

Modul Praktikum

Sistem Operasi



Disusun oleh :
Ir. Bana Handaga, M.T., PhD.
Aris Rakhmadi, S.T., M.Eng.
Jan Wantoro, S.T., M.Eng.
Nurul Kholisatul 'ulya, S.Kom, M.Eng.

Program Studi Informatika
Fakultas Komunikasi dan Informatika
Universitas Muhammadiyah Surakarta

Daftar isi

Halaman Judul -----	i
Daftar Isi -----	ii
Modul 1 : Pengenalan Sistem Pengembangan OS Dengan PC Simulator 'Bochs' -----	1

Tujuan -----	1
Pendahuluan -----	1
Peralatan -----	1
Langkah kerja -----	2
Tugas -----	17
 Modul 2 : Mengenal Proses Membuat 'Disk Boot' -----	18
Tujuan -----	18
Pendahuluan -----	18
Peralatan -----	19
Langkah kerja -----	19
Tugas -----	26
 Modul 3 : Mengenal Cara 'Debugging' Program Bootstrap-loader -----	27
Tujuan -----	27
Pendahuluan -----	27
Peralatan -----	28
Langkah kerja -----	28
Tugas -----	33
 Modul 4 : Pengolahan File dan Directory Menggunakan MS-Dos / Command Prompt di Windows -----	34
Tujuan -----	34
Pendahuluan -----	34
Peralatan -----	38
Langkah kerja -----	38
Tugas -----	43
 Modul 5 : Pengenalan Sistem Operasi Linux -----	44
Tujuan -----	44
Pendahuluan -----	44
Petunjuk Praktikum -----	49
Tugas -----	51
 Modul 6 : Managemen User dan Permission Direktori / File Pada Linux -----	52
Pendahuluan -----	52
Petunjuk Praktikum -----	53
Tugas -----	63

Modul 7 : Managemen User dan Permission Direktori / File Pada Linux-----	64
Tujuan-----	64
Pendahuluan-----	64
Petunjuk Praktikum-----	68
Modul 8 : System Call-----	72
Tujuan-----	72
Pendahuluan-----	72
Peralatan-----	73
Langkah Kerja-----	73
Modul 9 : File System Call-----	80
Tujuan-----	80
Pendahuluan-----	80
Peralatan-----	80
Langkah Kerja-----	81
Modul 10 : Simulasi Command (Perintah)-----	85
Tujuan-----	85
Pendahuluan-----	85
Peralatan-----	85
Langkah Kerja-----	85

Modul 1: Pengenalan Sistem Pengembangan OS dengan PC Simulator ‘Bochs’

Tujuan

Mengenal lingkungan dan tools untuk pengembangan dasar-dasar sistem operasi

Pendahuluan

Modul ini merupakan pengantar yang keberhasilanya akan menentukan kelancaran proses praktikum pada modul-modul berikutnya. Pada praktikum ini akan dikenalkan beberapa hal penting yang terkait dengan PC dan dasar sistem operasi, serta beberapa tools alternatif pengembangan sistem operasi.

Beberapa hal penting di samping pengetahuan tentang sistem operasi yang diperlukan dalam pengembangan sistem operasi antara lain adalah menguasai bahasa pemrogramman assembly dan C dengan baik, kemudian memahami cara kerja PC dan dapat menggunakan PC dengan lancar. Selain itu juga diperlukan pengetahuan sistem angka digital yang memadai, seperti representasi angka biner dan hexa, konversi angka desimal ke biner atau hexa. Hal-hal di atas merupakan dasar yang perlu dipahami dengan baik sebelum mengembangkan sebuah sistem operasi.

Pada modul ini akan belajar cara menggunakan beberapa program bantu untuk pengembangan sistem operasi, dan belajar lebih dekat tentang proses ‘boot’ pada PC dan beberapa hal yang terkait dengan hal tersebut. Praktikum akan dimulai dengan cara membuka direktori kerja, melihat file-file dalam direktori kerja, sekilas tentang ‘Makefile’, memahami ‘boot disk’, melihat data dalam bootsector

Peralatan

1. PC dengan sistem operasi Windows Xp
2. Program Simulator Boschs
3. Kompiler bahasa assembly ’nasm’
4. Kompiler bahasa C.
5. program bantu seperti ‘make’, ‘debug’, ‘dd’, dan ‘tdump’
6. notepad

Semua proses praktikum dilakukan dalam mode kerja ‘Command Prompt’, hal ini dimaksudkan agar praktikan dapat memahami prosesnya secara lebih baik.

Langkah Kerja

Menuju ke direktori kerja.

- a. Jalankan program command prompt atau cmd.
- b. Masuk ke direktori kerja 'C:\OS', dengan perintah 'cd os' <ENTER>.
- c. Masukan perintah dir, untuk melihat isi direktori di dalam folder tersebut. Akan muncul seperti di tampilkan pada **Error! Reference source not found..**

```
C:\OS>dir
Volume in drive C is S3A4070D001
Volume Serial Number is A8E5-3D6F

Directory of C:\OS

14/12/2008  03:48    <DIR>
14/12/2008  03:48    <DIR>
14/12/2008  03:43    <DIR>
14/12/2008  03:44    <DIR>
14/12/2008  03:48    <DIR>
14/12/2008  03:47      .
      .          Bochs-2.3.5
      .          Dev-CPP
      .          Lab
      57 Setpath.bat
      1 File(s)   57 bytes
      5 Dir(s)  15.977.971.712 bytes free

C:\OS>
```

Gambar 1.1 Struktur direktori kerja

- d. Jalankan file setpath, untuk menjalankannya ketik 'setpath' tekan <ENTER>
File 'setpath.bat' digunakan untuk mengatur lingkungan kerja ('path') selama anda melakukan praktikum, anda harus menjalankan progam ini sebelum memulai setiap sesi praktikum anda, untuk menjalakannya ketik 'setpath' tekan <ENTER>. Perhatikan teks yang muncul di layar, seperti ditampilkan pada **Error! Reference source not found..**.

Untuk melihat script yang terdapat di dalam file 'setpath.bat' dapat digunakan perintah 'type setpath.bat', cobalah. 'PATH' dapat bersisi banyak daftar lokasi, di antara item lokasi di batasi dengan karakter ';' (titik koma). Variabel 'PATH' ini akan digunakan oleh windows untuk mencari lokasi file yang dipanggil oleh 'user'. Urutan pencarian dimulai dari daftar lokasi yang paling awal.



Gambar 1.2 Pengaturan lingkungan kerja (path).

Untuk pengaturan ‘path’ seperti ditampilkan pada **Error! Reference source not found.**, maka ketika user memasukan perintah (seperti memanggil program) pertama kali yang dilakukan Windows adalah mencari file program di dalam direktori kerja saat itu, jika di sana tidak ditemukan file yang dimaksud, selanjutnya Windows akan meneruskan pencarian file di lokasi yang terdaftar pada variabel ‘PATH’ dimulai dari lokasi ‘C:\OS\Dev-Cpp\bin’ jika file belum ditemukan pencarian dilanjutkan ke lokasi ‘C:\Windows’, pencarian terus dilakukan sampai ke lokasi terakhir ‘C:\Windows\system32’. Jika sampai lokasi terakhir file tidak juga ditemukan maka windows akan manmpilkan informasi bahwa file yang dimaksud tidak ada di dalam sistem. Sebaliknya jika file sudah ditemukan maka windows akan menghentikan pencarian dilokasi terakhir ditemukannya file yang dimaksud.

Selain file pengatur lingkungan kerja, pada direktori kerja anda juga terdapat tiga folder yaitu folder ‘Bochs-2.3.5’, merupakan lokasi dari program ‘PC-Simulator’ yang akan digunakan untuk membuat PC-virtual, untuk menjalankan program ‘bootstrap loader’ dan program-program yang anda buat selama menjalankan praktikum.

Folder ‘Dev-Cpp’ berisi file program kompiler ‘nasm’ dan ‘gcc’ yang dapat digunakan untuk mengkompilasi program yang ditulis dalam bahasa c, c++ maupun bahasa assembly, selain itu pada folder ini juga terdapat beberapa program bantu, seperti ‘dd.exe’ untuk menyalin byte data dari satu file ke file yang lainnya. File-file kerja selama melakukan praktikum diletakan di bawah folder ‘Lab’

Melihat isi direktori kerja

Untuk masuk diirektori kerja untuk modul ini pertama adalah

- Jalankan command prompt
- Masuk ke direktori kerja pada ‘C:\OS\LAB\LAB1’.
- Cobalah untuk membuka file tersebut dengan perintah berikut, dari ‘COMMAND PROMPT’, ketikan ‘Notepad boot.asm’ dan tekan <ENTER>. Saat ini

sebaiknya anda tidak melakukan modifikasi terhadap program tersebut. Tutuplah kembali program ‘notepad’-nya. Source code prototype program bootloader disimpan dalam file ‘boot.asm’ sedangkan untuk source code prototype program kernel disimpan dalam file ‘kernel.asm’ listing lengkap keduanya bisa dibaca di lampiran. Sebaiknya anda memahami jalannya kedua program sebelum melakukan praktikum. Kedua program ditulis dalam bahasa assembly, anda dapat membukanya dengan menggunakan program aplikasi teks-editor sembarang, termasuk ‘Notepad.exe’ milik windows juga dapat digunakan.

- d. Selain kedua file source code ada sebuah file ‘image floppy’ dengan nama file ‘floppya.img’ file ini yang akan digunakan untuk menyimpan hasil kompilasi kedua source code, kemudian digunakan sebagai ‘boot disk’ pada PC-simulator ‘Bochs’. Sebagai kompiler digunakan program ‘nasm.exe’ yang terletak dalam subdirektori ‘C:\OS\Dev-Cpp\bin’ dan pada direktori tersebut juga ada program ‘dd.exe’ untuk memindahkan byte data dari satu file ke file yang lain, program ‘make.exe’ untuk mempercepat proses kompilasi. Pada saat ini kita akan menggunakan ketiga program tersebut yang dikemas dalam script ‘Makefile’ sehingga proses kompilasi menjadi lebih cepat, cukup dengan memanggil ‘make’. Script ‘makefile’ berupa file dengan format teks yang dapat anda sunting dengan menggunakan program aplikasi teks-editor.

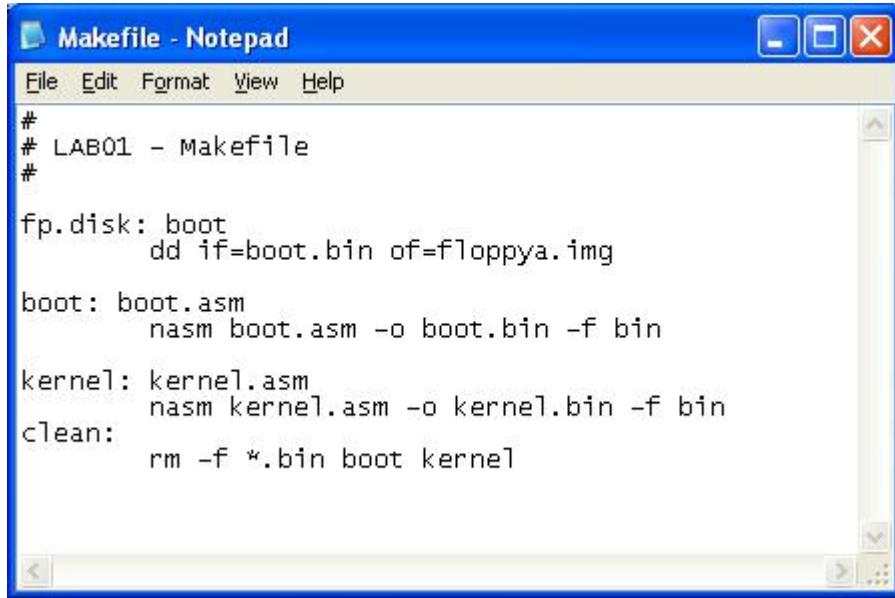
Sekilas tentang Makefile

‘Makefile’ adalah sebuah file teks yang berisi kumpulan perintah ‘Command Prompt’ mirip seperti BAT file, namun tidak dapat dijalankan secara langsung. File ini digunakan oleh program aplikasi ‘make.exe’ untuk mengendalikan proses kompilasi, lingking dan lain-lain, program ‘make.exe’ akan lebih terasa manfaatnya ketika source code program yang dibuat terdiri dari banyak file, proses kompilasi dan yang lain dapat dilakukan secara otomatis, sebagai kontrolnya adalah script yang terdapat pada ‘Makefile’. Cara ini banyak digunakan oleh para pengguna sistem operasi ‘linux’.

Seakarnya bukalah file ‘Makefile’, dari ‘Command Prompt’ untuk mengetahui script makefile dengan cara:

- a. bukalah direktori kerja anda ‘C:\OS\LAB\LAB1’ selanjutnya ketik ‘Notepad M’ tekan tombol ‘TAB’ sehingga muncul ‘Notepad Makefile’ dan tekan

<ENTER>. Jika langkah anda benar akan ditampilkan windows ‘Notepad’ dengan file ‘Makefile’ yang siap di sungting seperti ditampilkan pada .



```

Makefile - Notepad
File Edit Format View Help
#
# LAB01 - Makefile
#
fp.disk: boot
    dd if=boot.bin of=floppya.img

boot: boot.asm
    nasm boot.asm -o boot.bin -f bin

kernel: kernel.asm
    nasm kernel.asm -o kernel.bin -f bin
clean:
    rm -f *.bin boot kernel

```

Gambar 1.3 Menyunting file ‘Makefile’

Tanda karakter ‘#’ digunakan untuk menyisipkan baris komentar atau keterangan, karakter ini harus diletakkan pada kolom pertama, anda dapat menambahkan keterangan sebanyak yang anda inginkan.

Label yang berakhir dengan ‘:’, seperti ‘fp.disk:’ dan ‘boot:’ adalah merupakan kontrol proses (mirip menu) yang dapat dipilih saat menjalankan program ‘make.exe’. Anda dapat mengatur proses dengan menggunakan menu ini.

Perhatikan baris menu ‘fp.disk: boot’, baris ini memerintahkan pada komputer untuk mengerjakan baris perintah di bawah menu tersebut, tetapi setelah baris perintah yang terdapat di bawah menu ‘boot:’ dikerjakan. Jadi jika dilihat menu ‘fp.disk’ maka pertama kali ‘make.exe’ akan mengerjakan baris perintah yang terdapat di bawah label menu ‘boot:’ yaitu ‘nasm boot.asm -o boot.bin -f bin’, baris ini merupakan perintah untuk melakukan proses kompilasi program assembly dengan menggunakan program kompiler ‘nasm’. Sebagai source code adalah file ‘boot.asm’ kemudian hasil kompilasi akan dituliskan dalam output file (‘-o’) dengan nama ‘boot.bin’. sedangkan format (‘-f’) file output adalah dipilih dengan format ‘bin’. Program hasil kompilasi ini (‘boot.bin’) tidak dapat dijalankan secara langsung dari ‘Command Prompt’ karena tidak memiliki format ‘header’ yang dapat dimengerti oleh OS. Selanjutnya ‘make.exe’ akan melanjutkan pekerjaannya untuk memproses perintah yang

terletak di bawah menu ‘fp.disk:boot’ yaitu ‘dd if=boot.bin of=floppya.img’ ini adalah perintah untuk menyalin byte data dari file ‘boot.ini’ de dalam file ‘floppya.img’ pada lokasi sektor nomor ‘0’, sebanyak ‘512’ byte. Program aplikasi ‘dd.exe’ merupakan program bawaan linux yang di kemas ulang sehingga dapat dijalankan di windows, program bantu ini juga bersifat open source, banyak variasi yang dapat dilakukan oleh ‘dd’ tetapi tidak di bahas dalam modul ini anda dapat mencarinya di internet lewat yahoo atau google dan masukkan kata kunci ‘dd command’, cara ini juga berlaku untuk tema program yang lainnya.

File ‘Makefile’ akan di baca secara otomatis saat program ‘make’ dijalankan, menu dipilih saat menjalankan program make misalnya ‘make fp.disk’ berarti kita akan melakukan kompilasi hadap source code program ‘boot.asm’, sebagai outputnya file ‘boot.bin’ dan isinya disalin ke dalam bootsector file image floppy ‘floppya.img’.

Sekarang bukalah ‘Command Prompt’ dan buka direktori kerja ‘LAB1’ selanjutnya ketik ‘make fp.disk’, jika tidak ada kesalahan akan ditampilkan informasi seperti pada gambar .

```
C:\OS\Lab\LAB1>make fp.disk
nasm boot.asm -o boot.bin
dd if=boot.bin of=floppya.iso
rawwrite dd for windows version 0.5.
written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See
1+0 records in
1+0 records out
C:\OS\Lab\LAB1>
```

Gambar 1.4 Menggunakan perintah Make

Periksa hasil kompilasi dengan memasukan perintah ‘dir’, sekarang pada direktori kerja terdapat tambahan file baru, yaitu ‘boot.bin’ dan isinya sudah disalin kedalam bootsector ‘floppya.img’. Anda juga dapat melakukannya secara manual dari ‘Command Prompt’, pertama jalankan perintah baris satu yaitu ‘nasm boot.asm -o boot.bin -f bin’, jika tidak ada berita kesalahan lanjutkan dengan perintah ‘dd if=boot.bin of=floppya.img’. Selanjutnya silahkan di pelajari menu-menu lain dalam ‘makefile’ proses ini akan dilakukan berulang kali dalam praktikum-praktikum sealnjutnya. Perintah ‘rm ...’ pada baris paling bawah dalam ‘Makefile’ di atas,

digunakan untuk menghapus file-file yang sudah tidak terpakai. Daftar nama file yang dihapus diletakkan di belakang tanda (flag) ‘-f’ , kemudian diantara nama-nama file dipisahkan dengan ‘spasi’, anda dapat mencobanya dengan perintah ‘make clean’, periksa isi file dalam direktori kerja, sekarang ‘boot.bin’ sudah terhapus.

Mengenal ‘BOOT DISK’

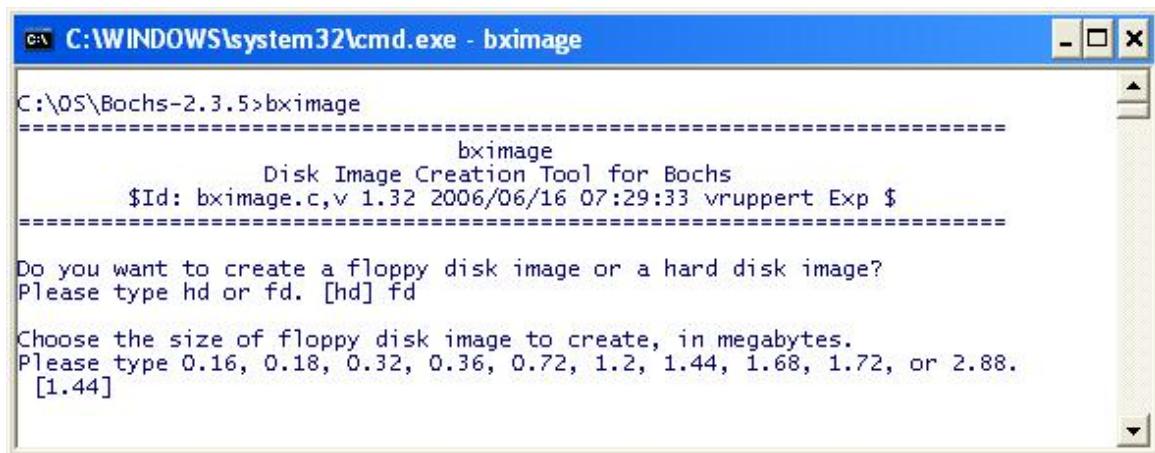
File ‘floppya.img’ adalah file image (bukan gambar), sebuah file yang isinya diformat seperti format floppy disk kapasitas 1.44M yang akan digunakan sebagai ‘bootdisk’ pada PC-Simulator, anda juga dapat memindahkan isi file image ini ke dalam floppy disk sebenarnya dengan menggunakan program bantu ‘rawrite’ atau yang lainnya. File image ini dapat dibuat dengan menggunakan program bantu bawaan ‘Bochs’ yaitu ‘bximage’ atau menggunakan program aplikasi ‘Virtual PC’ milik Windows (*free* juga) atau dengan program ‘Magiciso’. Sekarang kita coba membuat file image floppy baru dengan menggunakan program aplikasi ‘bximage.exe’, lakukan urutan perintah berikut.

