
- **get\_keyboard\_buffer()**, mengambil keyboard buffer yang dimiliki driver
- **is\_keyboard\_blocking()**, mengecek status keyboard apakah sedang membaca

### 3.2.3. Keyboard ISR

Sesuai dengan desain, driver akan berhenti pembacaan dari keyboard ketika **keyboard\_state.keyboard\_input\_on** mati. Pembacaan juga akan berhenti ketika tombol enter ditekan. Perhatikan bahwa mekanisme komunikasi CPU dan keyboard menggunakan

interrupt dan port I/O, hal ini menyebabkan **keyboard\_isr()** akan selalu dipanggil keyboard mengirimkan interrupt dan hal ini bersifat non-blocking.

Berikut adalah spesifikasi untuk **keyboard\_isr()**:

- Ketika **keyboard\_state.keyboard\_input\_on** bernilai true
  - Driver memproses scancode yang diterima ke ASCII character
  - Driver menyimpan hasil ASCII ke **keyboard\_state.keyboard\_buffer**
  - Menuliskan karakter ASCII ke layar dan menggeser kursor
  - Jika ASCII backspace ``b`` dibaca, hapus karakter dari buffer dan layar
  - Berhenti pembacaan input ada ASCII Line Feed / Enter ``n``
- Behavior wrapping & scrolling dibebaskan

Opsional: Buatlah **framebuffer\_get\_cursor()** untuk membantu pembuatan fungsi **keyboard\_isr()**.

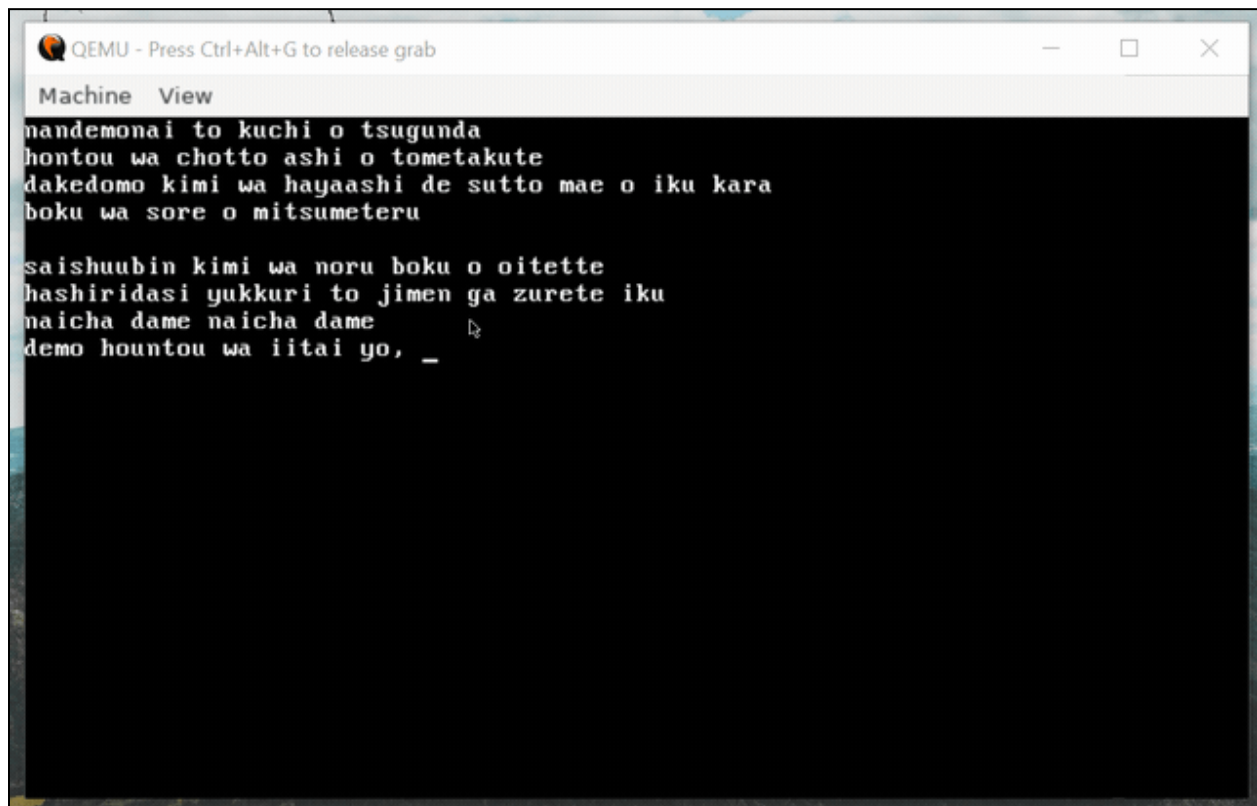
### 3.2.4. Testing Keyboard Driver

Jika sudah diimplementasikan dengan baik, keyboard driver dapat diuji dengan kode seperti berikut pada kernel

