Tugas Besar
IF2230 - Sistem Operasi
Milestone 02 of ??
"Your Job is System Engineer"
Pembuatan Sistem Operasi Sederhana Folder dan Shell

Dipersiapkan oleh :
Asisten Lab Sistem Terdistribusi

Didukung Oleh :

Waktu Mulai :
Kamis, 25 Februari 2021, 20.21 WIB

Waktu Akhir :
Rabu, 17 Maret 2021, 20.21 WIB

I. Latar Belakang

Setelah 2 minggu Anda menjadi seorang player, perkembangan Anda sangatlah pesat. Sekarang Anda sudah menduduki level 45. Akhirnya, sistem menawarkan Anda sebuah “Job Quest”. Ini merupakan kesempatan emas Anda untuk menduduki tingkatan yang lebih tinggi dan memperoleh suatu skill spesialis sendiri seperti hunter kelas S lainnya.

Anda pun dengan berani mengambil quest tersebut. Awal mulanya, dungeon quest tersebut hanyalah dungeon berlevel mudah seperti dungeon yang telah Anda kalahkan sebelumnya. Namun… pada dungeon tersebut, Anda tidak bisa kabur. Anda pun sadar bahwa Anda tidak bisa menggunakan potion-potion yang Anda telah kumpulkan.

Perlahan-lahan, dungeon menjadi semakin sulit dan diri Anda mulai kelelahan karena musuh terus berdatangan. Akhirnya, Anda pun tiba di sebuah ruangan... ruangan dengan sebuah throne. Kali ini, musuh yang Anda lawan levelnya jauh di atas Anda. Anda berusaha mencari segala cara, namun semuanya gagal. Hingga akhirnya... sang lawan membuat celah yang memampukan Anda untuk mengalahkannya. Anda telah memperoleh *item* yang mungkin akan berguna pada dungeon berikutnya. Namun ingat, musuh-musuh masih berdatangan untuk menyerang Anda.

Anda yang lelah setelah melawan musuh yang sangat kuat tersebut, hanya bisa pasrah. Di saat yang bersamaan sesuatu yang biasa membawakan kesedihan untuk Anda, malah membuat Anda selamat. Ya. Penalti quest menyelamatkan Anda.

Dengan bantuan penalti quest, Anda pun berhasil menghabiskan waktu di dungeon tersebut dengan selamat. Akhirnya... muncul sebuah pesan dari sistem.

“ [Your Job is ‘System Engineer’] ”

“What is this?”

“This is... my job?!”

Anda tidak tahu apa-apa mengenai job System Engineer tersebut. Kemudian, saat kamu mencoba menolaknya, muncullah suatu window lagi.

“**[System Engineer is a ‘Restless class'. Do you still wish to refuse?]**”

Anda pun berpikir bahwa kemampuan ini bisa digunakan untuk menciptakan sistem yang membantu Anda menyelesaikan dungeon ini. Anda pun menerima role tersebut dengan senang hati. Dan mulailah petualangan nan indah tersebut...

II. Deskripsi Tugas

Pada praktikum ini, kalian akan melanjutkan sistem operasi yang sudah kalian buat dengan menambahkan *filesystem* yang akan mendukung adanya direktori dan sebuah *shell* sederhana. Hal-hal yang akan anda lakukan pada sistem operasi ini adalah :

- Membuat *filesystem*
- Membuat *syscall* **readSector**, **writeSector**, **readFile** dan **writeFile**
- Membuat program kecil **loadFile**
- Membuat sebuah *shell* sederhana
- Membuat "program" kecil pada shell : **cd**, **ls**, **cat**, **ln**

III. Langkah Pengerjaan

3.1. Pembuatan *filesystem*

Filesystem dari sistem operasi anda terdiri atas 3 sektor khusus yang telah dialokasikan sejak awal, yaitu sektor **map** pada sektor 0x100, sektor **files** pada sektor 0x101-0x102, dan sektor **sectors** pada sektor 0x103.