- Hapuslah file ‘floppya.img’ jika sudah ada pada direktori kerja anda, dari ‘Command Prompt’ (lakukan dari direktori kerja) ketik ‘del floppya.img /P’ lanjutkan dengan tekan ‘Y’ dan <ENTER>. Pastikan bahwa file sudah benar-benar terhapus dengan perintah ‘dir’. Selanjutnya panggil ‘bximage’ sehingga ditampilkan window seperti pada **Error! Reference source not found..**

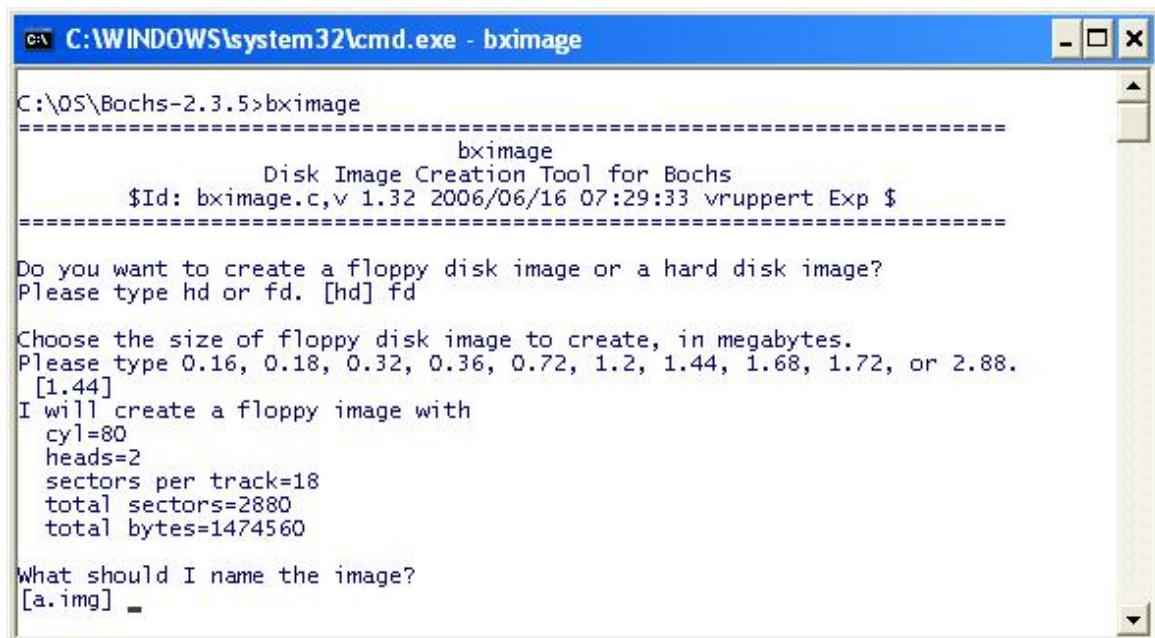


Gambar 1.5 Membuat file image floppy tahap-1

- Ada dua pilihan file image yaitu [hd] untuk membuat harddisk image atau [fd] untuk membuat floppy image. Kita akan membuat floppy image (karena ukurannya lebih kecil), selanjutnya ketikan ‘fd’ dan tekan <ENTER> sehingga muncul tahapan berikutnya seperti pada **Error! Reference source not found..**.



Gambar 1.6 Membuat file image floppy tahap-2



Gambar 1.7 Membuat file image floppy tahap-3.

```
C:\OS\Bochs-2.3.5>bximage
=====
      bximage
      Disk Image Creation Tool for Bochs
      $Id: bximage.c,v 1.32 2006/06/16 07:29:33 vruppert Exp $

Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] fd

Choose the size of floppy disk image to create, in megabytes.
Please type 0.16, 0.18, 0.32, 0.36, 0.72, 1.2, 1.44, 1.68, 1.72, or 2.88.
[1.44]
I will create a floppy image with
cyl=80
heads=2
sectors per track=18
total sectors=2880
total bytes=1474560

What should I name the image?
[a.img] floppya.img

Writing: [] Done.

I wrote 1474560 bytes to floppya.img.

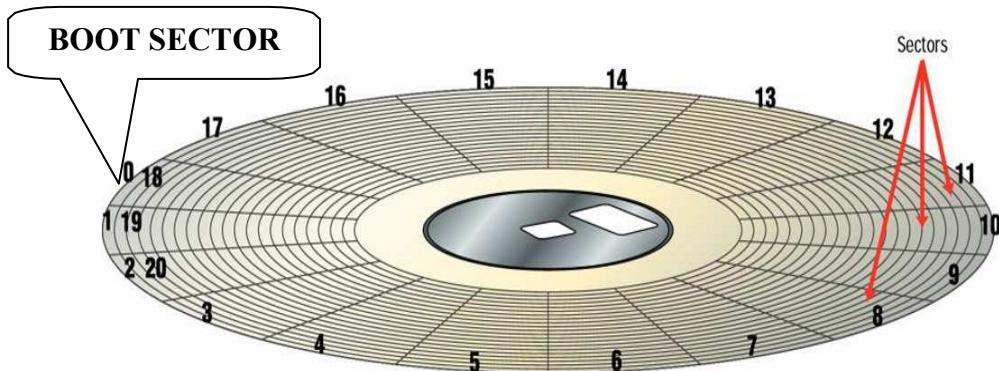
The following line should appear in your bochsrc:
  floppya: image="floppya.img", status=inserted
(The line is stored in your windows clipboard, use CTRL-V to paste)

Press any key to continue
```

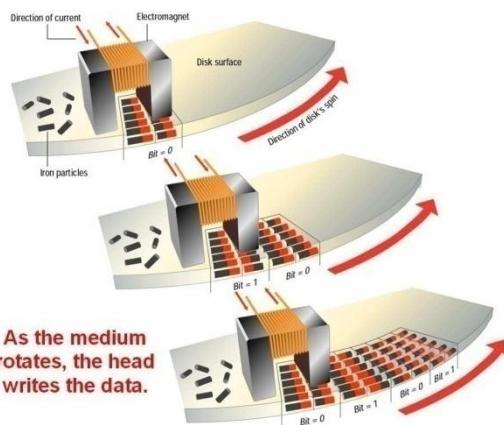
Gambar 1.8 Membuat file image floppy tahap-akhir.

- (c) Ada beberapa tipe (kapasitas) yang ditawarkan, pilih tipe yang paling banyak digunakan saat ini yaitu tipe floppy dengan kapasitas ‘1.44MB’, ditunjukan oleh angka [1.44]. Selanjutnya tekan <ENTER>, untuk memilih tipe lain, masukan angka yang menunjukkan kapasitas yang anda inginkan (lihat gambar di atas) dan <ENTER>. Pada windows akan di tampilkan berita seperti terlihat pada **Error! Reference source not found.** .
- (d) Terakhir anda diminta untuk memberikan nama file, ketikan ‘floppya.img’ dan <ENTER>, untuk saat ini jangan memberikan nama yang lain karena setting konfigurasi pada ‘Bochs’ sudah diatur menggunakan nama tersebut, jadi pastikan bahwa nama file imagnya adalah ‘floppya.img’, pastikan keberadaan file image tersebut dengan perintah ‘dir’, sudah adakah? Jika belum, anda harus mengulangi proses dari awal dengan memanggil program ‘bximage’.

Formatted Disk



Gambar 1.9 Format floppy disk FAT12



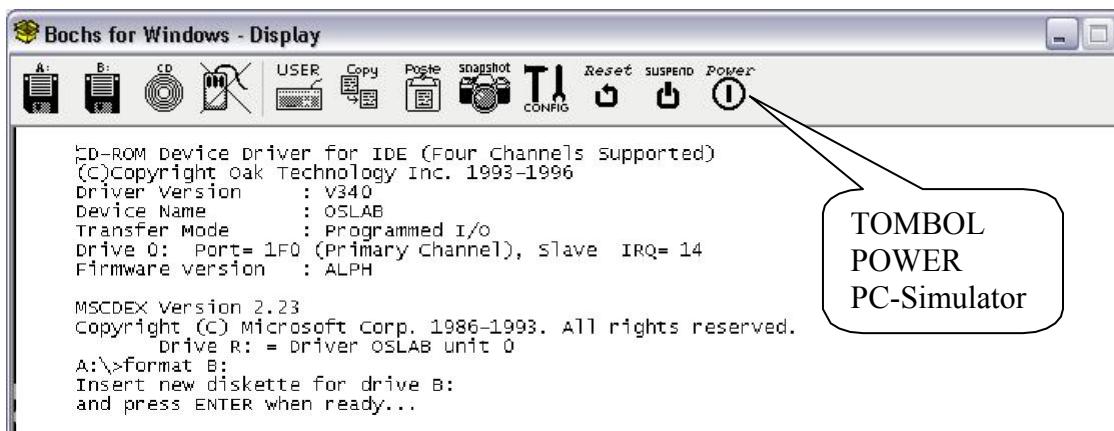
Gambar 1.10 Teknik penyimpanan data pada piringan DISK MAGNETIK.

File image floppy ini memiliki tipe Format FAT12, yang memiliki 80 cylinder, 2 head (atas dan bawah), 18 sektor tiap track, jumlah sektor 2880, jumlah byte data tiap sektor 512 byte sehingga total byte $2880 \times 512 = 1.474.560$ byte (1,44M). Format floppy secara fisik dapat dilihat pada **Error! Reference source not found.**, periksa dengan data di atas, samakah? Sistem yang serupa juga digunakan pada harddisk tetapi dengan variasi jumlah cylinder, head dan sector per track (disingkat CHS) yang berbeda. Cylinder adalah jumlah track (garis melingkar) pada setiap permukaan piringan, jumlah permukaan piringan ada 2 (atas dan bawah, 2 head) sehingga pada floppy 1.44M setiap permukaan terdapat 40 cylinder. Apakah anda dapat melihat track pada floppy yang sebenarnya, tanpa alat bantu? Bayangkan ukuran track yang terdapat pada harddisk dengan kapasitas 300G....dapatkah anda menghitung ukuran fisik (ketebalan garis) tiap track?

Perlu diingat bahwa File image floppy ‘floppya.img’ berada dalam direktori Bochs. Dan selanjutnya yang berkenaan dengan file ini kita harus merujuknya ke lokasi yang tepat.

File image floppy ‘floppya.img’ yang dibuat dengan langkah di atas belum dapat digunakan, karena belum di ‘FORMAT’, seperti pada floppy sebenarnya jika belum di format floppy tidak dapat dibaca. Langkah –langkah Untuk memformat ‘floppya.img’ adalah :

- a. jalankan PC-Simulator dari ‘Command Prompt’ dengan perintah ‘DOSFP’,
- b. pada konfigurasi PC-Simulator file ‘floppya.img’ terpasang pada ‘drive B:’. Selanjutnya dari prompt ‘A:>’ ketikan ‘Format B:’ <ENTER> [2x], seperti ditampilkan pada **Error! Reference source not found..**



Gambar 1.11 Bochs PC-Simulator

- c. Tutup kembali PC-Simulator dengan klik pada tombol power, di bagian menu atas-kanan. Sekarang ‘floppya.img’ sudah terformat dan dapat digunakan untuk menyimpan data, namun belum dapat digunakan untuk ‘booting’, karena pada ‘bootsector’ belum diisi program ‘bootloader’. Untuk lebih detail mengenai pembahasan ini ada dalam Modul kedua.

Melihat data dalam boot sector

Untuk membuat sistem operasi (OS – Operating system) salah satu tahap terpenting adalah menyusun program bootstrap-loader (akan dipraktikan mulai modul praktikum kedua). Ukuran program ini tidak besar, hanya 512 byte dan tersimpan dalam lokasi sektor 0 (track 0 dan cylender 0) disebut BOOT-SECTOR, penomoran dimulai dari ‘0’. Seperti terlihat pada **Error! Reference source not found..** Data pada lokasi ini hanya di

baca oleh PC pada saat proses ‘booting’ setelah itu PC tidak akan pernah membaca lagi, sampai Power OFF.

Tabel 1.1 Struktur data BOOTSECTOR

Offset (Hex)	Data
0000-0002	Jump ke lokasi Boot Code
0003-003D	Parameter DISK
003E-01FD	Boot Code (program boot)
01FE-01FF	Signature (ID bootable)

Struktur data dalam Bootsector seperti ditunjukkan pada

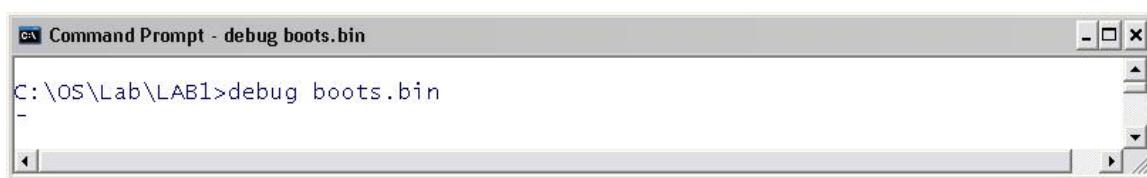
. Tiga byte pertama harus berupa kode program untuk meloncat (‘jump’) ke posisi 0x003E (hex). Karena data pertama bootsector akan diletakan pada alamat 0x7C00 pada memori kerja komputer (RAM) (simbol ‘0x’ adalah simbol untuk menunjukkan format angka hex, banyak digunakan dalam bahasa C atau Assembly, jadi 0x7C00 = 7C00 hexadecimal), maka instruksi pertama adalah ‘jump 0x7C3E’ dimana pada area tersebut tersimpan program bootstrap-loader. Lokasi data offset 0x0003-0x003D sebanyak 59 byte digunakan untuk menerangkan format disk, jumlah sektor per track, jumlah FAT (*Field Allocation Table*) kode penunjuk urutan lokasi cluster (sektor) data dari sebuah file dan sebagainya (selanjutnya dapat anda lihat pada source code ‘boot.asm’ bagian atas). ‘Boot code’ atau inti dari program ‘bootstrap loader’ diletakan pada lokasi byte mulai nomor 0x003E sampai dengan 0x01FD sebanyak 448 byte.

Dua byte terakhir berupa byte ‘Signature’ atau ID bahwa sebuah disk bersifat ‘Bootable’, yaitu data ‘0x55’ pada lokasi offset 510, sedangkan data ‘0xAA’ terletak pada lokasi offset 511, nomor lokasi data dimulai dari ‘0’ jadi total lokasi data dari 0 s/d 511 ada 512 byte. BIOS akan mengindifikasi byte ‘Signature’ untuk menentukan apakah sebuah disk bersifat ‘Bootable’ atau tidak. 0x55AA berarti disk bersifat ‘bootable’.

Untuk memeriksa data bootsector pada file image floppy dapat digunakan program ‘debug’ yang terdapat pada window dibantu program ‘dd.exe’, lakukan urutan perintah berikut ini. Copy 512 byte data bootsektor ke dalam sebuah file terpisah, caranya dari ‘Command Prompt’ ketik ‘dd if=floppya.img of=boots.bin count=1’ maksud dari perintah ini adalah menyalin byte data dari file ‘floppya.img’ ke dalam file

‘boots.bin’ sebanyak satu sektor mulai dari sektor 0. Hasilnya akan tersimpan dalam file bernama ‘boots.bin’ dengan ukuran data sebanyak 512 byte, ini merupakan data dalam bootsector pada file image ‘floppya.img’. Untuk melihat isinya gunakan program ‘debug’ dengan cara sebagai berikut.

Jalankan ‘Command Prompt’ dan ketik ‘debug boots.bin’ dan <ENTER>, tampilan windows seperti pada **Error! Reference source not found..** (Perlu diketahui untuk mengakhiri debug ketikkan ‘quit’ kemudian enter)



Gambar 1.12 Tampilan windows saat memulai program ‘debug’

Maksud perintah ini adalah memindah data dalam file ‘boots.bin’ ke dalam memori kerja ‘debug’ mulai dari alamat ‘0000:0100’. Debug hanya dapat bekerja dengan ukuran data/file maksimum sebesar 64KB, jadi dengan debug kita tidak bisa membuka file image floppy yang berukuran 1,4MB. file ‘boots.bin’ berukuran 512 B (atau 0x01FF) dan oleh debug data ini akan diletakkan di memori mulai dari alamat CS:0100 sampai dengan alamat CS:02FF. Simbol CS merupakan simbol register ‘CODE SEGMENT’, nilai CS tergantung pada kondisi memori komputer saat itu.

Untuk melihat perintah-perintah yang disediakan dalam ‘debug’ ketikan karakter ‘?’ Untuk melihat isi file ‘boots.bin’ ketikan perintah ‘D CS:0100 02FF’ dan tekan <ENTER>. Maksud perintah ini adalah manampilkan isi memori mulai CS:0100 sebanyak 512 byte dalam bentuk hex, seperti terlihat pada **Error! Reference source not found..** Angka pada kolom paling kiri menunjukkan lokasi alamat memori yang digunakan untuk menampung data ‘boots.bin’. Register CS menunjuk segmen alamat ‘13EA’, 16 kolom angka di sampingnya adalah data ‘boots.bin’ yang ditampilkan dalam format hex dikelompokan per byte, 2 angka hexa = 1 byte. Kelompok karakter pada kolom paling kanan adalah data ‘boots.bin’ yang ditampilkan sebagai ‘karakter’, karena data ini merupakan program maka karakter yang terlihat terkesan tidak beraturan, bahkan ada data yang tidak dapat ditampilkan sebagai karakter sehingga hanya muncul tanda ‘...’. Dapatkah anda mengidentifikasi ‘boot signature’ 0x55AA’, tunjukkan posisi alamatnya..!