**kernel.c**

```
void kernel_setup(void) {
    enter_protected_mode(&_gdt_gdtr);
    pic_remap();
    activate_keyboard_interrupt();
    initialize_idt();
    framebuffer_clear();
    framebuffer_set_cursor(0, 0);
    while (TRUE)
        keyboard_state_activate();
}
```

Jika driver sudah diimplementasikan dengan baik, mestinya kode diatas akan mendengar input keyboard dan menulis ke layar seperti berikut



Ik minor typo

### 3.3. File System - FAT32 - IF2230 edition

Sistem operasi akan mengimplementasikan file system FAT32 dengan beberapa modifikasi, sehingga sesuai dengan namanya "**FAT32 - IF2230 edition**". Namun *main idea* masih sama dengan FAT, sehingga referensi yang ada pada Internet mestinya masih dapat digunakan.

#### 3.3.1. Disk Driver & Image

Sebelum membuat File System, tentunya diperlukan disk driver yang akan berhubungan langsung dengan disk menggunakan **ATA PIO**.

Berbeda dengan keyboard driver, disk driver bersifat stateless dan hanya menyediakan dua interface yang dapat dipakai.

Buatlah:

- **read\_blocks()**, Fungsi yang membaca blocks secara LBA menggunakan ATA PIO
- **write\_blocks()**, Fungsi yang menulis blocks secara LBA menggunakan ATA PIO
- Buat sebuah **disk** berformat **raw binary** dengan qemu-img
- Setup qemu dan workflow sehingga **-drive** terattach dan dapat digunakan

#### 3.3.2. Desain FAT32 - IF2230 edition

Sesuai yang disebutkan sebelumnya, file system yang akan diimplementasikan berdasarkan FAT32. Namun terdapat beberapa modifikasi untuk mempermudah implementasi.

Berikut merupakan desain file system yang akan diimplementasikan:

- Boot sector hanya menyimpan **fs\_signature**, dan sisa disk akan digunakan penuh untuk file system
- Isi **fs\_signature** diperbolehkan untuk dimodifikasi (misalnya isi dengan nama sendiri)
- Sebagian besar parameter file system akan di hardcode dengan macro **#define**, sehingga **BPB** dan **FSInfo** tidak dibutuhkan
- **DirectoryEntry** tidak harus mensupport **FAT Long File Names**
- Boot sector (Sektor 0) - **File System Signature IF2230 - fs\_signature**
- Cluster 0 - Reserved

- Cluster 1 - FAT, satu cluster penuh akan digunakan untuk **FileAllocationTable**
- Cluster 2 - Root, root **DirectoryTable** file system.
- Untuk folder / direktori (termasuk root), pada FAT, **dijamin hanya 1 cluster** (tidak menunjuk ke cluster lain, alias dalam satu folder terbatas hanya 64 file dan folder jika menggunakan setting default).
- Direktori entry yang tidak kosong akan ditandai dengan flag **UATTR\_NOT\_EMPTY** pada **user\_attribute**
- Jika **DirectoryEntry** merupakan folder / direktori, attribute flag **ATTR\_SUBDIRECTORY** aktif
- **DirectoryTable** kosong akan memiliki 1 entry pada awal, yaitu **DirectoryEntry** yang menunjuk ke parent directory. **DirectoryEntry** ini memiliki nama direktori itu (ex. Folder "abcd", memiliki **DirectoryEntry** index 0 bernama "abcd" yang menunjuk ke parent).
- Khusus untuk root, parent direktori adalah dirinya sendiri
- Direktori / folder dikatakan kosong, **jika dan hanya jika**, DirectoryTable hanya berisikan 1 **DirectoryEntry** yang disebutkan diatas (pointer to parent direktori).
- Spesifikasi yang lebih detail akan dijelaskan pada guidebook
- Selain yang disebutkan, behaviour file system akan sama dengan FAT32.

Struktur data untuk **FAT32 - IF2230 edition** masih akan sama persis dengan **FAT32**. Namun beberapa atribut tidak dipakai pada tugas besar ini (opsional jika ingin menggunakan). Cek referensi untuk mendapatkan dokumen [Microsoft Hardware White Paper - FAT32 File System Specification](#).