Buatlah map.img, sectors.img dan files.img dengan perintah berikut:

```
dd if=/dev/zero of=map.img bs=512 count=1
dd if=/dev/zero of=files.img bs=512 count=2
dd if=/dev/zero of=sectors.img bs=512 count=1
```

Sektor **map** terdiri atas 512 byte dimana masing-masing byte menandakan apakah sektor terisi atau belum terisi pada filesystem. Mengingat bahwa anda perlu memuat kernel ke system.img, maka **map.img** perlu anda ubah kontennya. Dengan perintah **`hexedit map.img`** , ubah isi dari map.img agar byte ke-0 hingga ke-10 bernilai 0xFF yang berarti bahwa sektor tersebut terisi (oleh kernel). Range ini berasal dari **bootloader.asm**, di mana label KSIZE menyatakan ukuran kernel (dihitung dari sektor ke-1). Apabila anda mengubah nilai ini, maka sesuaikan dengan jangkauan byte yang perlu untuk anda isi dengan nilai 0xFF.

Salin map.img, files.img, dan sectors.img ke system.img dengan perintah berikut:

```
dd if=map.img of=system.img bs=512 count=1 seek=256 conv=notrunc
dd if=files.img of=system.img bs=512 count=2 seek=257 conv=notrunc
dd if=sectors.img of=system.img bs=512 count=1 seek=259 conv=notrunc
```

Sektor **files** terdiri atas 2 sektor berukuran 512 bytes yang akan menampung informasi dari file dan direktori yang ada pada filesystem kalian. Satu entry pada filesystem berukuran 16 bytes, yang terbagi atas : 1 byte penanda parent directory, 1 byte penanda index sectors, dan 14 bytes untuk nama file. *Filesystem* kali ini akan memberi dukungan untuk folder. Caranya adalah dengan menambahkan *flag* pada entri *file* untuk menandakan apakah sebuah entri adalah folder atau *file*. *Flag* yang digunakan untuk menandakan folder juga berfungsi sebagai penanda indeks tabel sektor untuk *file*.

Jika bingung, bisa dilihat ilustrasi berikut ini:

Struktur dari **2** sektor **files:**

idx																
00	**P**	**S**	**Nama direktori/file 0** (14 karakter)													
01	**P**	**S**	**Nama direktori/file 1** (14 karakter)													
	...															
3F	**P**	**S**	**Nama direktori/file 63** (14 karakter)													

P : Indeks parent dari direktori tersebut. 0xFF jika parentnya root
S : Indeks entri pada tabel sektor (jika folder, akan berisi 0xFF, jika file bisa berisi 0x00 - 0x1F)

Struktur dari sektor **sectors**:

idx																
00	**Sektor *file* 0** (16 sektor)															
01	**Sektor *file* 1** (16 sektor)															
	...															
1F	**Sektor *file* n** (16 sektor)															

Jika dilihat, kita akan mempunyai 4 sektor total (termasuk **map**) untuk *filesystem*. Oleh karena itu, keempat sektor ini akan diletakkan pada posisi-posisi berikut:

- **map** di 0x100
- **files** di 0x101 dan 0x102
- **sectors** di 0x103

Buatlah berkas untuk setiap komponen di atas (***map.img, files.img, sectors.img***) dan letakkan pada posisi yang telah disebutkan.

3.2. Pembuatan *syscall readSector, writeSector,* readFile dan writeFile

Implementasikan **void readSector(char *buffer, int sector)** dengan kode berikut:

```
interrupt(0x13, 0x201, buffer, div(sector, 36) * 0x100 + mod(sector, 18)
+ 1, mod(div(sector, 18), 2) * 0x100);
```

Keterangan:
Pada kode di atas, terdapat fungsi mod dan fungsi div. Fungsi ini harus kalian implementasikan sendiri.
Interrupt 0x13 dapat digunakan untuk berbagai macam hal, namun pada tugas ini hanya akan kita gunakan untuk membaca dan menulis sector. Interrupt tersebut memiliki beberapa argumen:

1. Argumen kedua terbagi menjadi dua bagian, pada kasus di atas 0x02 yang merupakan fungsi yang akan digunakan (pada kasus ini fungsi untuk membaca sector) dan 0x01 yang merupakan`jumlah sektor yang akan dibaca.
2. Argumen ketiga adalah lokasi dimana isi sektor akan disalin.
3. Argumen keempat adalah *cylinder* dan *sector. Sector* yang dimaksud adalah nomor *sector* pada satu *track*. Karena pada floppy, 1 *track* memiliki 18 *sector*, maka pada kode di atas, *sector* diisi dengan mod(sector, 18) + 1. *Cylinder* yang dimaksud adalah nomor *cylinder* pada satu *track.* Karena 1 *cylinder* memiliki 2 *head* yang masing-masing ada 18 *sector* dengan total 36 *sector* per *cylinder*, pada kode di atas *cylinder* diisi dengan div(sector, 36) * 0x100, yakni merujuk ke lokasi cylinder tempat sector yang dicari berada. Perkalian terhadap 0x100 (sama saja dengan logical bitshift left 8 kali) menandakan bahwa nilai ini diisi pada bagian MSB dari register CX (yakni CH)
4. Argumen kelima adalah *head* dan *drive*. Pada kode di atas, *head* nya adalah mod(div(sector, 18), 2) karena *head* hanya dapat bernilai 0 atau 1 dengan masing-masing memiliki 18 *sector*. dan drive nya adalah 0 (drive A:).

Lalu implementasikan **void writeSector(char *buffer, int sector)** yang mirip seperti **readSector(char *buffer, int sector)**, tetapi fungsi yang digunakan adalah 0x03 (Argumen kedua interrupt menjadi 0x301).

Buatlah *syscall* **readFile** dan **writeFile** untuk mendukung *filesystem* yang anda buat.
Input filename menjadi path relatif ke sebuah file. *Path* akan relatif dari sebuah *parentIndex*.

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

Untuk implementasi **readFile** dan **writeFile** perhatikan diagram alir berikut sebagai referensi.