Cara lain dapat digunakan program ‘tdump.exe’ milik borland, dari ‘Command Prompt’ ketikan ‘tdump boots.bin’ <ENTER> deretan data yang serupa dengan gambar 11 akan ditampilkan, seperti ditampilkan pada **Error! Reference source not found.**. Perbedaan terletak pada kolom paling kiri, dengan ‘tdump’ kolom paling kiri menunjukkan posisi relatif setiap byte data dimulai dari posisi ‘000000’.

```

C:\OS\Lab\LAB1>debug boots.bin
-D cs:0100 02FF
13EA:0100 EB 3C 90 4D 53 57 49 4E-34 2E 31 00 02 01 01 00 .<.MSWIN4.1.....
13EA:0110 02 E0 00 40 0B F0 09 00-12 00 02 00 00 00 00 00 ....@.....;.....
13EA:0120 00 00 00 00 00 29 EB-0C 6E 27 46 42 20 20 20 20 .....).n'FB
13EA:0130 20 20 20 20 20 20 46 41-54 31 32 20 20 20 33 C9 FAT12 3.
13EA:0140 8E D1 BC FC 7B 16 07 BD-78 00 C5 76 00 1E 56 16 ....{...x..v..V.
13EA:0150 55 BF 22 05 89 7E 00 89-4E 02 B1 0B FC F3 A4 06 U."..~..N.....
13EA:0160 1F BD 00 7C C6 45 FE 0F-38 4E 24 7D 20 8B C1 99 ...|.E..8N$} ...
13EA:0170 E8 7E 01 83 EB 3A 66 A1-1C 7C 66 3B 07 8A 57 FC ~....:f..|f;..w.
13EA:0180 75 06 80 CA 02 88 56 02-80 C3 10 73 ED 33 C9 FE u.....V....s.3..
13EA:0190 06 D8 7D 8A 46 10 98 F7-66 16 03 46 1C 13 56 1E ..}.F....f..F..V.
13EA:01A0 03 46 0E 13 D1 8B 76 11-60 89 46 FC 89 56 FE B8 .F....v.`.F..V..
13EA:01B0 20 00 F7 E6 8B 5E 0B 03-C3 48 F7 F3 01 46 FC 11 ....A...H...F.:
13EA:01C0 4E FE 61 BF 00 07 E8 28-01 72 3E 38 2D 74 17 60 N.a....(.r>8-t.
13EA:01D0 B1 0B BE D8 7D F3 A6 61-74 3D 4E 74 09 83 C7 20 ....}..at=Nt...
13EA:01E0 3B FB 72 E7 EB DD FE 0E-D8 7D 7B A7 BE 7F 7D AC ;.r.....}{....}.
13EA:01F0 98 03 F0 AC 98 40 74 0C-48 74 13 B4 0E BB 07 00 ....@t.Ht.....
13EA:0200 CD 10 EB EF BE 82 7D EB-E6 BE 80 7D EB E1 CD 16 ....}....}....}.....
13EA:0210 5E 1F 66 8F 04 CD 19 BE-81 7D 8B 7D 1A 8D 45 FE ^.f.....}....}....E.
13EA:0220 8A 4E 0D F7 E1 03 46 FC-13 56 FE B1 04 E8 C2 00 .N....F..V.....
13EA:0230 72 D7 EA 00 02 70 00 52-50 06 53 6A 01 6A 10 91 r....p.RP.Sj.j..
13EA:0240 8B 46 18 A2 26 05 96 92-33 D2 F7 F6 91 F7 F6 42 .F.&...3....B
13EA:0250 87 CA F7 76 1A 8A F2 8A-E8 C0 CC 02 0A CC B8 01 ...v.....;.....
13EA:0260 02 80 7E 02 0E 75 04 B4-42 8B F4 8A 56 24 CD 13 ...~.u..B..V$..
13EA:0270 61 61 72 0A 40 75 01 42-03 5E 0B 49 75 77 C3 03 aar.@u.B.^Iuw..
13EA:0280 18 01 27 0D 0A 49 6E 76-61 6C 69 64 20 73 79 73 ...'.Invalid sys
13EA:0290 74 65 6D 20 64 69 73 6B-FF 0D 0A 44 69 73 6B 20 tem disk...Disk
13EA:02A0 49 2F 4F 20 65 72 72 6F-72 FF 0D 0A 52 65 70 6C I/O error...Repl
13EA:02B0 61 63 65 20 74 68 65 20-64 69 73 6B 2C 20 61 6E ace the disk, an
13EA:02C0 64 20 74 68 65 6E 20 70-72 65 73 73 20 61 6E 79 d then press any
13EA:02D0 20 6B 65 79 0D 0A 00 00-49 4F 20 20 20 20 20 20 key....IO
13EA:02E0 53 59 53 4D 53 44 4F 53-20 20 20 53 59 53 7F 01 SYSMSDOS  SYS..
13EA:02F0 00 41 BB 00 07 60 66 6A-00 E9 3B FF 00 00 55 AA .A...`fj...;...U.
-
```

Gambar 1.13 data dalam bootsector file floppya.img

```
C:\OS\Lab\LAB1>tdump boots.bin
Turbo Dump Version 5.0.16.12 Copyright (c) 1988, 2000 Inprise Corporation
Display of File BOOTS.BIN

000000: EB 3C 90 4D 53 57 49 4E 34 2E 31 00 02 01 01 00 .<.MSWIN4.1.....
000010: 02 E0 00 40 0B F0 09 00 12 00 02 00 00 00 00 00 ...@.....
000020: 00 00 00 00 00 00 29 EB 0C 6E 27 46 42 20 20 20 .....).n'FB
000030: 20 20 20 20 20 20 46 41 54 31 32 20 20 20 33 C9 FAT12 3.
000040: 8E D1 BC FC 7B 16 07 BD 78 00 C5 76 00 1E 56 16 ....{...x..v..V.
000050: 55 BF 22 05 89 7E 00 89 4E 02 B1 0B FC F3 A4 06 U."...~..N.....
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 7D 20 8B C1 99 ...|.E..8N$} ...
000070: E8 7E 01 83 EB 3A 66 A1 1C 7C 66 3B 07 8A 57 FC .~...:f..|f;..w.
000080: 75 06 80 CA 02 88 56 02 80 C3 10 73 ED 33 C9 FE u.....V....s.3..
000090: 06 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E ..}.F...f..F..V.
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F....v..F..V..
0000B0: 20 00 F7 E6 8B 5E 0B 03 C3 48 F7 F3 01 46 FC 11 ....A...H...F.:
0000C0: 4E FE 61 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(.r>8-t.
0000D0: B1 0B BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 .....}.at=Nt...
0000E0: 3B FB 72 E7 EB DD FE 0E D8 7D 7B A7 BE 7F 7D AC ;.r.....}{...}.
0000F0: 98 03 F0 AC 98 40 74 0C 48 74 13 B4 0E BB 07 00 .....@.Ht.....
000100: CD 10 EB EF BE 82 7D EB E6 BE 80 7D EB E1 CD 16 .....}....}.
000110: 5E 1F 66 8F 04 CD 19 BE 81 7D 8B 7D 1A 8D 45 FE A.f.....}.}..E.
000120: 8A 4E 0D F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .N....F..V.....
000130: 72 D7 EA 00 02 70 00 52 50 06 53 6A 01 6A 10 91 r....p.RP.Sj.j..
000140: 8B 46 18 A2 26 05 96 92 33 D2 F7 F6 91 F7 F6 42 .F..&...3.....B
000150: 87 CA F7 76 1A 8A F2 8A E8 C0 CC 02 0A CC B8 01 ...v.....
000160: 02 80 7E 02 0E 75 04 B4 42 8B F4 8A 56 24 CD 13 ...~..u..B...V$..
000170: 61 61 72 0A 40 75 01 42 03 5E 0B 49 75 77 C3 03 aar.@u.B.A.Iuw..
000180: 18 01 27 0D 0A 49 6E 76 61 6C 69 64 20 73 79 73 ...'.Invalid sys
000190: 74 65 6D 20 64 69 73 6B FF 0D 0A 44 69 73 6B 20 tem disk...Disk
0001A0: 49 2F 4F 20 65 72 72 6F 72 FF 0D 0A 52 65 70 6C I/O error...Repl
0001B0: 61 63 65 20 74 68 65 20 64 69 73 6B 2C 20 61 6E ace the disk, an
0001C0: 64 20 74 68 65 6E 20 70 72 65 73 73 20 61 6E 79 d then press any
0001D0: 20 6B 65 79 0D 0A 00 00 49 4F 20 20 20 20 20 20 key....IO
0001E0: 53 59 53 4D 53 44 4F 53 20 20 20 53 59 53 7F 01 SYSMSDOS  SYS..
0001F0: 00 41 BB 00 07 60 66 6A 00 E9 3B FF 00 00 55 AA .A...`fj.;....U.
```

Gambar 1.14 Melihat data bootsector dengan program ‘tdump.exe’

‘Boot’ PC-Simulator dengan file image ‘floppya.img’

Salah satu file dalam direktori kerja adalah file ‘s.bat’, berisi dua baris perintah untuk memanggil PC-Simulator ‘Bochs’. Lihat isi file ‘s.bat’ dengan perintah ‘type s.bat’ <ENTER>, hasilnya terlihat seperti pada .

```
C:\OS\Lab\LAB1>type s.bat
..\..\bochs-2.3.5\bochs -q -f bochssrc.bxrc
C:\OS\Lab\LAB1>
```

Gambar 1.15 Isi file ‘s.bat’

Sebuah baris perintah untuk memanggil ‘Bochs’ (PC-Simulator), dengan file konfigurasi ‘bochssrc.bxrc’. File konfigurasi ini berupa file teks yang dapat di sunting dengan menggunakan ‘notepad’, digunakan untuk mengatur konfigurasi PC yang akan disimulasi seperti file ROM BIOS dan ROM VGA yang digunakan, kapasitas ‘RAM’, perangkat-

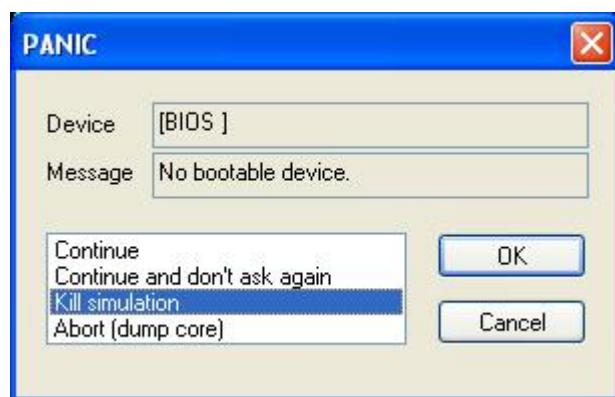
perangkat pendukung seperti floppy, harddisk dan lain-lain. Saat ini kita belum perlu merubah konfigurasi tersebut biarkan apa adanya. Pada konfigurasi ini ‘floppya.img’ di pasang pada ‘DRIVE A:’ dan digunakan sebagai ‘boot disk’.

Selanjutnya masukan perintah ‘s’ <ENTER>, akan ditampilkan windows ‘Bochs for windows – display’ yang sedang melakukan proses ‘booting’ namun tidak berhasil karena tidak menemukan diskboot, seperti ditampilkan pada teks di bawah

```
Booting from Floppy...

Invalid system disk
Replace the disk, and then press any key
```

Atau tampil seperti berikut:



‘Floppya.img’ belum terisi ‘system file’ atau ‘kernel’ sehingga proses boot gagal. Sekarang kita akan format ‘floppya.img’ dan menambahkan ‘system file’ ke dalamnya. Panggil ‘DosFp’ <ENTER>.

Pada windows ‘Bochs’ masukan perintah ‘A:>format B:/S ’ <ENTER>, selesaikan proses format dan berikan nama label disk jika perlu. Jika sudah selesai, untuk memastikan bahwa floppy pada drive ‘B:’ terisi dengan ‘system file’, periksa dengan perintah berikut ‘A:>dir B:’ <ENTER>, jika tidak ada kesalahan pada windows ‘Bochs’ akan ditampilkan berita berikut:

```
A:\>dir B:

Volume in drive B is DOS
Volume Serial Number is 3256-1404
Directory of B:\
```

```
COMMAND   COM          94,292  05-05-03  21:22
          1 file(s)      94,292 bytes
          0 dir(s)       1,235,968 bytes free
```

A:\>

Matikan PC-Simulator (klik tombol Power Off), sekarang kita coba untuk menggunakan ‘floppya.img’ sebagai ‘boot disk’ lagi, ketik ‘s’ <ENTER>. Jika tidak ada kesalahan akan ditampilkan teks seperti berikut:

```
Booting from Floppy...
Starting BootCD.....
```

```
Microsoft (R) MS-DOS 7.1
(C) Copyright Microsoft Corp 1981-1999.
```

A:\>

Pada baris kedua terdapat teks ‘starting BootCD’ Ini disebabkan oleh pembuatan boot disk awal menggunakan sistem yang berasal dari CD. Namun proses boot sebenarnya saat ini berasal dari ‘Floppya.img’. Pada modul selanjutnya kita akan belajar untuk membuat ‘bootloader’ dan ‘system file’ ini.

Tugas

- (1) Apa yang dimaksud dengan kode ‘ASCII’, buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extended, tuliskan kode ASCII dalam format angka desimal, binary dan hexadesimal serta karakter dan simbol yang dikodekan.
- (2) Carilah daftar perintah bahasa assembly untuk mesin intel keluarga x86 lengkap (dari buku referensi atau internet). Daftar perintah ini dapat digunakan sebagai pedoman untuk memahami program ‘boot.asm’ dan ‘kernel.asm’.

Modul 2 : Mengenal Proses Pembuatan ‘DISK BOOT’

Tujuan

Memahami langkah-langkah pembuatan ‘disk boot’ (floppy) pada PC.

Pendahuluan

Proses boot adalah proses yang pertama kali terjadi dalam sistem PC. Ketika tombol ‘Power’ di tekan, mikroprosesor memasuki keadaan ‘reset’ dan pada saat itu semua register di dalam mikroprosesor berada dalam keadaan kosong. Selanjutnya kode program pada memori yang terletak pada alamat **0xFFFF** di ‘load’ (dipindahkan) ke dalam ‘Code Segmen’ dan dieksekusi. Program yang terletak pada alamat **0xFFFF** ini biasanya disimpan dalam sebuah chip ROM dan dikenal dengan nama *Basic Input Output System* (BIOS). Program yang pertama kali dijalankan oleh BIOS adalah memeriksa memori diteruskan dengan memeriksa semua perangkat bantu yang terhubung pada PC, seperti serial port, floppy dan sebagainya. Jika tidak ditemukan kesalahan fatal pada proses ini, BIOS melanjutkan prosesnya dengan mencari file sistem operasi (*Operating System – OS*) atau ‘Kernel’. Urutan pencarian OS sesuai dengan pengaturan prioritas, dapat dimulai dari floppy disk, hardisk, CDROM, bootrom dan seterusnya. Pada sistem operasi lama seperti MSDOS, semua proses ini dilakukan dalam mode real (*real-mode*), sedangkan pada OS modern salah satu proses yang dilakukan oleh ‘kernel’ adalah merubah mode kerja dari ‘*real-mode*’ ke dalam mode kerja ‘*protect-mode*’. Pada modul praktikum ini akan dipelajari OS yang selalu bekerja pada mode-kerja ‘*real-mode*’. Mode kerja ‘*protect-mode*’ akan di bahas pada modul praktikum selanjutnya.

BIOS tidak memindahkan seluruh file ‘kernel’ ke dalam memori kerja (RAM), hanya sebagian kecil program saja yang dipindahkan, yaitu program yang terletak pada sektor pertama pada floppy disk atau harddisk disebut ‘BOOT SECTOR’. Ukuran program pada boot-sector hanya 512 byte dan dua byte terakhir berisi data **0xAA55**, disebut ‘boot-signature’. Jika kode ‘boot-signature’ ini tidak ada maka disk dianggap tidak bersifat ‘bootable’ dan BIOS akan mencari media booting yang lainnya. Potongan program sepanjang 512 byte pada boot-sector ini disebut ‘BOOTSTRAP LOADER’. BIOS memindahkan program bootstrap-loader ke dalam RAM (memori kerja) dengan alamat **0x7C00** kemudian mengeksekusinya. Inilah yang disebut dengan proses boot atau sering

disebut booting. Sebelum melakukan praktikum sebaiknya anda memahami proses ini dengan baik.

Pada praktikum ini anda akan mencoba untuk membuat program bootstrap loader sederhana pada sebuah disk. Semua percobaan akan dilakukan dengan menggunakan program PC-Simulator open source dengan nama ‘Bochs’ untuk memahami lebih jauh tentang ‘Bochs’ anda dapat mengunjungi alamat web-site ini www.bochs.sourceforge.net

Peralatan

Sebuah PC dengan sistem operasi Windows Xp yang didalamnya terdapat program PC-Simulator ‘Bochs’, kompiler bahasa assembly ‘nasm’, kompiler bahasa C dari ‘Dev-Cpp’ (gcc versi windows) dan beberapa program bantu seperti ‘make’, ‘debug’, ‘dd’, dan ‘tdump’. Sebagai ‘editor-text’ untuk memodifikasi program digunakan program ‘Notepad’ yang terdapat pada sistem operasi Windows Xp.

Daftar perintah bahasa assembly mesin intel x86 (dari tugas modul-1), sebagai alat bantu.

Langkah kerja

- (1) Buka ‘Command Prompt’, atur ‘path’ dan pergi ke direktori kerja. Klik ‘Start|run’ ketik ‘cmd’ <ENTER>, pada windows ‘Command Prompt’ ketik ‘CD OS’, dan jalankan perintah ‘setpath’, terakhir ketik ‘cd LAB/LAB2’ dilanjutkan dengan perintah ‘DIR’. Prosedure ini adalah standar prosedure awal yang harus anda lakukan sebelum melakukan praktikum di setiap modul. Direktori kerja akan tampak seperti pada gambar berikut.

```
C:\OS\Lab\LAB2>dir
Volume in drive C is S3A4070D001
Volume Serial Number is A8E5-3D6F

Directory of C:\OS\Lab\LAB2

15/12/2008  13:54    <DIR>      .
15/12/2008  13:54    <DIR>      ..
15/12/2008  13:26            1.625 bochsrc.bxrc
15/12/2008  13:48            15.914 boot.asm
15/12/2008  13:52              78 dosfp.bat
14/12/2008  11:45            7.966 kernel.asm
15/12/2008  13:51            228 Makefile
15/12/2008  12:20            44 s.bat
                           6 File(s)   25.855 bytes
                           2 Dir(s)  15.831.035.904 bytes free

C:\OS\Lab\LAB2>
```

Gambar 2.1 Direktori kerja LAB2

- (2) Menyiapkan file ‘floppya.img’. Ingat namanya harus ‘floppya.img’. Jalankan ‘bxImage’, selanjutnya jawablah pertanyaan-pertanyaan yang muncul dengan urutan berikut ‘fd’, ‘1.44’ dan ‘floppya.img’. Lihat proses ini pada modul-1 jika perlu. Pastikan hasil file ‘floppya.img’, dengan memasukan perintah ‘dir’, hasilnya seperti pada **Error! Reference source not found.** berikut

```
Volume Serial Number is A8E5-3D6F

Directory of C:\OS\Lab\LAB2

15/12/2008  16:26    <DIR>      .
15/12/2008  16:26    <DIR>      ..
15/12/2008  16:18            1.625 bochsrc.bxrc
15/12/2008  16:05            15.916 boot.asm
15/12/2008  13:52              78 dosfp.bat
15/12/2008  16:05            1.474.560 floppya.img
14/12/2008  11:45            7.966 kernel.asm
15/12/2008  16:21            228 Makefile
15/12/2008  12:20            44 s.bat
                           7 File(s)   1.500.417 bytes
                           2 Dir(s)  15.819.517.952 bytes free

C:\OS\Lab\LAB2>
```

Gambar 2.2 Memastikan adanya file ‘floppya.img’

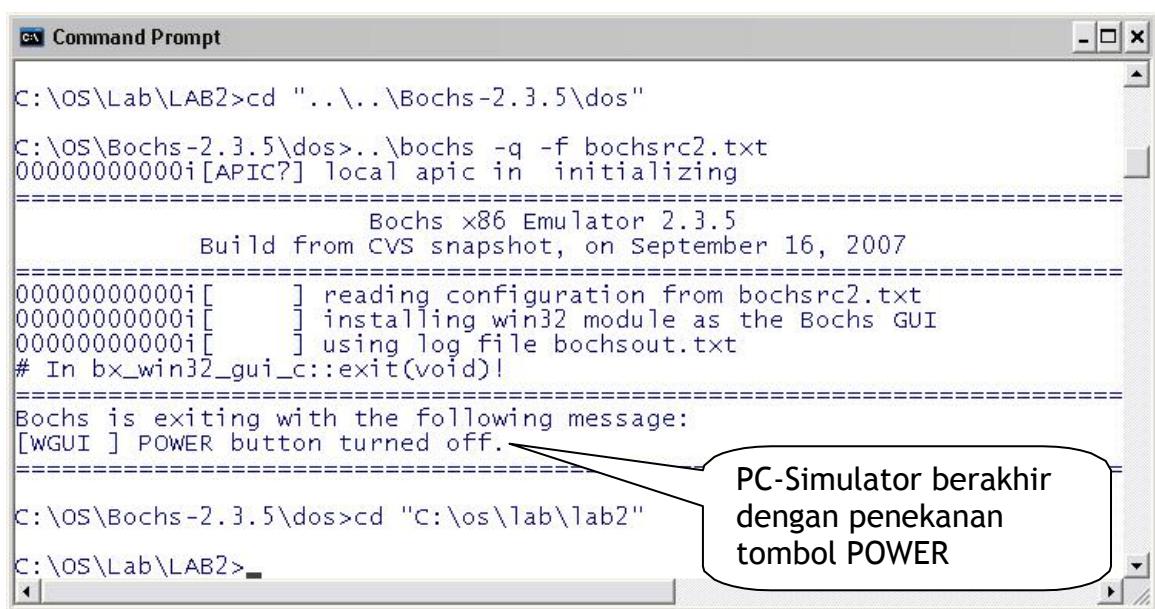
- (3) Mem-format ‘floppya.img’ dan mengisinya dengan sistem operasi DOS versi 7. jalankan perintah berikut: ‘dosfp’. Pindah ke windows ‘Bochs’ (PC Simulator), klik menu gambar floppy disk nomor dua dari kanan, seperti pada gambar berikut



- (4) Selanjutnya atur lokasi file image sehingga menunjuk ke file ‘floppya.img’ yang terdapat pada direktori kerja ‘LAB2’, klik tombol ‘BROWSE’ dan atur sehingga seperti tampak pada gambar di bawah, dan lanjutkan dengan klik ‘OK’.