Buatlah struktur data berikut:

- **FAT32FileAllocationTable**
- **FAT32DirectoryEntry**
- **FAT32DirectoryTable**

Jika merasa ada yang kurang jelas dan memiliki pertanyaan terkait desain file system, dapat menggunakan QnA dan asistensi.

### 3.3.3. Initialize File System

Dengan memperhatikan desain yang telah disebutkan dan struktur data yang telah dibuat sebelumnya, development file system dapat dilanjutkan dengan membuat fungsi-fungsi untuk inisiasi file system

Buatlah:

- **FAT32DriverState**, struktur data yang menyimpan fat32 driver state
- **initialize\_filesystem\_fat32()**, menginisiasi file system sistem operasi
- **is\_empty\_storage()**, mengecek apakah disk tidak memiliki signature
- **create\_fat32()**, membuat file system FAT32 - IF2230 edition pada disk

Opsional:

- **init\_directory\_table()**, menginisiasi **DirectoryTable** dengan sesuai dengan desain
- Membuat wrapper **read\_clusters()** dan **write\_clusters()**
- **cluster\_to\_lba()**, mengkonversikan cluster number ke LBA

### 3.3.4. CRUD Operation

Setelah memiliki struktur data dan file system initializer, pengerjaan dapat dilanjut ke pembuatan **CRUD operation** pada file system.

Buatlah:

- **FAT32DriverRequest**, digunakan untuk semua CRUD operation
- **read()**, membaca sebuah file pada file system
- **read\_directory()**, membaca sebuah folder pada file system
- **write()**, menuliskan sebuah file atau folder ke file system
- **delete()**, menghapus sebuah file atau folder kosong pada file system

Untuk mempermudah operasi file system, ubah **KERNEL\_STACK\_SIZE** pada **kernel\_loader.s** ke **2097152 bytes (2 MiB)**, sehingga stack kernel cukup longgar untuk melakukan operasi file system. Ingat, satu cluster dengan ukuran default berukuran **2048 bytes**, sehingga operasi file system akan cepat menggunakan kernel stack space.

Usahakan buat seluruh kode file system self-contained / decoupled (selain disk driver & standard library) pada **fat32.h** & **fat32.c**. Mestinya kedua file hanya membutuhkan **read\_blocks()** & **write\_blocks()**. Nantinya driver fat32 akan digunakan untuk membuat **program pada host OS** untuk memasukkan sembarang file ke dalam file system pada milestone berikutnya.



Gunakan kode berikut untuk mengetes file system

#### kernel.c

```
void kernel_setup(void) {
    enter_protected_mode(&_gdt_gdtr);
    pic_remap();
    initialize_idt();
    activate_keyboard_interrupt();
    framebuffer_clear();
    framebuffer_set_cursor(0, 0);
    initialize_filesystem_fat32();
    keyboard_state_activate();

    struct ClusterBuffer cbuf[5];
    for (uint32_t i = 0; i < 5; i++)
        for (uint32_t j = 0; j < CLUSTER_SIZE; j++)
            cbuf[i].buf[j] = i + 'a';

    struct FAT32DriverRequest request = {
        .buf = cbuf,
        .name = "ikanaide",
        .ext = "uwu",
        .parent_cluster_number = ROOT_CLUSTER_NUMBER,
        .buffer_size = 0,
    };

    write(request); // Create folder "ikanaide"
    memcpy(request.name, "kano1\\0\\0\\0", 8);
    write(request); // Create folder "kano1"
    memcpy(request.name, "ikanaide", 8);
    memcpy(request.ext, "\\0\\0\\0", 3);
    delete(request); // Delete first folder, thus creating hole in FS

    memcpy(request.name, "daijoubu", 8);
    request.buffer_size = 5*CLUSTER_SIZE;
    write(request); // Create fragmented file "daijoubu"

    struct ClusterBuffer readcbuf;
```

```
read_clusters(&readcbuf, ROOT_CLUSTER_NUMBER+1, 1);
// If read properly, readcbuf should filled with 'a'

request.buffer_size = CLUSTER_SIZE;
read(request); // Failed read due not enough buffer size
request.buffer_size = 5*CLUSTER_SIZE;
read(request); // Success read on file "daijoubu"

while (TRUE);
}
```

Kondisi akhir **FAT** setelah kode diatas mestinya adalah seperti berikut

FileAllocationTable				
Offset	0	1	2	3
0	CLUSTER_0_VALUE	CLUSTER_1_VALUE	Root directory	"daijoubu.uwu"-0
4	Folder "kano1"	"daijoubu.uwu"-1	"daijoubu.uwu"-2	"daijoubu.uwu"-3
8	"daijoubu.uwu"-4 EOF			

Sistem operasi yang telah dibuat pada state ini semestinya sudah memiliki *"freedom"* pada tingkat kernel. Descriptor-descriptor yang wajib ada dan proses inisiasi pada x86 protected mode sudah diimplementasikan, sehingga diperbolehkan untuk berkreasi dengan menambahkan fitur lain.