Diagram Alir **writeFile** sebagai referensi

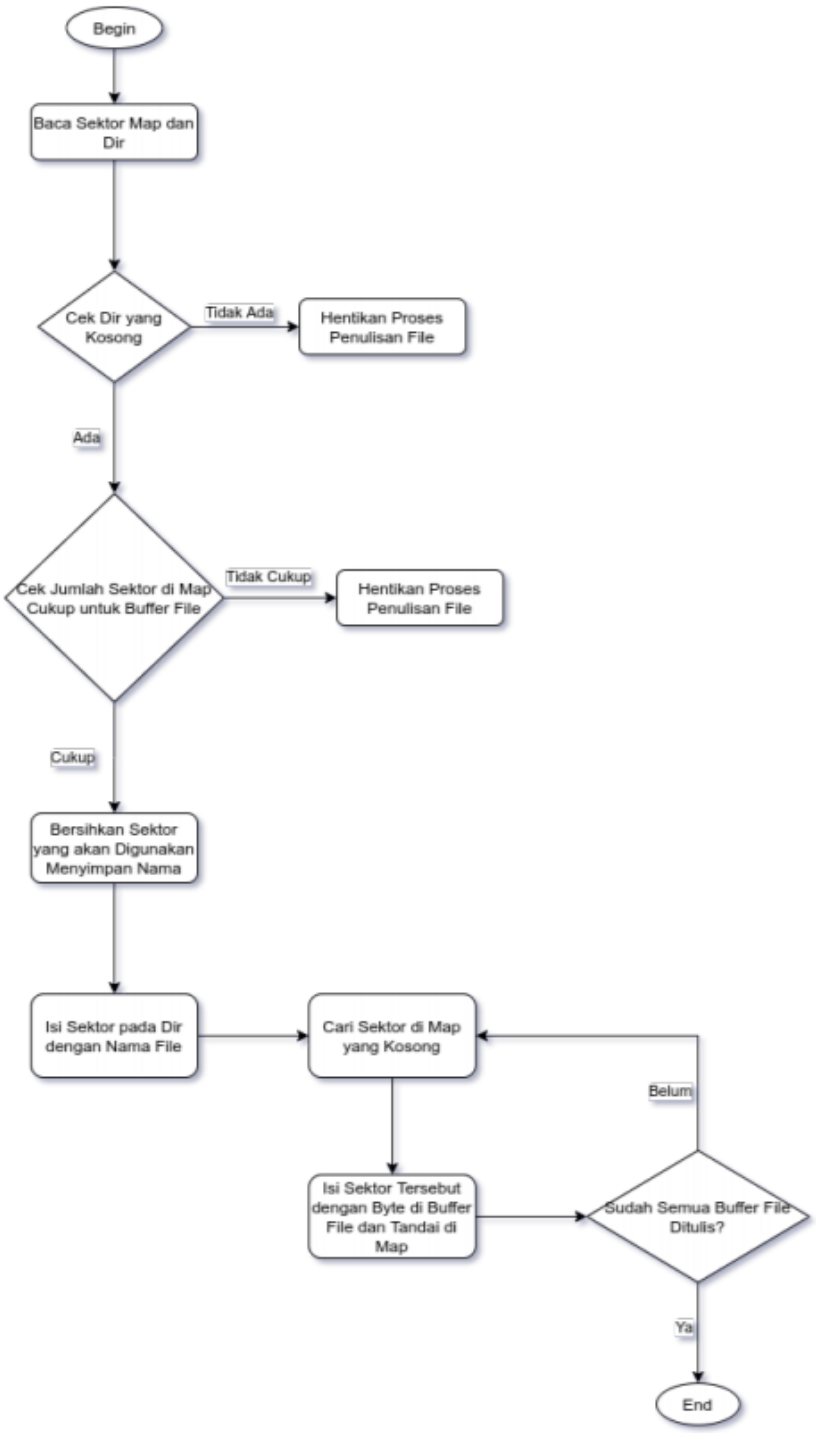
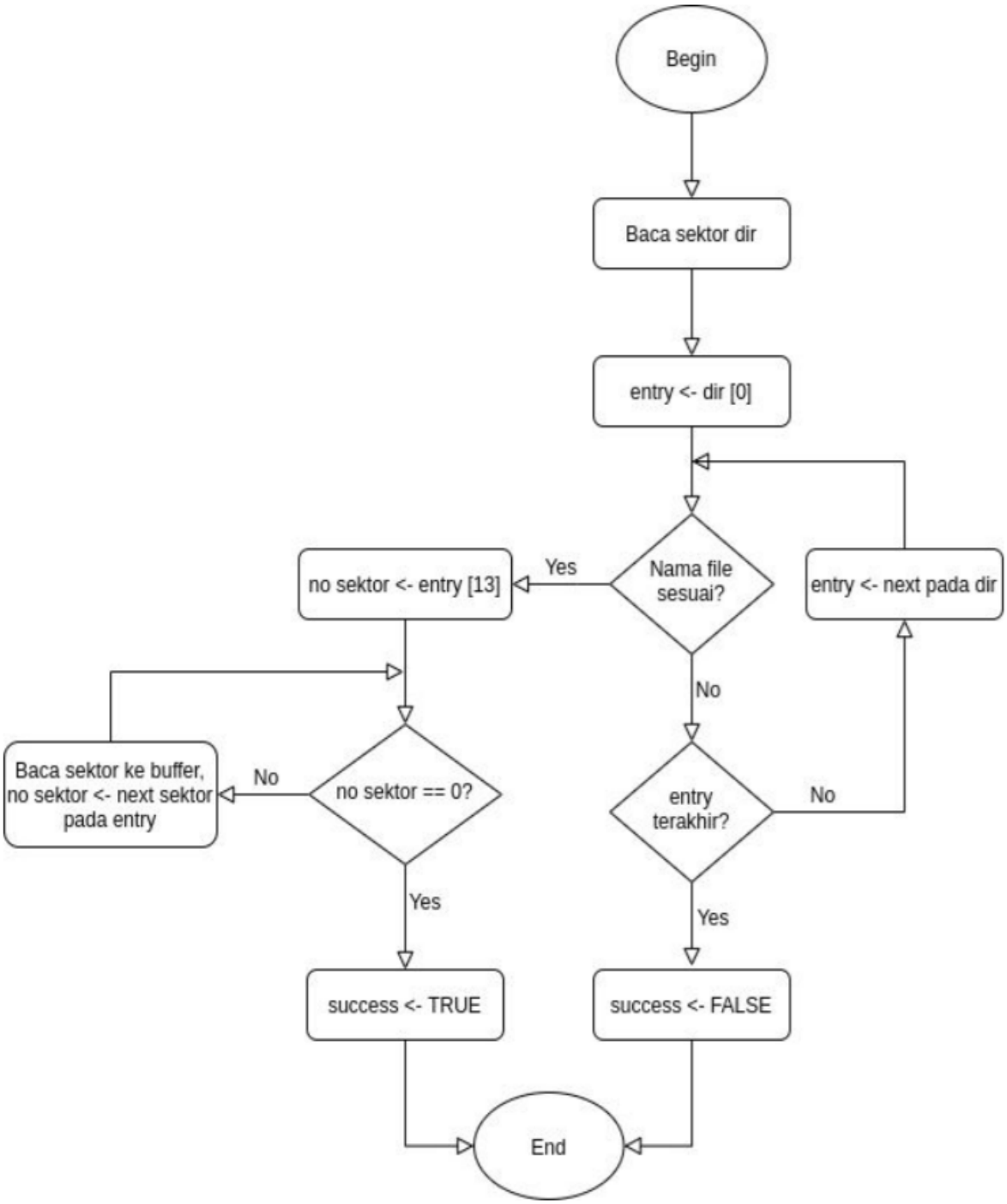