- (5) Dari prompt ‘A:>’ ketikan perintah ‘A:>Format B: /S’ selesaikan prosesnya. Jika tidak ada kesalahan maka ‘floppya.img’ sekarang dapat digunakan untuk proses booting. Tutup windows ‘Bochs’, klik menu POWER. Pada window ‘Command Prompt’ akan tampak bekas akatifitas ‘Bochs’ seperti pada gambar berikut:



- (6) BOOT PC-simulator dengan file ‘floppya.img’. Pindah ke direktori kerja pada window ‘Command Prompt’ dan jalankan perintah ‘s’ <ENTER>. Sekarang PC-Simulator akan melakukan proses boot dengan disk boot yang berasal dari file ‘floppya.img’ yang diletakkan pada ‘dirve A:’, dan proses boot telah berjalan dengan lancar. Pastikan dengan menekan tombol gambar floppy yang tidak di silang (paling kiri). Tutup kembali PC-Simulator, klik tombol POWER.



- (7) Kompilasi source code ‘boot.asm’ dan memindah hasilnya ke bootsector ‘floppya.img’. Pindah ke direktori kerja ‘LAB2’ ketikan perintah ‘cls’ untuk membersihkan layar. Selanjutnya jalankan perintah ‘make fp.disk’ <ENTER> jika langkah ini berhasil, pada layar akan ditampilkan teks berikut:

```
C:\OS\Lab\LAB2>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppya.iso
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
1+0 records in
1+0 records out

C:\OS\Lab\LAB2>
```

- (8) Boot PC Simulator dengan program bootstraploader yang baru. Jalankan PC-Simulator ketik ‘s’ <ENTER>. Tampilan pada PC-Simulator sekarang adalah sebagai berikut:

```
Booting from Floppy...

Loading kernel ver 0.01
.....
ERROR : Press Any Key to Reboot
```

Bootloader telah berubah, sekarang telah menggunakan program bootstrap-loader yang berasal kompilasi file ‘boot.asm’ yaitu file ‘boot.bin’ yang telah disalin kedalam file ‘floppya.img’ tetapi proses boot gagal karena tidak ada file

‘KERNEL.BIN’ pada ‘floppya.img’. Kita akan buat file ‘KERNEL.BIN’ pada tahap berikutnya. Sekarang kita coba merubah tampilan program bootstrap- loader, tutup PC-Simultor, klik tombol POWER pada PC-Simultor.

- (9) Menyunting file ‘boot.asm’, ketikan ‘notepad boot.asm’ <ENTER>, cari teks ‘Loading kernel’ kelompok baris bawah, pada windows ‘Notepad’, tekan tombol ‘CTRL + F’ masukan kata kunci ‘Loading kernel’, <ENTER>, klik ‘CANCEL’. Tampilan pada Notepad tampak seperti gambar berikut:

```

boot.asm - Notepad
File Edit Format View Help
ImageName db "KERNEL BIN"
=====
; Teks yang akan ditampilkan saat mulai proses
; BOOT : Loading kernel ver 0.01
; Di awali dan diakhiri dengan tanda :
; -- 0xD akhir baris
; -- 0xA baris baru
; -- 0x00 Karakter 'NULL' pembatas dengan data di bawahnya
=====
msgLoading db 0xD, 0xA, "Loading kernel ver 0.01 ", 0xD, 0xA, 0x00
msgCRLF db 0xD, 0xA, 0x00
msgProgress db ".", 0x00

```

Teks yang ditampilkan pada awal proses BOOT.

Sekarang sunting baris dibelakang variabel ‘msgLoading’ sehingga menjadi seperti berikut

```
'msgLoading db 0xD, 0xA, "Belajar membuat BOOTSTRAP-LOADER",
0xD, 0xA, 0x00'
```

Buatlah sehingga menjadi satu baris. Anda juga dapat mengganti kata-kata ‘ERROR: Press Any Key to Reboot’ dengan cara yang sama, yang didimpan pada variabel ‘msgFailure’.

Simpan file ‘boot.asm’, tekan ‘CTRL+S’ pada notepad dan pindah ke window ‘Command Prompt’ direktori kerja, ulangi proses kompilasi file ‘boot.asm’, jalankan perintah ‘Make fp.disk’ <ENTER>, setelah proses kompilasi, jalanan PC-Simulator ‘s’<ENTER>. Sekarang teks yang ditampilkan di layar adalah seperti yang anda tuliskan pada file ‘boot.asm’ yaitu seperti berikut:

```

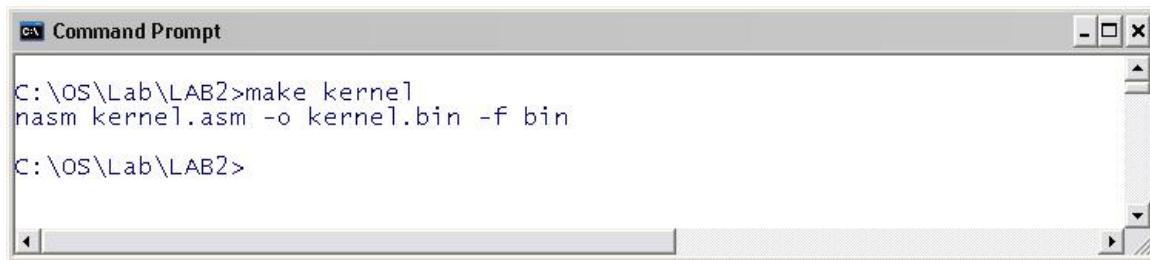
Booting from Floppy...

Belajar membuat BOOTSTRAP-LOADER
.....
ERROR : Press Any Key to Reboot

```

- (10) Menyiapkan file ‘KERNEL.BIN’: Prototype source code program kernel di simpan pada file ‘kernel.asm’, periksa keberadannya pada direktori kerja anda.

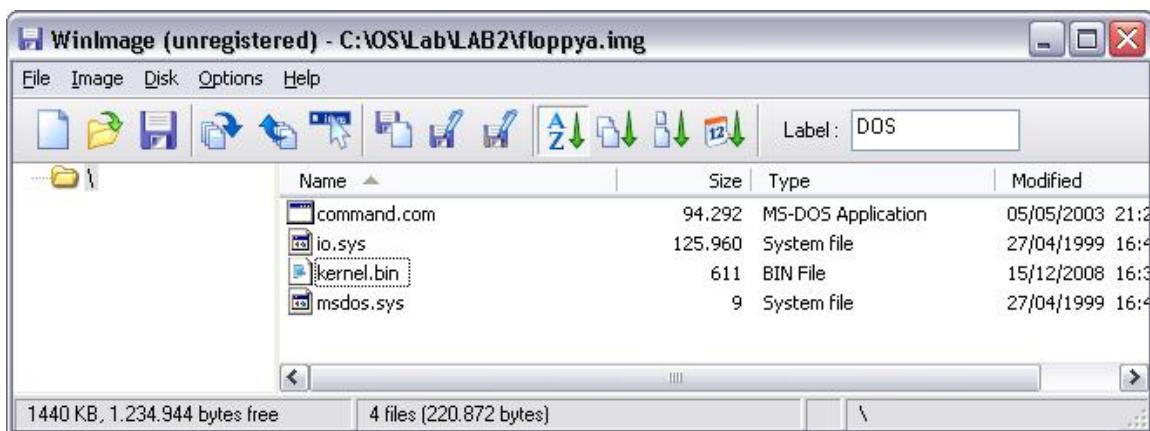
Selanjutnya lakukan proses kompilasi untuk menhasilkan file ‘KERNEL.BIN’, huruf besar atau kecil pada nama file tidak berpengaruh. Jalankan perintah berikut: ‘make kernel’ pada layar tampak seperti gambar berikut:



```
C:\OS\Lab\LAB2>make kernel
nasm kernel.asm -o kernel.bin -f bin
C:\OS\Lab\LAB2>
```

Jika tidak ada berita kesalahan, berarti proses kompilasi telah berhasil dan pada direktori kerja anda tredapat tambahan file baru yaitu ‘kernel.bin’, periksalah dengan perintah ‘dir’.

- (11) Memindahkan file ‘kernel.bin’ ke dalam file image ‘floppya.img’. Proses ini sedikit rumit, seperti proses kompresi sebuah file ke dalam file ‘zip’. Untuk melakukan hal ini kita gunakan program shareware namanya ‘winimage’. Bukanlah direktori kerja ‘C:\OS\LAB\LAB2’ dengan menggunakan ‘Windows Explorer’ kemudian dobel klik pada file ‘floppya.img’. atau panggil dari program winimage klik ‘Start>All Programs|Winimage|winimage’ selanjutnya klik menu ‘open’ dan cari file ‘floppya.img’ pada direktori kerja. Untuk memasukan file ‘kernel.bin’, pada window ‘WinImage’ klik menu ‘image|inject’ cari file ‘kernel.bin’ pada direktori kerja dan OK. Selanjutnya lakukan proses penyimpanan ‘File|Save’ dan keluar dari ‘WinImage’. Hasil akhir seperti di tampilkan pada gambar berikut:



- (12) Selanjutnya siap melakukan proses boot pada PC Simulator dengan menggunakan ‘floppya.img’ yang sudah diberi tambahan file ‘kernel.bin’. Kembali ke Windows ‘Command Prompt’ direktori kerja, jalankan PC-Simulator dengan memasukan perintah ‘s’ <ENTER>. Jika tidak ada kesalahan pada proses sebelumnya maka pada layar PC Simulator akan ditampilkan teks seperti berikut ini:

```
Booting from Floppy...

Belajar membuat BOOTSTRAP-LOADER
.....
.....
...
Welcome to MY KERNEL Ver 0.01
MY KERNEL$>
```

- (13) Memodifikasi file ‘kernel.asm’ : pada bagian berikut ini kita akan mencoba melakukan modifikasi kecil pada file ‘kernel.asm’. Jalankan perintah berikut ‘Notepad kernel.asm’ <ENTER>. Cari teks ‘Welcome to MY KERNEL ...’ dari windows ‘notepad’, tekan tombol ‘CTRL+F’ ketikan ‘Welcom to MY KERNEL’ dan tekan <ENTER> dan klik ‘CANCLE’. Teks tersebut disimpan pada variabel ‘strWelcomeMsg’, gantikan isinya dengan ‘Belajar membuat KERNEL’:

```
strWelcomeMsg    db    "Belajar membuat KERNEL", 0x00
```

Simpan file ‘kernel.asm’, tekan ‘CTRL+S’, tutup windows ‘Notepad’ dan kembali ke ‘Command Prompt’ direktori kerja.

Ulangi proses kompilasi, jalankan perintah ‘make kernel’<ENTER>, selanjutnya ulangi proses yang dilakukanp ada nomor 11 (memindahkan file ‘kernel.bin’ ke dalam file ‘floppya.img’ dengan ‘WinImage’).

Terakhir jalankan PC Simulator, ‘s’<ENTER>. Jika langkah yang anda lakukan tidak ada kesalahan maka pada layar PC Simulator akan di tampilkan teks seperti berikut:

```
Booting from Floppy...
```

```
Belajar membuat BOOTSTRAP-LOADER
.....
.....
..
Belajar membuat KERNEL
MY KERNEL$>
```

- (14) Proses diatas merupakan proses pengembangan sistem operasi yang paling dasar, setiap sistem operasi, seperti windows, linux, unix, mac dan lain-lain, akan melakukan tahapan proses tersebut. Anda juga dapat mengembangkan ‘embedded system’ dengan cara yang serupa.

Untuk memastikan apakah ‘floppya.img’ benar-benar dapat digunakan pada PC yang sebenarnya, anda dapat memindah isi ‘floppya.img’ ke dalam floppy sebenarnya dengan program ‘WinImage’ (sudah tentu pada PC anda harus ada FDD Drive), jalankan program ‘WinImage’ buka file ‘floppya.img’ dan klik menu ‘disk|Write disk’. Selanjutnya lakukan proses ‘BOOT’ pada PC anda dengan floppy tersebut, jika pada langkah anda tidak ada kesalahan maka pada PC akan ditampilkan teks seperti pada PC Simulator, silahkan mencoba?

Pada modul berikutnya akan kita akan belajar cara menjelajah tahapan proses (*trace* atau *debugging*) program ‘bootstraploader’ dan ‘kernel’.

Tugas

- (1) Pelajari cara kerja program ‘boot.asm’ buatlah algoritma dari program tersebut dalam bentuk flowchart. Untuk memudahkan dalam memahami proses boot buatlah dua jenis algoritma, pertama buat algoritma yang bersifat global dan kedua buat algoritma yang bersifat lebih detail.
- (2) Lakukan hal yang sama untuk program ‘kernel.asm’.

Hint: Gunakan buku referensi atau akses internet sebagai panduan

Modul 3 : Mengenal cara ‘Debugging’ program bootstrap-loader

Tujuan

Memahami cara melakukan proses ‘debugging’ program bootstrap-loader dan kernel dengan menggunakan program PC-Simultor ‘Bochs’.

Pendahuluan

Pada tahapan pengembangan program selalu dihadapi munculnya kesalahan pada program, terdapat dua jenis kesalahan yang biasa terjadi pada proses program, yaitu kesalahan syntax dan kesalahan logika.

Kesalahan syntax atau bentuk adalah kesalahan yang terjadi pada source code yang dapat mengakibatkan gagalnya proses kompilasi. Hal ini disebabkan oleh adanya beberapa struktur penulisan dan/atau istilah yang tidak di mengerti oleh kompiler (penerjemah bahasa pemrograman dari bahasa tingkat tinggi menjadi bahasa mesin). Kesalahan ini juga dapat disebut sebagai kesalahan penulisan bahasa pemrogramman. Kesalahan jenis ini sangat mudah di betulkan karena terlihat jelas. Biasanya kompiler yang baik akan menunjukkan letak kesalahannya dengan memberikan informasi mengenai nomor baris pada source code yang menyebabkan terjadinya kesalahan.

Jenis kesalahan kedua adalah kesalahan logika, kesalahan ini lebih sulit dideteksi, karena secara syntak sudah betul namun setelah program dijalankan, hasil akhirnya tidak sesuai dengan apa yang diharapkan oleh programmer, terdapat kesalahan alur pikir atau logika di dalam program. Cara paling mudah untuk mendeteksi jenis kesalahan ini adalah dengan menggunakan fasilitas pembantu yang disebut ‘debugger’, hampir setiap bahasa pemrogramman menyediakan fasilitas debugger. Namun dalam pengembangan sistem operasi ini kita tidak dapat menggunakan fasilitas debugger yang disediakan oleh kompiler. Karena PC harus di reset untuk menjalankan program bootstrap-loader, dan jika PC di reset semua yang tersimpan dalam RAM juga akan di reset. Diperlukan cara tersendiri untuk melakukan debugging pada program bootstrap-loader dan kernel.

Salah satu cara melakukan debugging pada sistem ini adalah dengan menggunakan PC-Simulator yang menyediakan fasilitas debugging seperti ‘Bochs’. Pada modul ini akan dipelajari cara melakukan proses ‘debugging’ program bootstrap loader dengan PC-

Simulator. Untuk melakukan proses ini sebaiknya anda sudah memahami alur logika atau cara kerja program ‘boot.asm’ dan ‘kernel.asm’.

Prototipe program ‘boot.asm’ oleh BIOS akan di letakkan pada alamat ‘0000:07C00’, selanjutnya BIOS akan menyerahkan tugasnya pada program ‘boot.asm’, dimulai dengan mengatur register segmen, kemudian mengidentifikasi format disk boot untuk menentukan lokasi sektor yang digunakan untuk menyimpan data root direktori (daftar nama direktori dan nama file). Selanjutnya mencari nama file yang akan digunakan sebagai kernel (pada modul ini namanya ‘KERNEL.BIN’), dan memindahkan isi file kernel pada lokasi memori ‘0100:0000’ kemudian menjalankan program kernel mulai dari alamat tersebut.

Prototipe program ‘kernel.asm’ relatif lebih sederhana yaitu menjalankan program ‘shell’ sangat sederhana, menunggu perintah yang dimasukan user dari ‘keyboard’, jika ada perintah yang dimasukan user, selanjutnya shell mengidentifikasi perintah dan menjalankan operasi sesuai dengan perintah user. Saat ini hanya perintah ‘ver’ dan ‘exit’ yang dimengerti oleh ‘shell’ yang dijalankan dari ‘kernel.bin’.

Peralatan

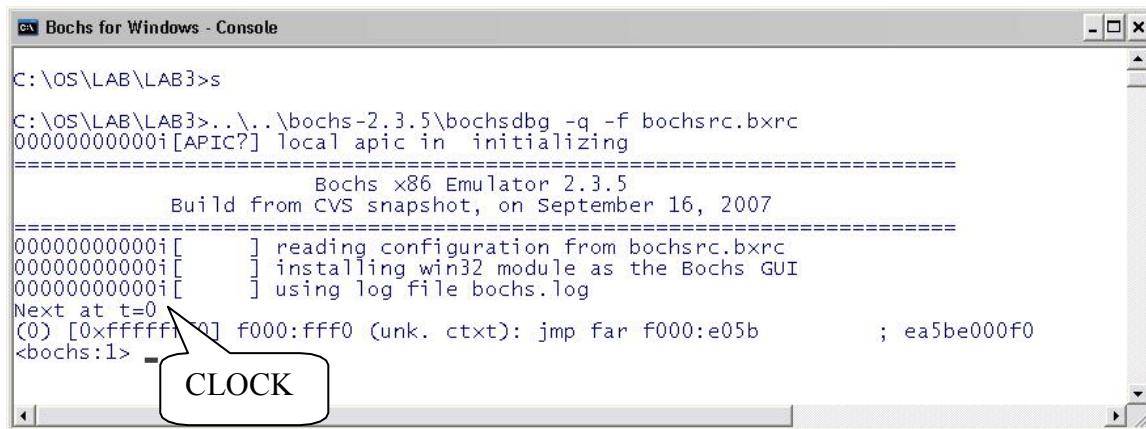
Sebuah PC dengan sistem operasi Windows Xp yang didalamnya terdapat program PC-Simulator ‘Bochs’, kompiler bahasa assembly ‘nasm’, kompiler bahasa C dari ‘Dev-Cpp’ (gcc versi windows) dan beberapa program bantu seperti ‘make’, ‘debug’, ‘dd’, dan ‘tdump’. Sebagai ‘editor-text’ untuk memodifikasi program digunakan program ‘Notepad’ yang terdapat pada sistem operasi Windows Xp.