*Next up in the list: Interactive shell, virtual memory, and the dreaded user mode*

### 3.4. Bonus

**Bonus merupakan bagian yang tidak wajib untuk dikerjakan.**

- **Unlimited DirectoryEntry untuk file system (5)**

Seperti yang disebutkan pada desain **FAT32 - IF2230 edition**, dijamin folder hanya memiliki satu cluster untuk memudahkan implementasi. Namun diperbolehkan jika ingin mendukung cluster chain untuk directory.

Catatan : Jika DIY file system juga mendukung unlimited directory entry, bonus ini juga akan dihitung.

- **Dukungan CMOS time untuk file system (5)**

Spesifikasi wajib tidak mewajibkan untuk menggunakan date/time yang ada pada metadata file system **FAT32**. Bonus ini membutuhkan komunikasi dengan CMOS menggunakan port I/O atau dalam kata lain perlu menulis device driver untuk CMOS.

- **Desain file system sendiri (20)**

Untuk menunjang kreativitas yang sangat brutal dan *mistis*, diperbolehkan untuk membuat desain file system yang berbeda dari spesifikasi wajib.

Requirement desain file system:

- **Boot sector wajib IF2230 fs\_signature**, selain itu dibebaskan sepenuhnya (termasuk isinya juga bebas)
- **Mendukung metadata** minimal nama, filesize, is\_present flag, dan parent directory
- **Mendukung direktori / folder**, termasuk nested folder
- Driver menyediakan **file system initializer**
- Driver menyediakan **interface CRUD** yang dapat digunakan

Penilaian akan dilakukan *case-by-case basis* dan nilai bonus yang berubah sesuai dengan originalitas & kompleksitas desain.

- **Speedrun any% god mode (10)**

Selesaikan tugas besar hingga tag release telah dibuat sebelum guidebook dirilis. **Tidak diperbolehkan untuk merevisi release tag setelah guidebook dirilis.**

- **No guidebook 100% run (hanya yang diatas yang tahu)**

Buatlah semua requirement yang diberikan pada spesifikasi tanpa melihat guidebook. Diperbolehkan assign **fs\_signature** sendiri jika mengerjakan bonus ini. Bonus ini memiliki nilai *indeterminate form*. smiley face

## IV. Penilaian

- **Interrupt (20)**
  - a. Struktur data IDT **(5)**
  - b. PIC Remapping & ISR setup **(5)**
  - c. Load IDT & Interrupt berhasil memanggil **main\_interrupt\_handler()** **(10)**
- **Keyboard Driver (20)**
  - a. IRQ1 sudah aktif **(5)**
  - b. Keyboard Driver + ISR berjalan sesuai desain **(15)**
- **File System FAT32 - IF2230 edition (60)**

**Note : Pentingkan eksekusi berhasil sebelum mengikuti desain spesifikasi**

  - a. Disk driver & Image terimplementasi **(5)**
  - b. Struktur data FAT32 terdefinisi dengan baik **(5)**
  - c. Fungsi File system Initializer berjalan sesuai spesifikasi **(10)**
  - d. FS CRUD operation berjalan sesuai ekspektasi **(40)**