Diagram Alir **readFile** sebagai referensi



Agar fungsi *interrupt* dapat mendukung variabel >=4 maka AX dapat digunakan untuk 2 variabel seperti berikut

```
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;
      default:
         printString("Invalid interrupt");
   }
}
```

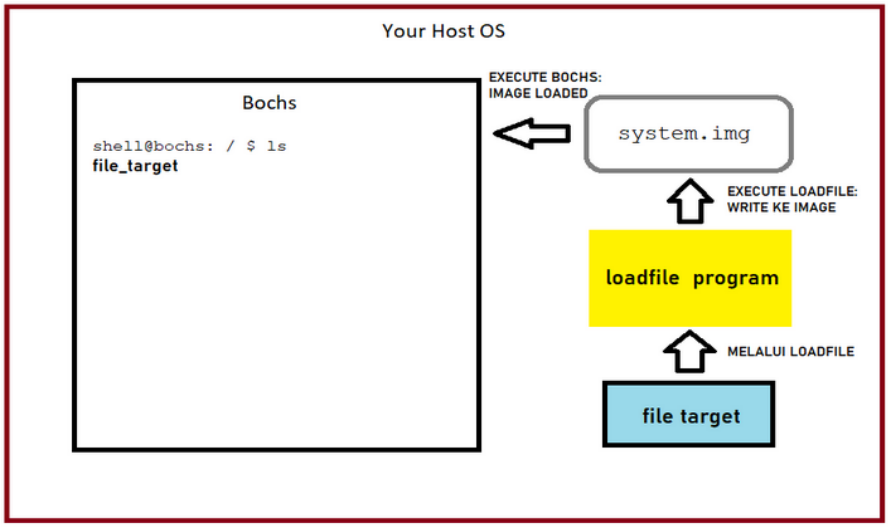
Selain itu tambahkan sistem *error code* pada variabel result di **readFile** dan sectors di **writeFile**. Kode yang digunakan adalah sebagai berikut:

Kode	Arti (readFile)	Arti (writeFile)
-1	File tidak ditemukan (readFile)	File sudah ada
-2		Tidak cukup entri di files
-3		Tidak cukup sektor kosong
-4		Folder tidak valid

3.3. Pembuatan program *loadFile*

Buatlah program terpisah (pada *host*) yang dapat memuat berkas dari *host* ke dalam *filesystem* sistem operasi yang anda buat. Diperbolehkan menggunakan bahasa pemrograman apapun.