Direktori kerja ‘C:\OS\LAB\LAB3’

Langkah Kerja

- (1) ‘Start|run’ ketik ‘cmd’ lanjutkan dengan ‘CD OS’, ‘setpath’ dan ‘cd LAB/LAB3’.
- (2) File sudah di siapkan, tugas anda adalah melakukan proses ‘debugging’. Program ‘Bochs’ yang diaktifkan adalah program versi debug yaitu ‘Bochsdbg’, lihatlah pada perintah yang tersimpan pada file ‘s.bat’, ketika ‘type s.bat’.
- (3) Mulai melakukan ‘debugging’: masukan perintah ‘s’ <ENTER>. Layar pada PC-Simulator akan terlihat gelap, tidak ada aktifitas, tidak ada kesalahan disana tetapi

jalannya program dihentikan oleh ‘Bochs’ menunggu masukan dari user. Pindahlah ke window ‘Command Prompt’ yang sekarang muncul tampilan seperti pada gambar berikut:



```
C:\OS\LAB\LAB>>s
C:\OS\LAB\LAB>..\..\bochs-2.3.5\bochsdbg -q -f bochssrc.bxrc
00000000000i[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
00000000000i[      ] reading configuration from bochssrc.bxrc
00000000000i[      ] installing win32 module as the Bochs GUI
00000000000i[      ] using log file bochs.log
Next at t=0
(0) [0xfffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be000f0
<bochs:1>
|-----|
|-----| CLOCK
|-----|
```

- (4) Kondisi pada gambar di atas menjelaskan kondisi PC pada mode ‘Real-Mode’ yang sedang akan menjalankan program yang pertama kali (0), yaitu program yang terdapat pada alamat ‘F000:FFF0’, lokasi alamat tersebut pada PC merupakan alamat paling atas yang dapat di akses oleh PC dan biasanya digunakan untuk menyimpan alamat ROM BIOS. Selanjutnya PC akan memanggil program yang terdapat pada BIOS. Pertama kali yang dilakukan BIOS adalah loncat ke alamat yang lebih rendah yaitu alamat f000:e05b terlihat pada layar sebagai baris perintah ‘jmp far f000:e05b’ kode bahasa mesinnya adalah ‘ea5be000f0’ akan lebih mudah dimengerti jika dipisahkan dalam bentuk seperti ini ‘ea 5be0 00f0’, Kode ‘ea’ : adalah kode untuk perintah ‘jmp far’ sedangkan data 8 byte berikutnya merupakan lokasi alamat yang ditunjuk, 4 byte paling akhir adalah menunjukkan alamat segmen yang dituju dengan susunan dibalik ‘00f0’ menjadi ‘f000’. Sedangkan 4 byte di tengah adalah alamat penunjuk (IP), susunannya juga dibalik ‘5be0’ menjadi ‘e05b’ inilah aturan penyimpanan data pada PC IBM compatible. Lokasi program yang akan dijalankan oleh PC pada mode kerja ‘Real-Mode’ adalah lokasi yang ditunjuk oleh kombinasi register CS:IP. Anda dapat melihat isi register CS dan IP dengan perintah ‘r’. Sekarang ketikan ‘r’ <ENTER> akan ditampilkan teks berikut :

```

Bochs for Windows - Console
000000000000i[      ] reading configuration from bochssrc.bxrc
000000000000i[      ] installing win32 module as the Bochs GUI
000000000000i[      ] using log file bochs.log
Next at t=0
(C0) [0xfffffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b ; ea5be000f0
<bochs:1> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000ffff
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2>

```

- (5) Selanjutnya kita suruh PC untuk mengeksekusi perintah tersebut, ketikan ‘s’<ENTER> kemudian lanjutkan dengan perintah ‘r’ <ENTER>. Pada layar akan ditampilkan terks berikut:

```

Bochs for Windows - Console
rip: 0x00000000:0000ffff CLOCK pertama (1) pf cf
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2> s
Next at t=1
(C0) [0x000fe05b] f000:e05b (unk. ctxt): xor ax, ax ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000e05b
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:4>

```

- (6) Jika anda ingin tahu tahapan detail yang dilakukan oleh PC, jalankan dengan perintah ‘s’<ENTER> secara berulang. Tetapi dengan cara ini, untuk sampai pada proses bootstrap mungkin tidak akan selesai dalam satu hari. Cara yang efektif adalah kita biarkan PC melanjutkan pekerjaannya, kemudian kita suruh berhenti saat PC mulai memasuki tahapan proses BOOT, yang dimulai pada alamat 0000:7C00. Hal ini dapat dilakukan dengan cara memberikan sebuah sinyal berhenti yang disebut ‘break point’.

Masukan perintah berikut ‘vb 0:0x7C00’ <ENTER>

Maksud perintah ini adalah membuat titik pemberhentian (halte) pada alamat 0000:7C00. Selanjutnya kita perintahkan PC untuk melanjutkan pekerjaannya

sekarang, yaitu melanjutkan program yang terdapat pada BIOS untuk memeriksa RAM dan peralatan lainnya.

Masukkan perintah ‘c’<ENTER>

Maksud perintah ini adalah teruskan (Continue) prosesnya sampai ke titik pemberhentian. Dalam sekejap PC sudah sampai pada pemberhentian yang kita buat di atas yaitu pada alamat 0000:7C00, pada layar akan tampak teks seperti berikut.

```

Bochs for Windows - Console
(C0) [0x0000fe05b] f000:e05b (unk. ctxt): xor ax, ax ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000e05b
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt df if tf sf zf af pf cf
<bochs:4> vb 0:0x7C00
<bochs:5> c
(2089918400) Breakpoint 51119118, in 0000:7c00 (0x00007c00)
Next at t=2082128
(C0) [0x00007c00] 0000:7c00 (unk. ctxt): jmp .+0x003b (0x00007c3e) ; e93b00
<bochs:6>

```

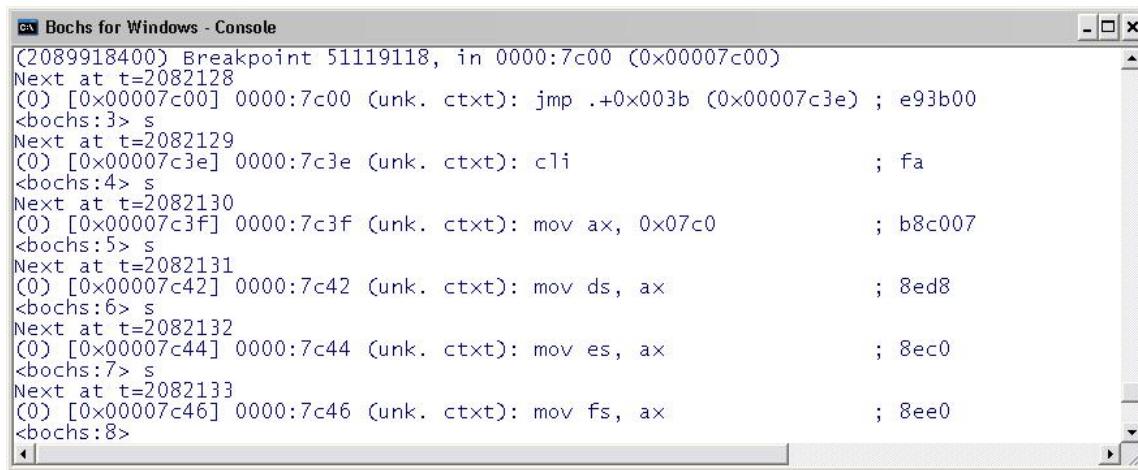
Jumlah CLOCK
dihitung sejak
POWER ON

Sedangkan pada PC Simulator mulai ditampilkan teks yang berakhir dengan tulisan berikut (periksalah):

Booting from Floppy...

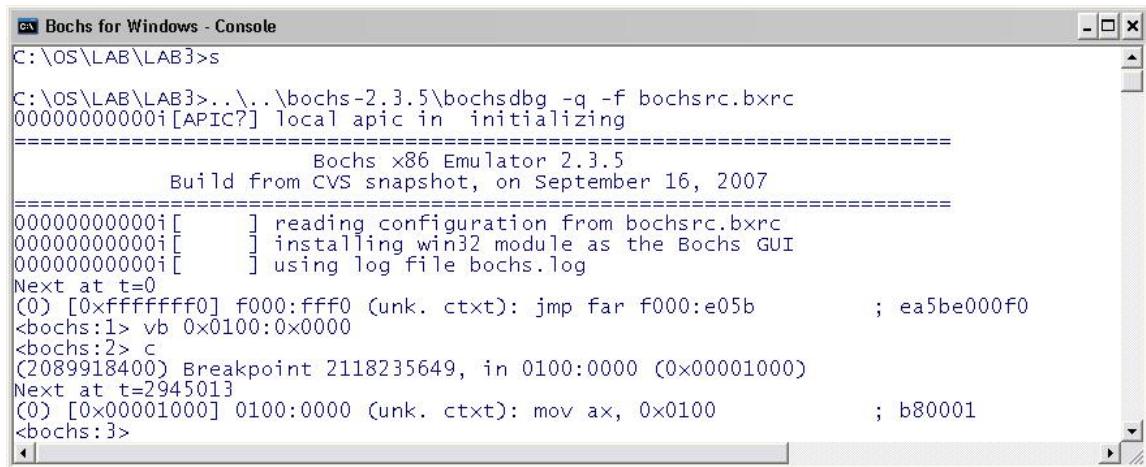
- (7) Sekarang PC mulai memasuki tahapan ‘BOOTSTRAP-LOADER’, untuk sampai pada tahap ini PC sudah menghabiskan clock sebanyak ‘2082128’ clock (lihat gambar di atas). Termasuk langkah memindahkan data bootsector ke dalam memori pada alamat 0000:7C00. Program yang akan dieksekusi selanjutnya adalah program yang berasal dari BOOTSECTOR pada diskboot. Jika digunakan ‘floppya.img’ sebagai disk-boot maka sekarang PC akan mulai menjalankan program ‘boot.asm’. Selanjutnya bandingkan 10 instruksi berikutnya yang akan dieksekusi oleh PC dengan program yang terdapat pada ‘boot.asm’, lakukan dengan cara menjalankan PC langkah demi langkah (debugging) dengan perintah ‘s’ <ENTER>, tulislah setiap teks yang ditampilkan pada setiap langkah. Perhatikan semua perubahan yang terjadi. Jika terjadi kesalahan, anda dapat menghentikan proses dengan perintah ‘q’<ENTER> dan mengulangi proses dari awal atau klik tombol POWER pada PC Simulator. Untuk melihat daftar menu

yang disediakan dapat digunakan perintah ‘h’<ENTER>. Untuk memerintahkan PC agar melanjutkan prosesnya masukan ‘c’<ENTER>. Empat instruksi selanjutnya akan tampak seperti pada gambar berikut:



```
(2089918400) Breakpoint 51119118, in 0000:7c00 (0x00007c00)
Next at t=2082128
(C) [0x00007c00] 0000:7c00 (unk. ctxt): jmp .+0x003b (0x00007c3e) ; e93b00
<bochs:3> s
Next at t=2082129
(C) [0x00007c3e] 0000:7c3e (unk. ctxt): cli ; fa
<bochs:4> s
Next at t=2082130
(C) [0x00007c3f] 0000:7c3f (unk. ctxt): mov ax, 0x07c0 ; b8c007
<bochs:5> s
Next at t=2082131
(C) [0x00007c42] 0000:7c42 (unk. ctxt): mov ds, ax ; 8ed8
<bochs:6> s
Next at t=2082132
(C) [0x00007c44] 0000:7c44 (unk. ctxt): mov es, ax ; 8ec0
<bochs:7> s
Next at t=2082133
(C) [0x00007c46] 0000:7c46 (unk. ctxt): mov fs, ax ; 8ee0
<bochs:8>
```

- (8) Selanjutnya anda dapat memerintahkan PC Simulator untuk melanjutkan pekerjaannya. Anda juga dapat menambahkan ‘break-point’ yang lain (maksimal 7). Tahapan penting berikutnya adalah ketika PC Simulator mulai menjalankan program ‘kernel.bin’, hal ini terjadi pada alamat ‘0100:0000’.
- (9) Menghentikan PC Simulator pada saat akan menjalankan program ‘kernel.bin’: Mulailah dari awal, hentikan ‘debugging’ sebelumnya dengan memasukan perintah ‘q’<ENTER>, jika PC Simulator sedang bekerja, hentikan dengan menekan tombol ‘CTRL+C’ kemudian ‘q’<ENTER>. Selanjutnya mulai dari awal, ketik ‘s’<ENTER> (dari ‘Command Prompt’). Kemudian buatlah breakpoint, masukan perintah ‘vb 0x0100:0x0000’ untuk mengehentikan langkah saat PC mulai mengeksekusi instruksi dari program ‘kernel.bin’. Selanjutnya perintahkan PC untuk melanjutkan perkerjaan, ‘c’<ENTER>. Jika langkah anda sesuai dengan petunjuk di atas maka pada layar ‘debugging’ akan di tampilkan teks berikut:



The screenshot shows a Windows console window titled 'Bochs for Windows - Console'. The command entered is 'C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsdbg -q -f bochsrc.bxrc'. The output shows the Bochs x86 Emulator version 2.3.5, build from CVS snapshot on September 16, 2007. It indicates configuration reading from 'bochsrc.bxrc', installing the win32 module as the GUI, and using log file 'bochs.log'. A breakpoint is set at address 0x000001000, and the assembly code at that location is shown:

```

000000000000i[      ] reading configuration from bochsrc.bxrc
000000000000i[      ] installing win32 module as the Bochs GUI
000000000000i[      ] using log file bochs.log
Next at t=0
(C0) [0xfffffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be000f0
<bochs:1> vb 0x0100:0x0000
<bochs:2> c
(2089918400) Breakpoint 2118235649, in 0100:0000 (0x00001000)
Next at t=2945013
(C0) [0x00001000] 0100:0000 (unk. ctxt): mov ax, 0x0100      ; b80001
<bochs:3>

```

- (10) Selanjutnya teruskan langkah PC Simulator step-by-step minimal sebanyak 10x, ketik ‘s’<ENTER>, step berikutnya dapat dilakukan dengan cara menekan tombol <ENTER> secara langsung. Perhatikan dan catat setiap perubahan teks yang ditampilkan. Bandingkan dengan source-code pada program ‘kernel.asm’.
- (11) Langkah-langkah di atas adalah merupakan salah satu cara yang banyak digunakan para pengembang perangkat lunak, termasuk pengembangan sistem operasi. Jika anda tertarik sebaiknya anda membiasakan dengan kegiatan di atas.

Tugas :

- (a) Buatlah tabel pemetaan memori pada PC selengkap mungkin.
- (b) Baca buku referensi, jelaskan perbedaan antara mode kerja ‘Real-Mode’ dan mode kerja ‘Protect-Mode’ pada PC IBM Compatible.

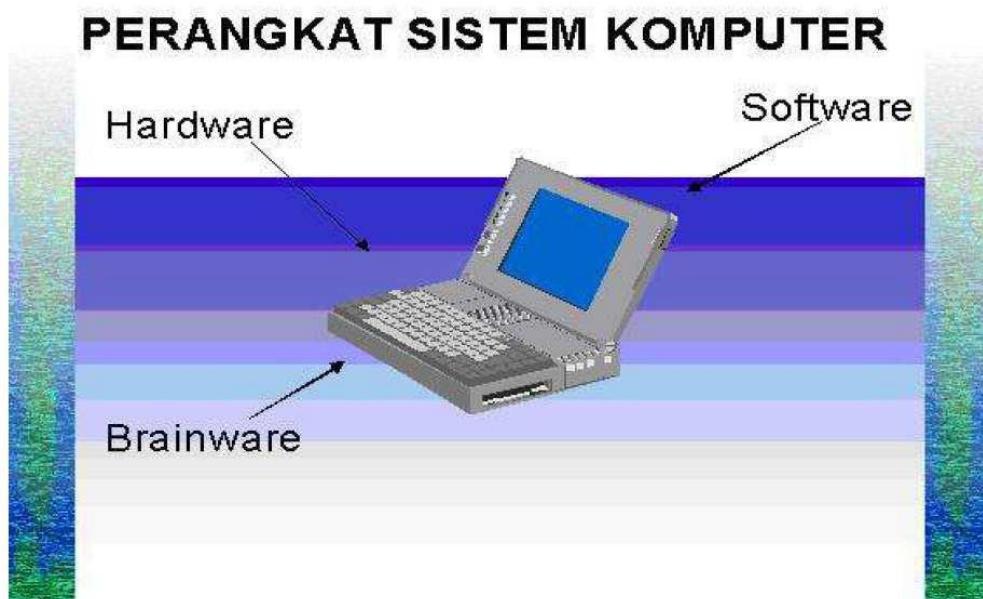
Modul 4 : Pengolahan File dan Directory Menggunakan MS-DOS / Command Prompt Di Windows

Tujuan

Mengenal perintah-perintah pengolahan file dan directory menggunakan command prompt pada windows xp

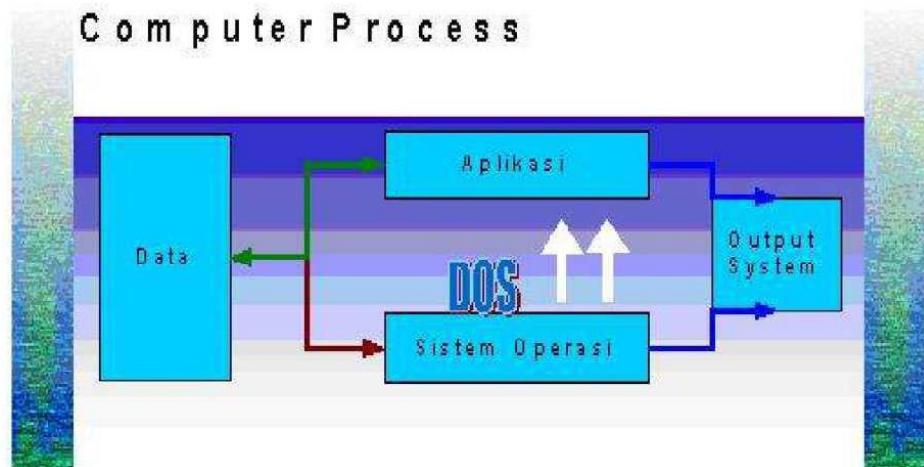
Pendahuluan

MS-DOS adalah singkatan dari Microsoft Disk Operating System, yaitu Sistem Operasi berbasis baris-perintah (command-line) yang digunakan pada PC. Seperti sistem operasi lain contohnya OS/2, ia menterjemahkan input dari keyboard menjadi pekerjaan yang dapat dilakukan oleh komputer, ia juga dapat menangani pekerjaan seperti input dan output pada disket atau harddisk, dukungan video, kontrol keyboard , dan banyak lagi fungsi-fungsi internal lainnya yang berkaitan dengan eksekusi sebuah program dan pemeliharaan file.



Gambar 4.1 perangkat sistem komputer

DOS menempati posisi sebagai operating system yang menggunakan CUI(Character User Interface). Dalam hal ini DOS yang saat ini telah tergantikan/diperbarui dengan adanya Microsoft Windows versi 9x, 2k, dan sebagainya yang berbasiskan GUI (Grafical user Interface).



Gambar 4.2 proses komputer

Beberapa Fungsi dari Operating system (DOS) adalah:

1. Mengorganisasikan atau mengendalikan kegiatan computer
2. Mengatur Memori
3. Mengatur proses input dan output data
4. Menegement file
5. Management directory

Perintah MS-DOS diketikkan dalam sebuah jendela yang disebut Command Prompt Window. Untuk keluar dari MS-DOS, ketik exit dalam jendela tersebut yaitu pada kursor yang berkedipkedip. MS-DOS Mode adalah sebuah shell dimana lingkungan MS-DOS di-emulasikan dalam Sistem Operasi 32-bit, seperti Windows. Program berbasis MS-DOS dapat berjalan di Windows dan biasanya ia membuat sebuah file yang disebut Program Information File (PIF) yang muncul sebagai shortcut di desktop anda.

Dalam pengoperasian DOS terdapat Command-command/perintah yang dikelompokkan dalam 2 kelompok yaitu:

1. Internal Command

Adalah perintah yang tidak lagi membutuhkan file khusus,karena semua instruksi internal sudah ditampung dalam file command.com

2. External Command

Untuk mempermudah mampelajari fasilitas DOS maka tiap perintah sudah terdapat file

Help Untuk menjalankannya bisa digunakan perintah seperti contoh berikut:

A:>Copy/?

Atau

A:>help copy

Perintah-Perintah Internal MS-DOS / Command Prompt dalam pengolahan file dan directory

Berikut ini adalah daftar perintah-perintah command prompt yang ada pada MS-DOS mode di Windows XP. Untuk informasi lebih spesifik mengenai suatu perintah, ketik HELP nama perintah di jendela Command Prompt.

ASSOC

Menampilkan atau mengubah asosiasi ekstensi file.

CD / CHDIR

Menampilkan nama atau mengubah direktori sekarang.

CLS

Menghapus layar jendela command prompt.

CMD

Menjalankan interpreter perintah Windows yang baru.

COPY

Menyalin satu atau beberapa file ke lokasi lain.

DEL / ERASE

Menghapus satu atau beberapa file.

DIR

Menampilkan daftar file dan subdirektori dalam sebuah direktori.

EXIT

Keluar dari program CMD.EXE dan menutup jendela command prompt.

HELP

Menyediakan informasi bantuan untuk perintah-perintah Windows.

MD / MKDIR

Membuat direktori.

MOVE

Memindahkan satu atau beberapa file dari satu direktori ke direktori yang lain.

PROMPT

Mengubah command prompt Windows.

RD /RMDIR

Menghapus direktori.

REN / RENAME

Mengubah nama file.

REPLACE

Mengganti file.

TREE

Menampilkan secara grafis struktur direktori dari sebuah drive atau path.

TYPE

Menampilkan isi dari sebuah file teks.

VER

Menampilkan versi Windows yang anda gunakan.

XCOPY

Menyalin file serta pohon direktori.