## V. Pengumpulan dan Deliverables

1. Tugas ini akan melanjutkan milestone sebelumnya sehingga pengerjaan akan tetap menggunakan repository **GitHub Classroom yang sama dengan milestone 1**.
2. Kit utama untuk milestone tugas besar IF2230 tersedia pada repository GitHub berikut [link repository](#).
3. Kreativitas dalam pengerjaan sangat dianjurkan untuk memperdalam pemahaman. Penilaian sepenuhnya didasarkan dari kriteria penilaian.
4. Lakukanlah commit yang wajar dan sesuai best practice (tidak semua kode satu commit).
5. Hindari menaruh binary file hasil kompilasi pada repository. Gunakan .gitignore untuk mengabaikan file-file tersebut.
6. Pengumpulan dilakukan dengan membuat **release** dengan tag **v2.0** pada repository yang telah kelompok Anda buat sebelum deadline. Jika terdapat revisi, tambahkan angka minor pada versi tag (**v2.1, v2.2, ..., v2.x**) Pastikan tag sesuai format. **Repository team yang tidak memiliki tag ini akan dianggap tidak mengumpulkan Milestone 2.**
7. Kami akan **menindaklanjuti segala bentuk kecurangan** yang terstruktur, masif, dan sistematis.
8. Hindari menggunakan fitur code auto-complete seperti Copilot, Codex, dll pada pengerjaan tugas besar ini sepenuhnya. Jika kode hasil tools tersebut diketahui tim asisten mengambil langsung dari internet, **praktikan yang menggunakan tools tersebut akan dianggap melakukan kecurangan** (Tanggung jawab kode akan berada pada praktikan, bukan tools auto-completion).
9. Jika ada pertanyaan atau masalah pengerjaan harap segera menggunakan sheets **QnA mata kuliah IF2230 Sistem Operasi** pada [link berikut](#).
10. Karena ada kendala dan konsiderasi lain, tutorial yang awalnya direncanakan sebelumnya akan ditiadakan untuk keseluruhan tugas besar ini. Gunakan asistensi, QnA, dan PR feedback pada GitHub sebagai pengganti.
11. Akan terdapat asistensi opsional bagi kelompok yang membutuhkan bantuan lebih lanjut. Setiap kelompok yang membutuhkan asistensi dapat mengisi form Asistensi IF2230 Sistem Operasi pada [form berikut](#). **Asisten akan mengontak maksimal paling lambat H+1 setelah request.** Jika tidak ada yang mengontak, **silahkan mengisi form kembali.**

## VI. Tips Pengerjaan

- Dokumen ini digunakan untuk spesifikasi dan outline yang harus dikerjakan. Untuk penjelasan & guide, dapat mengecek dokumen guidebook dan kit. Keduanya akan diberikan akses **setelah beberapa hari perilisan tugas besar.**

Dokumen guidebook - [IF2230 - Guidebook - Milestone 2](#)

Kit - [Sister20/kit-OS-milestone-2-2023](#)

- Gunakan script makefile & susunan folder untuk merapikan repository. Semakin rapi repository, semakin mudah untuk mengerjakan fitur tambahan
- Mulai dari milestone ini, debugger akan menjadi companion untuk melakukan debugging. Jika mengalami bug, gunakan debugger untuk mengecek nilai variabel atau memory.
- Jika sudah mencoba tetapi mengalami stuck, gunakan asistensi. Anggap tugas besar ini sebagai sarana untuk pembelajaran suatu hal baru, dengan asisten menyediakan layanan bantuan ketika stuck.
- **Pro tip:** Tugas besar ini ditujukan untuk pengerjaan secara kelompok dan jangka panjang.

**Penilaian akhir akan menggunakan weighting**, sehingga nilai individu akan berbeda-beda dalam satu kelompok. Weighting akan berdasarkan peer dan penilaian dari asisten case-by-case basis.

**Jika tidak mengerjakan apapun, expect nilai individu kurang dari nilai kelompok, hingga untuk kasus ekstrim dapat di nol-kan untuk nilai individu**


## VII. Referensi

1. **Kitab Intel x86 dan x64 - Intel® 64 and IA-32 Architectures Developer's Manual Vol 3a**

<https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3a-part-1-manual.html.html>

2. **Microsoft Hardware White Paper - FAT32 File System Specification**

<https://www.win.tue.nl/~aeb/linux/fs/fat/fatgen103.pdf>

3. Guidebook IF2230 -  IF2230 - Guidebook - Milestone 2

4. <https://littleosbook.github.io>

5.  IF2230 - Debugger & WSL





~normal day in lab sister~

Sementara itu, di dunia lain, Pa Una yang dari kemarin-kemarin tidak kelihatan ternyata sedang melakukan hal lain. Pa Una memanfaatkan genjutsunya untuk menembus dimensi dunia *bit* dan sampai di suatu tempat rahasia. Sesuai ekspektasi Pa Una, dunia *bit* tersebut bekerja dengan prinsip yang sama dengan genjutsu miliknya. Pa Una melakukan ini bukan karena alasan, ia ingin bertemu langsung dengan Gadis Misterius itu untuk membahas sesuatu.. "Kita harus bicara, Omega."

~Dzaky~

Spoiler:



~Amar~

Twibbon di atas berasal dari hasil tubes semester 6

~Rozan~

Mau jago baca spek gajelas? Ayo latihan dengan light novel!

~Erik~

~Watashi kininarimasuuuu~

~Gare~

Kalau lelah nonton Zeta dulu bang

~Andreas~

"Segala sesuatu hal akan menjadi mudah jika tidak dikerjakan"

~Stanley~

[Momen ketika milestone selanjutnya lebih susah](#)

~Fawwaz~

Just surtr it

~Alif~

[FD Metal - Another fc when](#)

- Brush