Berikut merupakan penjelasan visualisasi cara kerja program loadfile yang harus Anda buat:



1. Buatlah implementasi sebuah program loadfile yang menerima path sebuah file eksternal (**file target**) yang berada di Host OS Anda.
2. File target ini akan diwrite isinya oleh program loadfile Anda ke system.img yang telah Anda buat. Perlu diperhatikan skema writefile nya mengikuti skema filesystem yang telah dijelaskan sebelumnya. Eksekusi program loadfile Anda untuk melakukan write sebelum menjalankan sistem operasi kalian (inilah yang dimaksud dengan “terpisah, pada host”)
3. Setelah berhasil melakukan write, berarti file Anda sudah masuk ke filesystem di image Anda
4. Contohnya di gambar (asumsikan sudah membuat shell), saat Anda start bochs dan melakukan **ls** (refer ke bagian 3.4) maka file test Anda akan terlihat di output ls (dan juga bisa dilihat isinya menggunakan **cat**)

3.4. Membuat *shell* sederhana

Agar tiap operasi di sistem operasi Anda menjadi lebih mudah, dibutuhkan sebuah *shell* sederhana. *Shell* ini akan memiliki fitur berikut:

- Menampilkan *current directory* di *prompt*
- Menerima perintah sederhana : **cd**, **ls**, **cat**, **ln**
- Dukungan **cd** (*change directory*) secara relatif dengan “. .”. Misal:

  ```
  cd ../../a/b/c/../d
  cd ./c/./d
  ```

- **ls** dapat melakukan directory listing (termasuk file) pada posisi shell sekarang
- **cat** untuk melakukan print dari isi file ke layar (gunakan printString dan readFile)
- **ln** untuk melakukan symbolic link dalam mode hard link

Apabila kurang jelas dalam mengimplementasikan command, rujuk ke https://man7.org

3.5. Bonus

Bonus untuk milestone ini (diurutkan berdasarkan kesulitan dan skor dari yang terkecil)

- Symbolic link (**ln**) dapat dilakukan dalam mode soft link dengan menggunakan *flag* "-s"
- Autocomplete file yang tersedia untuk **cd**, **ls**, dan **cat**
- Autocomplete folder saat melakukan **cd**
- Shell memiliki history (minimal 3 command)

Apabila ingin melihat konsep dari file system, bisa merujuk ke **Chapter 10 Operating System Concepts, 8th edition oleh Silberschatz, Galvin, Gagne**.

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. 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*

4. Kelompok tetap sama dan dapat dilihat pada link berikut s.id/kelompok-os19.

5. **Mulai** Kamis, 25 Februari 2021, 20.21 WIB waktu server.
   Deadline Rabu, 17 Maret 2021, 20.21 WIB waktu server.

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

6. Pengumpulan dilakukan dengan membuat ***release*** dengan tag **"v.2.0.0"** (tanpa tanda kutip) pada repository yang telah kelompok Anda buat sebelum deadline. **Repository team yang tidak memiliki tag ini akan dianggap tidak mengumpulkan Milestone 2. Kesalahan dalam penulisan tag akan berakibat pada pengurangan nilai.**

7. Teknis pengumpulan adalah via kode yang terdapat di *repository* saat *deadline*. Kami akan menindaklanjuti **segala bentuk kecurangan**

8. 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.

9. Dilarang melakukan kecurangan lain yang merugikan peserta mata kuliah IF2230.
10. 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 (HxD pada Windows).
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