Untuk membuka jendela Command Prompt, klik Start, pilih All Programs, pilih Accessories, dan kemudian klik Command Prompt

Peralatan

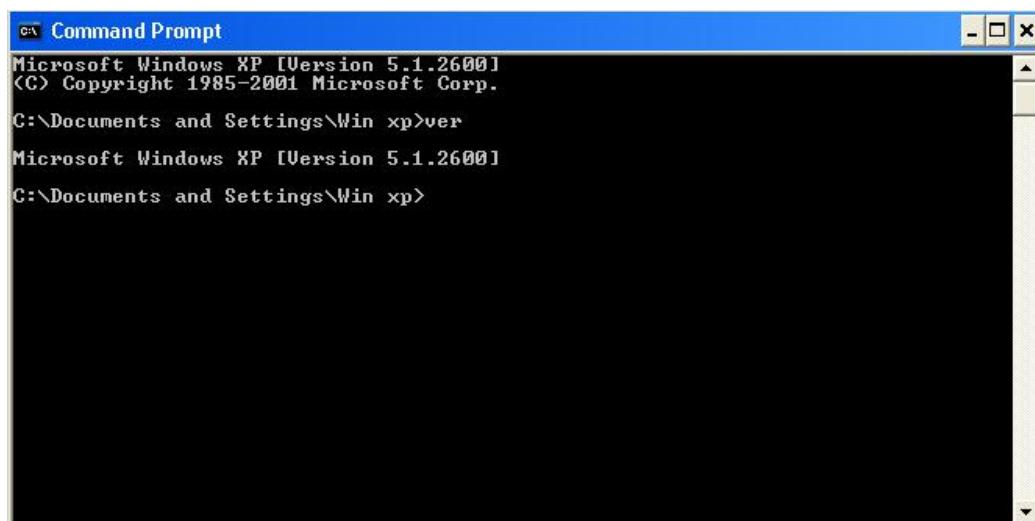
Sebuah PC dengan sistem operasi windows xp

Modul sistem operasi

Langkah kerja

Percobaan pertama : Melihat versi MS-Dos (ver)

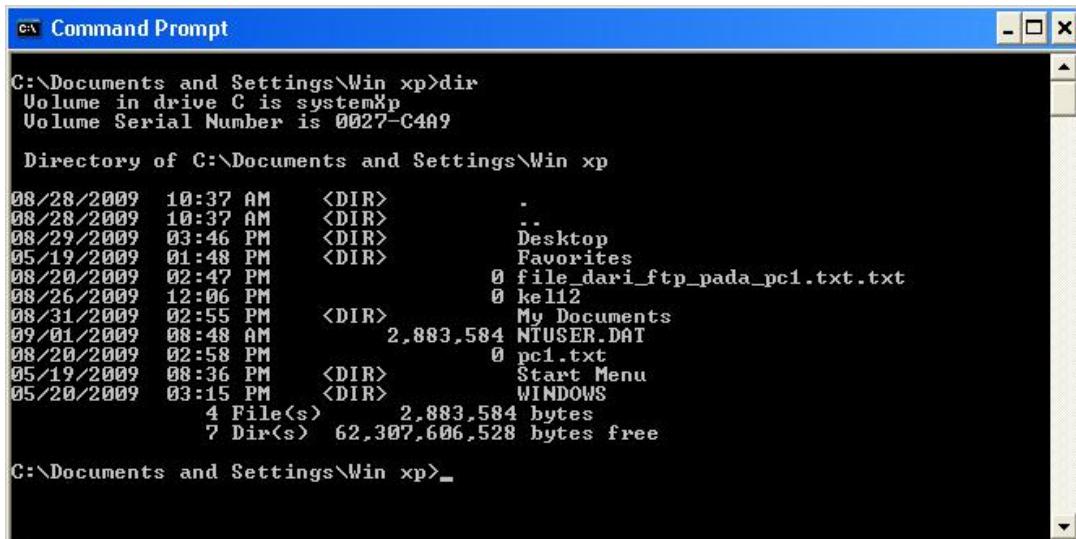
1. klik start - all programs - accessories - command prompt sehingga muncul command prompt
2. Ketik perintah `ver` untuk melihat versi command prompt yang kita gunakan



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Win xp>ver
Microsoft Windows XP [Version 5.1.2600]
C:\Documents and Settings\Win xp>
```

Percobaan kedua : melihat, masuk, keluar dari directory dan keluar dari command prompt (dir, cd, cd.., exit)

3. ketik perintah `cls` untuk membersihkan tulisan pada layar
4. ketik perintah `dir` untuk melihat isi dari directory yang ada pada komputer kita



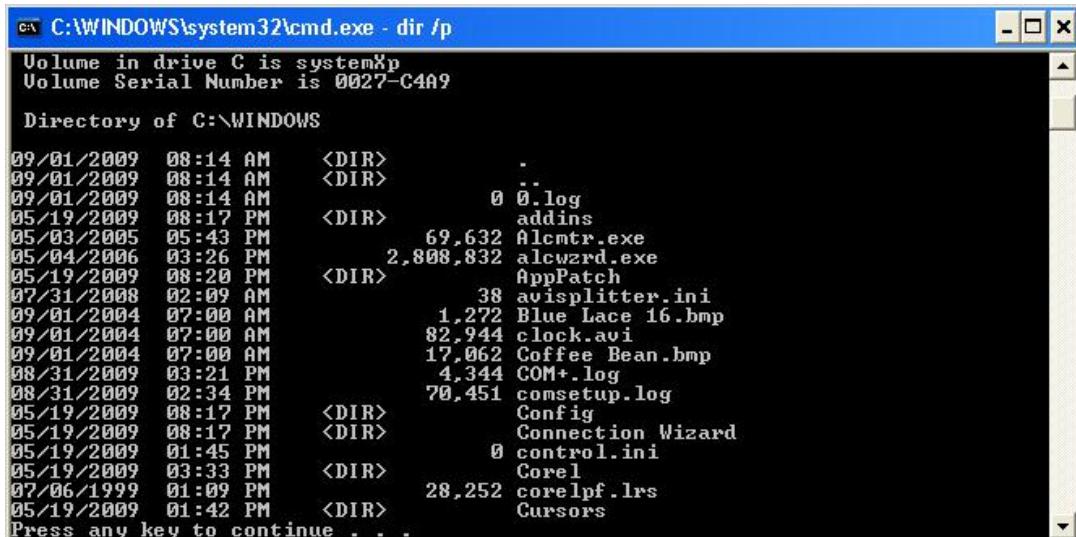
```
C:\Documents and Settings\Win xp>dir
Volume in drive C is systemXp
Volume Serial Number is 0027-C4A9

Directory of C:\Documents and Settings\Win xp

08/28/2009  10:37 AM    <DIR>        .
08/28/2009  10:37 AM    <DIR>        ..
08/29/2009  03:46 PM    <DIR>        Desktop
05/19/2009  01:48 PM    <DIR>        Favorites
08/20/2009  02:47 PM          0 file_dari_ftp_pada_pc1.txt.txt
08/26/2009  12:06 PM          0 kel12
08/31/2009  02:55 PM    <DIR>        My Documents
09/01/2009  08:48 AM          2,883,584 NTUSER.DAT
08/20/2009  02:58 PM          0 pc1.txt
05/19/2009  08:36 PM    <DIR>        Start Menu
05/20/2009  03:15 PM    <DIR>        WINDOWS
               4 File(s)      2,883,584 bytes
               7 Dir(s)   62,307,606,528 bytes free

C:\Documents and Settings\Win xp>
```

5. Ketik perintah `cd..` untuk keluar dari directory Win xp dan ketik perintah `cd..` untuk keluar dari directory document and settings sehingga masuk pada drive C
6. Ketik perintah `dir` untuk melihat isi directory yang ada pada drive C. Kemudian ketik perintah `cd windows` untuk masuk pada directory windows.
7. ketik perintah `dir/p` untuk menampilkan nama file perhalaman, kemudian tekan spasi untuk menampilkan halaman berikutnya dan tekan spasi untuk halaman berikutnya lagi dan seterusnya sampai di akhir halaman sesuai dengan banyaknya file dalam directory tersebut.



```
C:\WINDOWS\system32\cmd.exe - dir /p
Volume in drive C is systemXp
Volume Serial Number is 0027-C4A9

Directory of C:\WINDOWS

09/01/2009  08:14 AM    <DIR>        .
09/01/2009  08:14 AM    <DIR>        ..
09/01/2009  08:14 AM          0 0.log
05/19/2009  08:17 PM    <DIR>        addins
05/03/2005  05:43 PM          69,632 Alcmtr.exe
05/04/2006  03:26 PM          2,808,832 alcwzrd.exe
05/19/2009  08:20 PM    <DIR>        AppPatch
07/31/2008  02:09 AM          38 avisplitter.ini
09/01/2004  07:00 AM          1,272 Blue Lace 16.bmp
09/01/2004  07:00 AM          82,944 clock.avi
09/01/2004  07:00 AM          17,062 Coffee Bean.bmp
08/31/2009  03:21 PM          4,344 COM+.log
08/31/2009  02:34 PM          70,451 comsetup.log
05/19/2009  08:17 PM    <DIR>        Config
05/19/2009  08:17 PM    <DIR>        Connection Wizard
05/19/2009  01:45 PM          0 control.ini
05/19/2009  03:33 PM    <DIR>        Corel
02/06/1999  01:09 PM          28,252 corelpf.lrs
05/19/2009  01:42 PM    <DIR>        Cursors

Press any key to continue . . .
```

8. Ketik perintah `dir/w` menampilkan file secara mendatar. Kemudian ketik perintah `dir/a` untuk menampilkan semua file terutama file yang terhidden.
9. ketik perintah `exit` untuk keluar dari command prompt

Percobaan ketiga : berubah nama file (ren, rename)

10. Buka windows explorer kemudian masuk ke drive C. Buat folder baru dan beri nama “files”. Masuk ke folder files kemudian arahkan kursor ke daerah yang kosong, klik kanan – new – microsoft word document. Beri nama dengan “file1”.
11. ulangi langkah 1 dan langkah 5 untuk masuk ke command prompt dan berada pada drive C.
12. ketik `dir` kemudian ketik `cd files` untuk masuk ke directory files. Ketik `dir` lagi untuk melihat isi directory files.
13. ketik perintah `ren file1.doc file.doc` untuk mengubah nama file.
14. ketik perintah `rename file.doc file2.doc` untuk mengubah nama file.
15. ketik perintah `dir` untuk melihat directory files kemudian lihat perubahan file1.doc menjadi file.doc
16. Sekarang lihatlah lewat windows explorer dan buka folder files, pastikan bahwa nama file1.doc telah berubah menjadi file2.doc

Percobaan keempat : copy file dan menghapus file (copy, del, xcopy)

17. Ketik perintah `cd..` untuk keluar dari directory files.
18. buka windows explorer kemudian masuk ke drive C. buatlah folder baru dengan mengarahkan kursor ke daerah yang kosong kemudian klik kanan – new – folder, beri nama dengan “files2”.
19. Kembali ke command prompt.
20. ketik perintah `copy C:\files C:\files2` untuk mengcopy isi folder files ke folder files2.
21. buka windows explorer untuk memastikan isi folder files telah tercopy ke folder files2.
22. kembali ke command prompt
23. ketik perintah `cd files` untuk masuk ke directory files
24. ketik perintah `copy c:\files\file.doc d:\` untuk mengcopy file.doc ke drive D.
25. ketik perintah `D:` untuk pindah ke drive D kemudian ketik perintah `dir` untuk melihat isi drive D dan pastikan bahwa file.doc telah tercopy.

26. ketik `c:` untuk kembali ke directory awal kemudian ketik perintah `cd..` untuk masuk drive C.
27. ketik `del files` untuk menghapus file yang terdapat pada directory files kemudian tekan yes
28. ketik perintah `cd documents and settings` untuk masuk ke folder documents and settings
29. ketik perintah `cd win xp` untuk masuk ke folder win xp
30. ketik perintah `xcopy desktop D:\baru\ /s/e` untuk mengcopy isi file kemudian membuat folder baru dan meletakkan isi copyan ke folder baru yang terdapat pada directory files ke drive d. Kemudian tekan a

```
C:\Documents and Settings\Win xp>xcopy desktop d:\baru\ /s/e
Overwrite D:\baru\Adobe Photoshop CS.lnk <Yes/No/All>? a
desktop\Adobe Photoshop CS.lnk
desktop\numsUpdate.exe
desktop\CorelDRAW 12.lnk
desktop\DIGITALQURAN.lnk
desktop\linguist.lnk
desktop\Microsoft Office Excel 2003.lnk
desktop\Microsoft Office PowerPoint 2003.lnk
desktop\Microsoft Office Word 2003.lnk
desktop\IRC.lnk
desktop\Nero StartSmart.lnk
desktop\Remote Desktop Connection.lnk
desktop\Skype.lnk
desktop\xAMPP Control Panel.lnk
13 File(s) copied

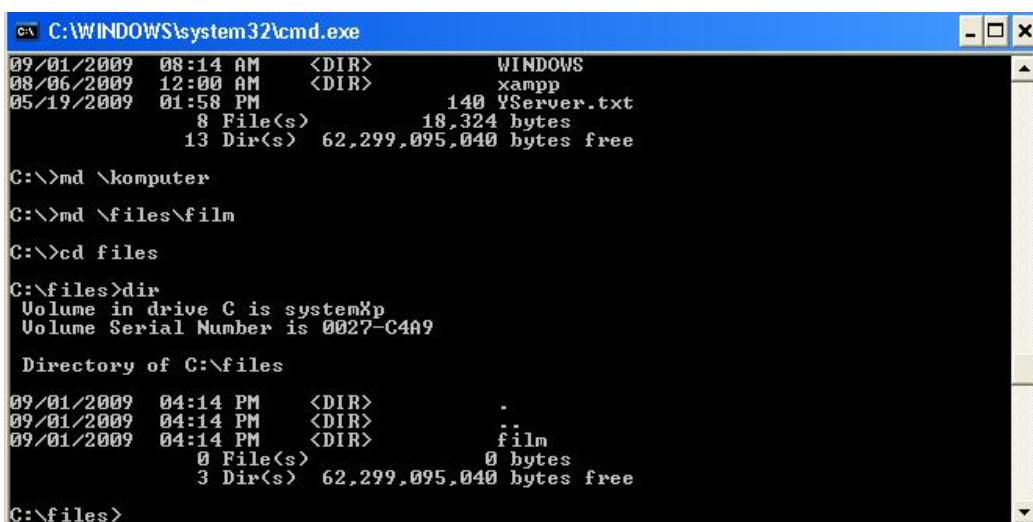
C:\Documents and Settings\Win xp>_
```

31. ketik perintah `d:` untuk masuk ke drive D
32. ketik perintah `dir` untuk melihat directory
33. ketik perintah `cd baru`
34. ketik perintah `dir` untuk melihat isi directory baru

Percobaan kelima : membuat dan menghapus directory (md, rd, mkdir, rmdir)

35. ketik perintah `c:` untuk kembali ke folder awal
36. ketik perintah `cd..` kemudian `cd..` lagi untuk berada pada drive c
37. ketik perintah `md baru` untuk membuat directory/folder yang bernama “baru”
38. ketik perintah `dir` untuk melihat isi directory files
39. ketik perintah `rd baru` untuk menghapus directory baru
40. ketik perintah `dir` untuk melihat directory files sekarang
41. ketik perintah `mkdir barul` untuk membuat directory/folder yang bernama “barul”

42. ketik perintah `dir` untuk melihat isi dari directory files
43. ketik perintah `rmdir baru1` untuk melihat isi directory files
44. ketik perintah `dir` untuk melihat isi dari directory files
45. ketik perintah `md \komputer` untuk membuat folder bernama komputer didalam drive C
46. ketik perintah `dir` untuk mengecek folder yang telah dibuat
47. ketik perintah `md \files\film` untuk membuat folder bernama film didalam folder files.
48. ketik `cd files` untuk masuk ke dalam directory files kemudian ketik perintah `dir` untuk melihat isi dari directory files.



```
C:\WINDOWS\system32\cmd.exe
09/01/2009 08:14 AM <DIR>      WINDOWS
08/06/2009 12:00 AM <DIR>      xampp
05/19/2009 01:58 PM           140 YServer.txt
                           8 File(s)   18,324 bytes
                           13 Dir(s)  62,299,095,040 bytes free

C:>>md \komputer
C:>>md \files\film
C:>>cd files
C:>dir
 Volume in drive C is systemXp
 Volume Serial Number is 0027-C4A9

 Directory of C:\files

09/01/2009 04:14 PM <DIR>      .
09/01/2009 04:14 PM <DIR>      .
09/01/2009 04:14 PM <DIR>      film
                           0 File(s)   0 bytes
                           3 Dir(s)  62,299,095,040 bytes free

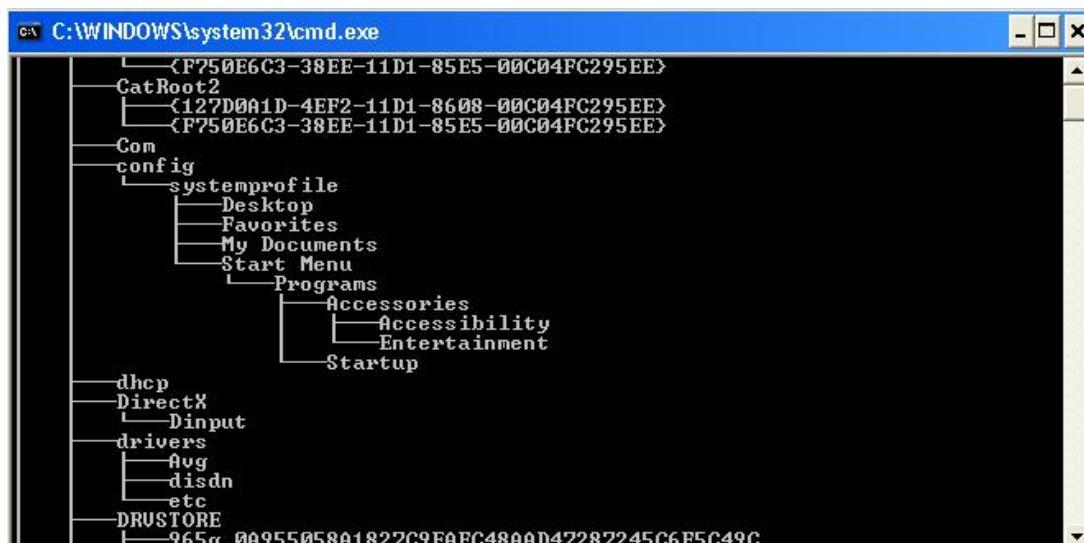
C:\files>
```

49. ketik perintah `exit` untuk keluar dari command prompt
50. buka windows explorer untuk melihat hasilnya.

Percobaan keenam : memindahkan file dan menampilkan isi folder secara terstruktur (move, tree)

51. Ulangi langkah 1 dan 5 sampai berapa pada drive c.
52. buka windows explorer kemudian masuk ke `C:\files\baru`. Arahkan kursor pada daerah yang kosong kemudian klik kanan - new - micorosoft word document. Beri nama dengan “pindah.doc”
53. Arahkan kursor pada daerah yang kosong lagi kemudian klik kanan - new - text document. Beri nama dengan “pindah.txt”
54. kembali ke command prompt
55. ketik perintah `move c:\files\baru\pindah.doc c:\files2` untuk memindah file pindah.doc ke folder files2

56. ketik perintah `move c:\files\baru\pindah.txt c:\files2` untuk memindah file pindah.txt ke folder files2
57. ketik perintah `cd files2` untuk masuk ke directory files2
58. ketik perintah `dir` untuk melihat isi dari directory
59. sekarang buka windows explorer untuk melihat hasilnya
60. kembali ke command prompt
61. ketik perintah `cd..` untuk keluar dari directory files2
62. ketik `tree c:\files` untuk melihat isi folder files secara terstruktur
63. ketik `tree c:\windows` untuk melihat seluruh isi folder windows secara terstruktur



```
C:\WINDOWS\system32\cmd.exe
└── CF750E6C3-38EE-11D1-85E5-00C04FC295EE
    ├── CatRoot2
    │   └── {127D0A1D-4EF2-11D1-8608-00C04FC295EE}
    │       └── CF750E6C3-38EE-11D1-85E5-00C04FC295EE
    ├── Com
    ├── config
    │   └── systemprofile
    │       ├── Desktop
    │       ├── Favorites
    │       ├── My Documents
    │       ├── Start Menu
    │       └── Programs
    │           ├── Accessories
    │           │   ├── Accessibility
    │           │   └── Entertainment
    │           └── Startup
    ├── dhcp
    ├── DirectX
    │   └── Dinput
    ├── drivers
    │   ├── Avg
    │   └── disdn
    ├── etc
    └── DRUSTORE
        └── 965g_0A955058A1827C9EAFC48AAD47287245C6F5C49C
```

Tugas :

- Buatlah sebuah directory / folder yang bernama multimedia di drive D
- Copykan 1 buah file musik (.mp3), gambar (.jpg), document (.doc), teks (.txt) ke dalam folder multimedia
- Tuliskan langkah-langkahnya ke dalam laporan praktikum

Modul 5 : Pengenalan Sistem Operasi Linux

Tujuan

1. Mengenal sistem operasi Linux
2. Memahami proses login/logout pada sistem operasi Linux
3. Memahami perintah-perintah dasar Linux
4. Memahami teknik mencari bantuan pada sistem operasi Linux.

Pendahuluan

Sebuah komputer dapat dioperasikan apabila dalam komputer tersebut terdapat sistem operasi yang kompatibel. Saat ini banyak sistem operasi yang dapat berjalan di PC seperti DOS, Windows, Linux, dan Unix.

Pada awal perkembangannya sekitar akhir 1960 sistem operasi masih belum memiliki arsitektur sistem yang standard, sampai pada awal tahun 1969 lahir sistem operasi yang bernama UNICS (Uniplexed Information and Computer System) yang dikembangkan oleh ken thompson seorang peneliti di BELL Lab. Ia membangun sistem operasi dengan bahasa pemrograman assembly pada komputer PDP7 yang disebut UNICS. model dari UNICS ini masih mengikuti pola pada sistem operasi UNIX yaitu pada model perintah pendek dari UNIX, ls, cp, rm, mv, dll.

Pada tahun 1973 Ken Thompson bersama Dennis Ritchie seorang pembuat C compiler, mengembangkan sistem operasi UNIX kernel dengan bahasa pemrograman C, dan pada tahun 1974 lahirlah sistem operasi UNIX generasi ke-5. pada tahun 1978 UNIX kembali dirilis untuk edisi yang ke-7. Pada generasi ke-7 ini UNIX pecah menjadi dua cabang yaitu SYSV(System 5) dan BSD(Berkeley Software Distribution). SYSV dikembangkan oleh BELL AT&T Lab sebagai paket sistem operasi UNIX komersial, sedangkan BSD dikembangkan oleh Ken Thompson bersama dengan para mahasiswa dalam sebuah penelitian di universitas Berkeley. Generasi baru dari SYSV adalah SYSVR4(System 5 Release 4). Pada dasar antara SYSV dan BSD memiliki kesamaan yaitu keduanya sama berasal dari satu rumpun UNIX namun dari keduanya terdapat beberapa sedikit perbedaan sperti pada tabel 1 berikut.

Feature	Typical SYSV	Typical BSD
kernel name	/unix	/vmunix
boot init	/etc/rc.d directories	/etc/rc.* files
mounted FS	/etc/mnttab	/etc/mtab
default shell	sh, ksh	csh, tcsh
FS block size	512 bytes->2K	4K->8K
print subsystem	lp, lpstat, cancel	lpr, lpq, lprm
echo command (no new line)	echo "\c"	echo -n
ps command	ps -fae	ps -aux
multiple wait syscalls	poll	select
memory access syscalls	memset, memcpy	bzero, bcopy

Linux merupakan kombinasi turunan dan SYSV dan BSD, pada tahun 1991 Linus B Torvalds, seorang mahasiswa di universitas Helsinki mengembangkan sebuah sistem operasi bernama Linux. Linux mengadopsi standard spesifikasi UNIX yaitu POSIX. (Portable Operating System Interface). Linux dirilis dalam bentuk open source, dimana sifat alamiah open source Linux adalah semua kode kernel Linux didistribusikan dan dapat ditambah atau dimodifikasi oleh siapa saja.

Arsitektur Sistem Operasi Linux

Seperti halnya semua sistem operasi pada umumnya Linux memiliki konsep sama dalam hal arsitektur. Perbedaanya adalah bentuk disetiap komponen pada arsitektur tersebut. Komponen pada arsitektur sistem operasi Linux adalah sebagai berikut :

1. Kernel

Kernel adalah bagian terpenting dari sebuah sistem operasi. Pada sistem operasi kernel berfungsi untuk mengendalikan semua perangkat keras yang ada di PC seperti CPU(Central Processing Unit), Graphic Cards, Sound Card, Perangkat USB, Harddisk, dll. Selain itu juga kernel berfungsi untuk menjembatani antara aplikasi yang berjalan dengan perangkat keras yang diakses oleh aplikasi tersebut. Sebagai contoh aplikasi text editor (vi,notepad,emacs) membutuhkan resources layar monitor sebagai tampilan bagi pengguna, dalam kasus ini maka aplikasi tersebut akan dihubungkan oleh kernel melalui device driver VGA Card untuk mengeluarkan data menuju layar monitor. Selain dari aplikasi kernel juga menjembatani perintah yang ditulis oleh pengguna melalui shell atau GUI. Kernel Linux merupakan turunan dari Unix BSD dan SYSV, dan model dari kernel Linux mengacu pada spesifikasi POSIX(Portable Operating System IX).

Sistem kerja dari kernel adalah ketika PC melakukan booting pertama kali maka file biner dari kernel akan dipindah menuju memory dan akan menjalankan semua fungsi manajemen resources pada PC tersebut. Setiap PC memiliki jenis perangkat keras yang berbeda-beda, sehingga kernel akan menginisialisasi perangkat tersebut dengan module device driver yang berbeda. File biner kernel linux dapat dilihat pada direktori “/boot/vmlinuz” sedangkan source kode dapat dilihat di “/usr/src/linux”. Sampai saat ini (Agustus 2009) kernel linux stable sudah keluar dengan versi 2.6.30.5. Untuk lebih detail mengenai perkembangan generasi kernel dapat dilihat pada situs <http://www.kernel.org/>.

2. Shells/GUI

Shell merupakan antarmuka linux kernel dengan penggunanya. Di dalam shell pengguna dapat menuliskan perintah yang kemudian akan dieksekusi langsung oleh kernel. Perintah-perintah yang dapat dituliskan merupakan perintah yang berhubungan dengan pengelolaan sistem operasi, misalnya melihat isi file yang tersimpan dalam sebuah direktori, membuat file baru, atau melihat isi dari sebuah file. Selain operasi file shell juga dapat digunakan untuk operasi I/O atau operasi penanganan pengguna.

Pada sistem operasi linux ada dua jenis shell atau penghubung antara pengguna dengan sistem operasi yang pertama adalah text-based shell dan yang kedua adalah GUI(Graphic User Interface). Beda dari keduanya adalah cara penyajiannya kepada pengguna, jenis text-based adalah antarmuka paling sederhana dimana pengguna hanya dapat berinteraksi melalui keyboard saja, sedangkan GUI memungkinkan pengguna berinteraksi dengan sistem operasi melalui keyboard saja, mouse dan perangkat lainnya dapat digunakan. Selain itu tampilan pada GUI sudah lebih modern yaitu berbasis tampilan grafis.

3. Utilitas sistem

Setiap sistem operasi pada dasarnya mengacu pada sebuah spesifikasi tertentu seperti halnya Unix pada POSIX.2 dimana pada spesifikasi ini dijelaskan konsep implementasi utilitas sistem untuk membantu pengguna berinteraksi dengan sistem operasi.

Utilitas sistem dalam sebuah sistem operasi berfungsi sebagai pengaturan sumber daya yang sudah diinisialisasi oleh sistem operasi pada saat proses booting awal dilakukan. Fungsi monitoring, fungsi pengaktifan dan penon-aktifan sebuah

service dalam sistem operasi merupakan pekerjaan yang dapat dilakukan dengan menggunakan utilitas sistem yang ada dalam sebuah sistem operasi seperti Linux. Beberapa contoh utilitas sistem dalam Linux adalah ls, man, fdisk, grep, awk, sed, cp, mv, more, dan lain sebagainya. Seperti halnya dalam sistem operasi Windows beberapa utilitas misalnya control panel, dir, cd, dan lain-lain.

Selain itu juga utilitas sistem juga termasuk program yang bersifat server, dimana program ini akan berjalan secara background (tidak tampak), seperti misalnya utilitas untuk menangani koneksi dari jarak jauh dengan telnet atau ssh, kedua layanan ini merupakan utilitas sistem yang bersifat server dan dijalankan secara background. contoh lain dari server program dalam Linux adalah lpd untuk mengangani proses printer, httpd sebagai web server, crond untuk menjalankan tugas administratif sistem operasi secara reguler. Utilitas sistem yang bersifat server dan dijalankan secara backgroung dalam Linux disebut DAEMON(Disk And Execution MONitor). daemon diaktifkan pada saat proses booting sistem operasi, dan akan berjalan secara terus menerus. Program ini akan memantau event yang terjadi dan apabila sesuai dengan yang dikehendaki maka daemon akan menjalankan sesuatu. Sebagai contoh program daemon telnet server. Daemon ini akan melakukan pemantauan pada semua koneksi yang datang dari luar PC melalui jaringan pada nomer port 23. jika ada koneksi dari luar yang menghubungi nomer tersebut maka telnet server akan menerima koneksi tersebut dan melayani proses yang akan dilakukan.

4. Aplikasi

Aplikasi merupakan level tertinggi pada sistem operasi. Dimana pada aplikasi ini pengguna sistem operasi dapat menggunakan aplikasi berdasarkan kebutuhannya. Misalnya aplikasi untuk menulis, pada sistem operasi Linux dapat dilakukan dengan menggunakan bantuan aplikasi Open Office, sedangkan untuk melakukan editing gambar dapat menggunakan GIMP. Karena aplikasi merupakan level tertinggi dari sebuah sistem operasi maka setiap aplikasi dibentuk dengan cara berbeda untuk masing-masing sistem operasi dan hanya bisa berjalan pada sistem operasi tempat dimana aplikasi tersebut dibangun. Misalnya MS Office tidak dapat berjalan pada sistem operasi Linux, kecuali pembangun MS Office mengeluarkan distribusi paket MS Office for Linux.

Distribusi Linux yang beredar saat ini, selain kernel juga menyertakan paket-paket aplikasi yang dapat dimanfaatkan oleh pengguna, namun tidak semua aplikasi

disertakan, tergantung sifat dari aplikasi tersebut komersial atau open source. Sementara ini yang bersifat open source saja yang disertakan. Contoh aplikasi open source GCC, G++, Xfix, Latex, dll.

Memulai bekerja dengan Linux

Untuk memulai bekerja pada sistem operasi Linux hal yang paling awal dilakukan adalah proses login. Pada proses ini Linux akan meminta pengguna memasukkan username dan password. Jika berhasil maka pengguna telah berada dalam lingkungan sistem operasi Linux.

Login dan Logout

Ada dua jenis login yang dapat dilakukan dalam Linux. Yang pertama berbasis text atau lebih dikenal dengan istilah TTY, yang kedua berbasis GUI. Pada login berbasis text user akan diminta memasukkan username dan password seperti berikut ini :

Login : <username> tekan ‘enter’

Password : <password>

Setelah berhasil memasukkan dengan benar username dan password maka akan muncul prompt di monitor dengan ciri terdapat karakter ‘\$’.

mrbee@froodo:~\$

Arti dari tulisan tersebut adalah “mrbee” merupakan username sedangkan @ merupakan simbol tempat atau lokasi dan “froodo” merupakan nama host atau komputer, sehingga jika diartikan user “mrbee” login pada host bernama “froodo”.

Setelah proses login dilalui maka proses selanjutnya adalah bergantung pada keinginan pengguna. Aturan mainnya sangat sederhana hanya cukup menuliskan perintah-perintah yang diterima oleh Linux kemudian tekan ‘enter’, jika perintah yang dituliskan benar maka sistem akan merespon keluaran perintah tersebut.

Sebagai contoh :

mrbee@froodo:~\$ls ‘enter’

Perintah ‘ls’ akan menampilkan seluruh isi dari direktori saat itu. Setelah selesai memainkan perintah yang bisa dijalankan dalam lingkungan sistem operasi Linux, jika diinginkan untuk keluar dari shell maka perintah yang dapat digunakan adalah “logout” atau bisa juga dengan tekan “ctrl-d”. Jika perintah ini benar maka sistem akan keluar dan tampilan monitor akan menjadi :

Login :

Maka secara umum operasi perintah dalam Linux dapat disimpulkan ke dalam bentuk umum

\$<perintah> <option> “enter”

Petunjuk praktikum

1. Nyalakan komputer dan pilih system operasi Linux yang tersedia
2. Tunggu proses booting selesai yaitu pada saat keluar permintaan untuk memasukkan username dan password. Masukkan username kemudian tekan enter.
3. Buka Applications – Accessories – Terminal. Untuk menggunakan command line.
4. Jika menggunakan ubuntu login root dengan menggunakan “sudo su” kemudian ‘enter’, setelah itu masukkan password milik user admin.(tanyakan kepada asisten praktikum)

Setelah proses selesai dan berada dalam shell, tuliskan perintah-perintah berikut ini.

Perintah harus dijalankan kemudian analisis atau maknai respon yang muncul pada layar monitor. Contoh :

Daftar perintah :

echo halo dunia ‘enter’
date ‘enter’
hostname ‘enter’
arch ‘enter’
uname -a ‘enter’
dmesg more ‘enter’ (tekan ‘q’ untuk keluar)
uptime ‘enter’
whoami ‘enter’
who ‘enter’
id ‘enter’
last ‘enter’
finger ‘enter’
w ‘enter’
top ‘enter’ (tekan ‘q’ untuk keluar)

echo \$\$HELL ‘enter’
echo {con,pre} {sent,fer} {s,ed} ‘enter’
man ls ‘enter’ (tekan ‘q’ untuk keluar)
man who ‘enter’ (tekan ‘q’ untuk keluar)
who can tell me about linux ‘enter’
last ‘enter’
clear ‘enter’
fdisk -l ‘enter’
users ‘enter’
cat /etc/fstab ‘enter’
cal 2000 ‘enter’
cal 9 1752 ‘enter’
cal 10 2007 ‘enter’
bc -l ‘enter’ (tulis “quit” untuk mengakhiri)
echo 5+4 bc -l ‘enter’
yes please ‘enter’ (tekan ‘ctrl-c’ untuk keluar)
pwd ‘enter’
history ‘enter’
tail -f /var/log/message ‘enter’ (tekan ctrl-c untuk keluar)
lsmod ‘enter’
ps -axu ‘enter’
lspci ‘enter’
free ‘enter’
cat /proc/cpuinfo ‘enter’
finger root ‘enter’
reboot ‘enter’
halt ‘enter’

5. Catat versi kernel yang digunakan saat itu dengan menggunakan perintah “uname -a” kemudian tekan ‘enter’.

Tugas

1. Jelaskan distro linux yang ada saat ini (minimal 5).
2. Jelaskan 20 perintah yang sama diantara masing-masing distro.
3. Jelaskan maksud perintah ‘init 0’, ‘init 1’, ‘init 2’, ‘init 3’, ‘init 4’, ‘init 5’, dan ‘init 6’.
4. Jelaskan maksud dari perintah ‘quota’

Modul 6 : Instalasi sistem operasi linux, dan menambah aplikasi linux

Pendahuluan

Linux Ubuntu 18.04 LTS memiliki dua varian yang dapat digunakan sebagai desktop dan server. Perbedaan yang paling menonjol dari varian Ubuntu 18.04 LTS desktop dan server adalah pada versi desktop memiliki GUI (grafik user interface). Ubuntu Desktop memiliki antar muka yang menarik untuk para pengguna. Sedangkan pada versi Ubuntu LTS server tidak memiliki GUI, sehingga antar muka bagi pengguna hanya berupa shell atau layar terminal.

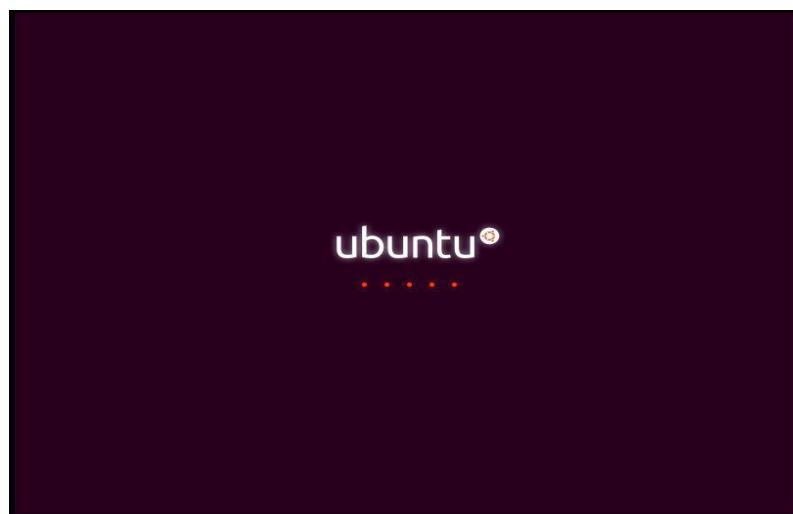
Alasan yang mendasar mengapa Ubuntu 18.04 LTS tidak memiliki GUI yaitu dikarenakan sebuah GUI membutuhkan memori pemroses yang cukup besar dan dapat menyita atau mengurangi kemampuan fungsi komputer. Dengan tidak adanya GUI pada komputer server, diharapkan kegiatan pemrosesan menjadi lebih cepat. Dalam melakukan instalasi sistem operasi linux ubuntu 18.04 dibutuhkan beberapa komponen agar kegiatan instalasi dapat berjalan sukses. Komponen yang mendasar dalam persiapan instalasi Ubuntu 18.04 LTS Desktop adalah dengan mempersiapkan spesifikasi komputer yang memadai. Ubuntu 18.04 LTS Desktop membutuhkan spesifikasi komputer sebagai berikut:

1. Hardisk 25 GB (ATA, SATA/SCSI)
2. Processor dual core 2 GHz atau lebih
3. Memori 2 GB
4. VGA (64 bit)
5. Mouse and Keyboard (USB/PS2)
6. Monitor (CRT/LCD)
7. CD/DVD ROM
8. CD Ubuntu 18.04 LTS Desktop

Untuk mendapatkan CD Ubuntu 18.04 LTS Desktop dapat mengakses di internet dengan memilih folder Ubuntu dan mendownload file bertipe .iso dengan nama Ubuntu_18.04_LTS/Desktop.iso dengan ukuran file sekitar 1,85 GB, atau langsung mengunjungi situs resmi www.ubuntu.com.

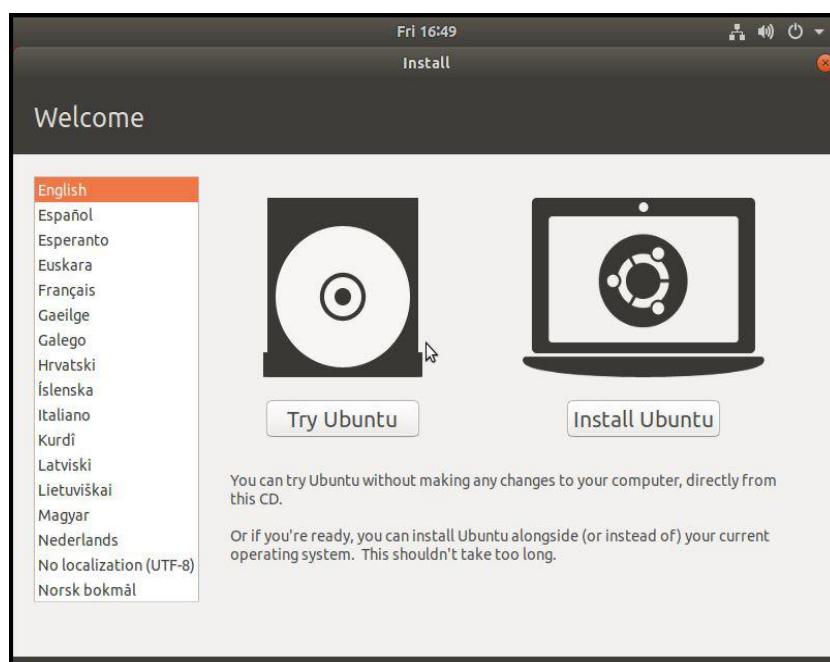
Petunjuk Praktikum

Melakukan instalasi Ubuntu 18.04 LTS Desktop dapat dimulai dengan mengubah struktur booting pada BIOS, dengan menjadikan CD/DVD ROM sebagai perangkat keras utama yang akan dibaca ketika pertama kali komputer dijalankan. Komputer akan membaca CD Ubuntu 18.04 LTS Desktop kemudian melakukan proses pembacaan awal seperti yang ditunjukkan pada gambar 6.1.



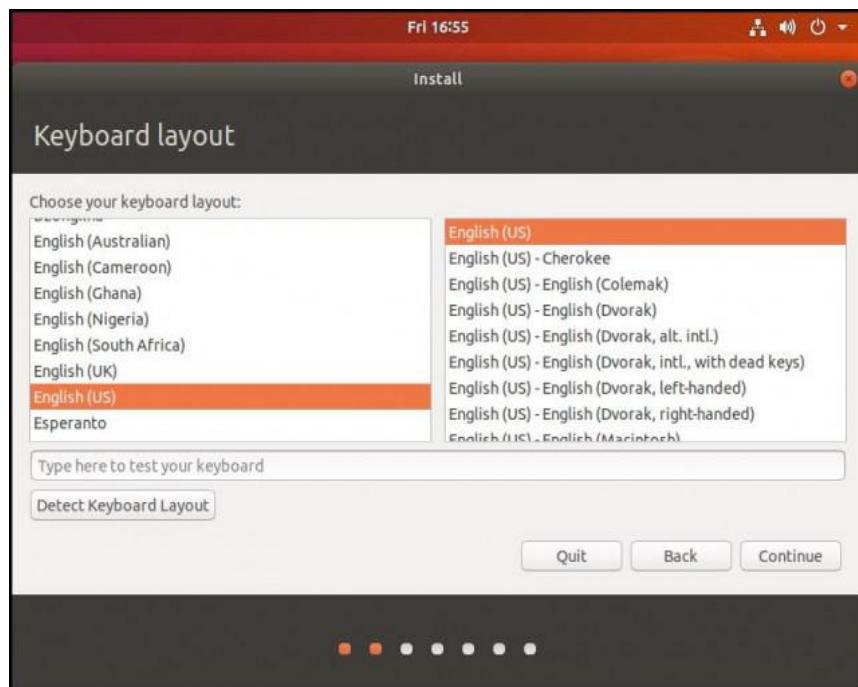
Gambar 6.1 Proses awal pembacaan CD Ubuntu

Komputer membutuhkan beberapa menit untuk melakukan proses pembacaan CD Ubuntu 18.04 LTS Desktop.



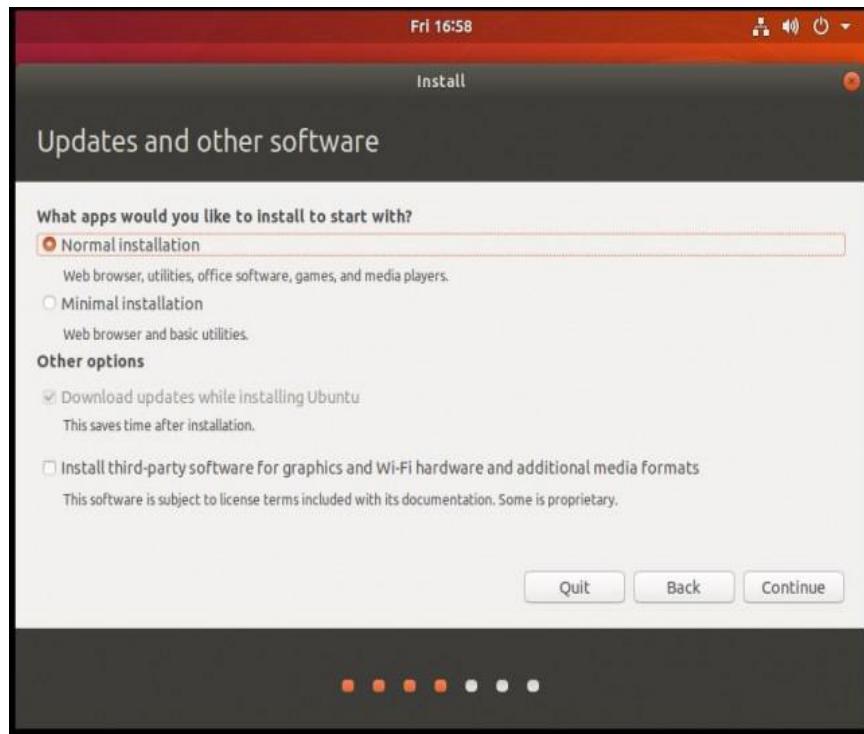
Gambar 6.2 Gambar 2.2 Live CD dan Install

Proses selanjutnya adalah live cd dan install, seperti yang ditunjukkan pada gambar 6.2 dimana terdapat dua pilihan. Pilihan pertama dimaksutkan untuk penggunaan Ubuntu 18.04 LTS Desktop dengan live cd. Istilah live cd artinya pengguna dapat menjalankan sistem operasi Ubuntu 18.04 LTS Desktop dengan menggunakan cd installer. Pilihan yang kedua adalah pengguna dapat melanjutkan instalasi Ubuntu 18.04 LTS Desktop secara permanen pada PC (personal computer). Lanjutkan dengan menekan tombol Install Ubuntu 18.04 LTS.



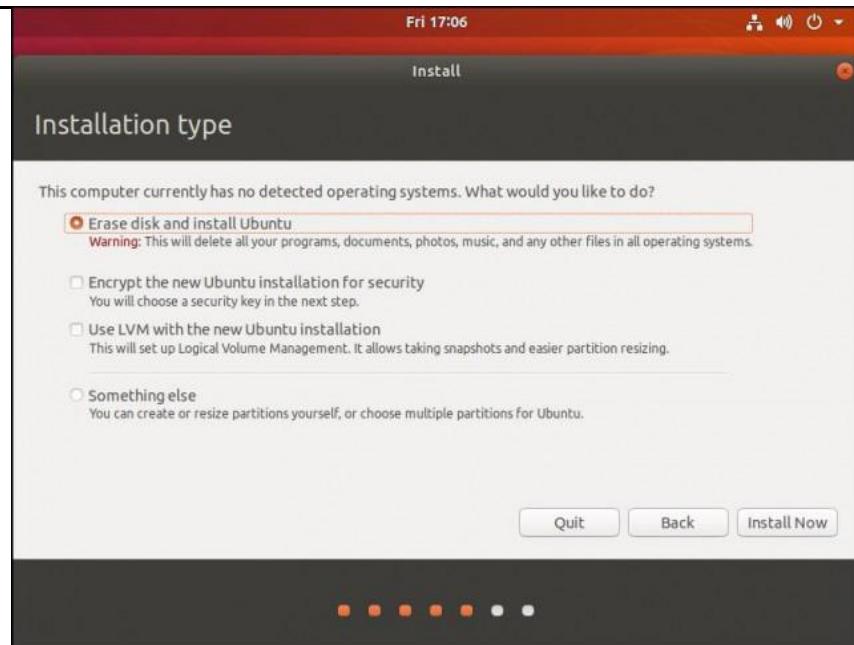
Gambar 6.3 Pemilihan keyboard layout

Tahapan selanjutnya yaitu mengatur keyboard layout sesuai dengan yang digunakan, default layout keyboard yang adalah **US** selanjutnya klik continue.



Gambar 6.4 Pemilihan Metode instalasi

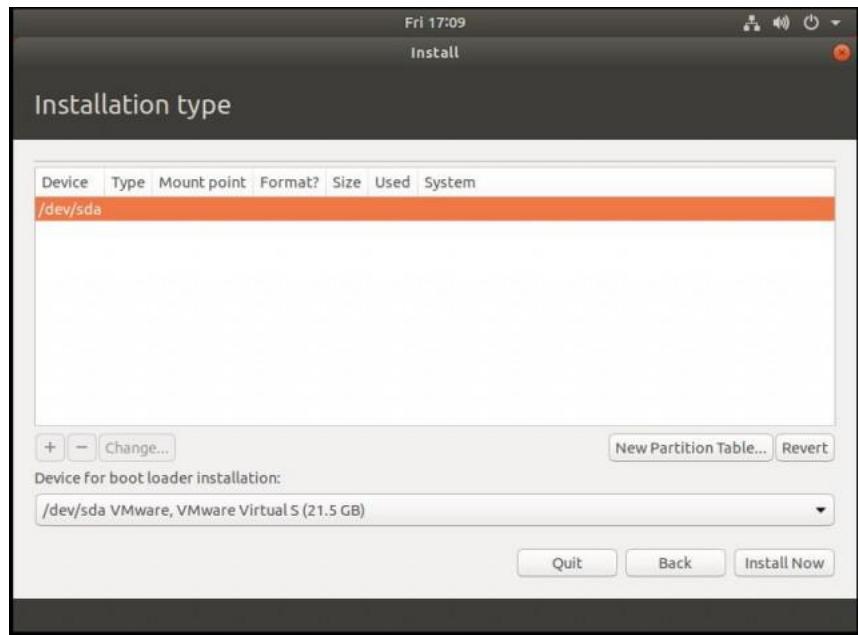
Pada langkah berikutnya adalah memilih metode instalasi, terdapat "Normal Installation" dan "Minimal Instalation". Pada normal instalation ubuntu yang di install akan otomatis terinstall aplikasi office seperti sepaket libre office, beberapa aplikasi utilitas, beberapa game dan media player. sedangkan pada metode Minimal Instalation ubuntu yang akan diinstall hanya memiliki web browser dan beberapa aplikasi utilitas saja. Pilih saja Normal Instalation agar Ubuntu yang sudah diinstal bisa langsung digunakan. pada pilihan "install third party ..." lakukan unchecklist karena ini akan membuat proses instalasi menjadi semakin lama. lalu klik Continue.



Gambar 6.5 Partisi Hard Disk Manual

Bagian terpenting dari instalasi Ubuntu 18.04 LTS Desktop adalah pada bagian partisi, bagi para pemula ini adalah tahapan yang menyulitkan. Jika komputer merupakan komputer baru maka bisa memilih “erase disk and install ubuntu”. Hal ini akan membuat proses instalasi semakin cepat. Untuk proses pembelajaran diharapkan untuk memilih bagian “something else” (advanced), lanjutkan dengan menekan tombol Forward. Pembagian hard disk secara manual untuk Ubuntu

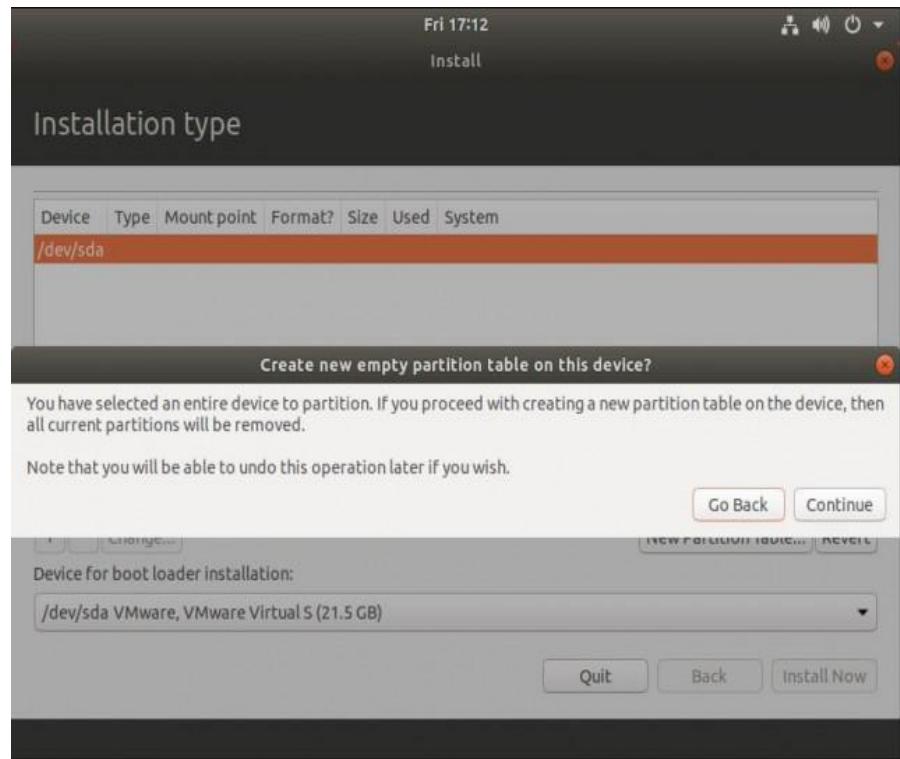
LTS Desktop seperti dijelaskan pada contoh pembagian hard disk berikut ini.
kapasitas hard disk yang dimiliki 21,5 GB ram
• **swap** 4 GB (4096 MB) dengan perhitungan jumlah swap = 2 x RAM
• file sistem / **root** = 17 GB



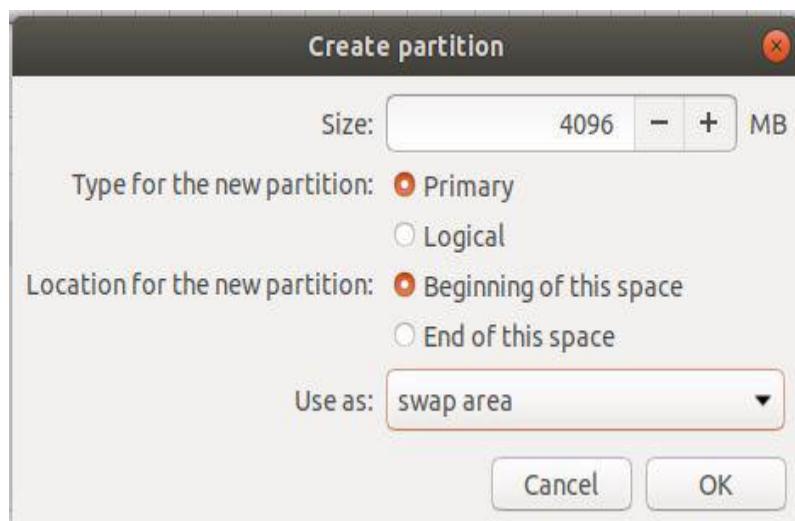
Gambar 6.6 Persiapan Pembagian Hard Disk

Perhatikan gambar 6.6, halaman tersebut menampilkan perangkat keras dengan nama **/dev/sda**. Penamaan hard disk dalam sistem operasi linux berbeda dengan sistem operasi lain seperti windows. Hard disk yang diberinama /dev/sda artinya hard disk tersebut diletakkan pada direktori /dev dan bernama sda. Semua perangkat keras dalam sistem operasi linux akan tersimpan pada direktori /dev. Makna dari sda adalah hard disk bertipe SATA. Jika hard disk bertipe ATA maka penamaan hard disk akan menjadi hda.

Arahkan kursor mouse pada /dev/sda kemudian lanjutkan dengan menekan tombol **New Partition Table** kemudian **continue**.

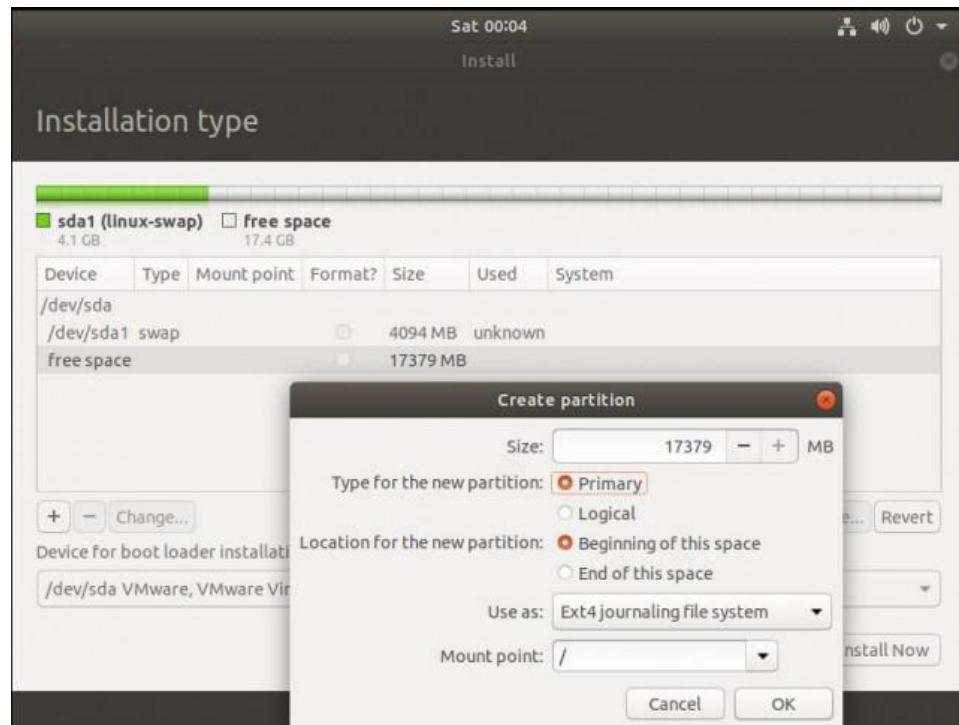
**Gambar 6.7 Format Hard Disk**

Memulai pembagian partisi dengan mengarahkan kursor pada **free space** lanjutkan dengan klik icon (+) dan isikan size sebesar dua kali ukuran ram.

**Gambar 6.8 Membuat Swap**

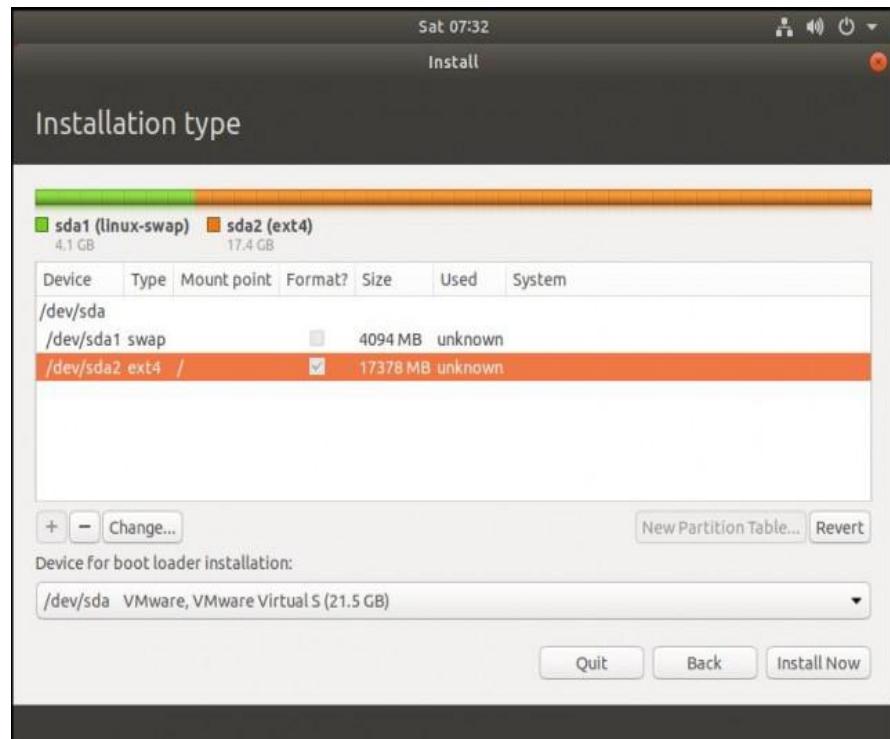
Swap adalah memori virtual yang digunakan sistem operasi linux untuk menyimpan proses. Memori swap atau sering disebut swap tanpa kata memori, tidak berbeda dengan memori utama yang sering disebut dengan RAM. Ukuran swap biasanya didapat dari rumus ($\text{swap} = 2 \times \text{RAM}$), jika $\text{RAM} = 512 \text{ MB}$ maka ukuran swap : $2 \times 512 = 1024 \text{ MB/1 GB}$. Swap digunakan linux apabila proses yang dilakukan sangat berat sehingga membutuhkan memori tambahan. Bagaimana bila RAM yang dimiliki sebesar 8 GB,

haruskah membuat swap sebesar 16 GB sesuai dengan rumus yang telah dibuat diatas. Ada beberapa pendapat mengatakan batas ukuran swap adalah 2 GB, sehingga bila ukuran RAM sangat besar ada kemungkinan linux tidak membutuhkan swap. Tekan OK untuk melanjutkan tahap selanjutnya.

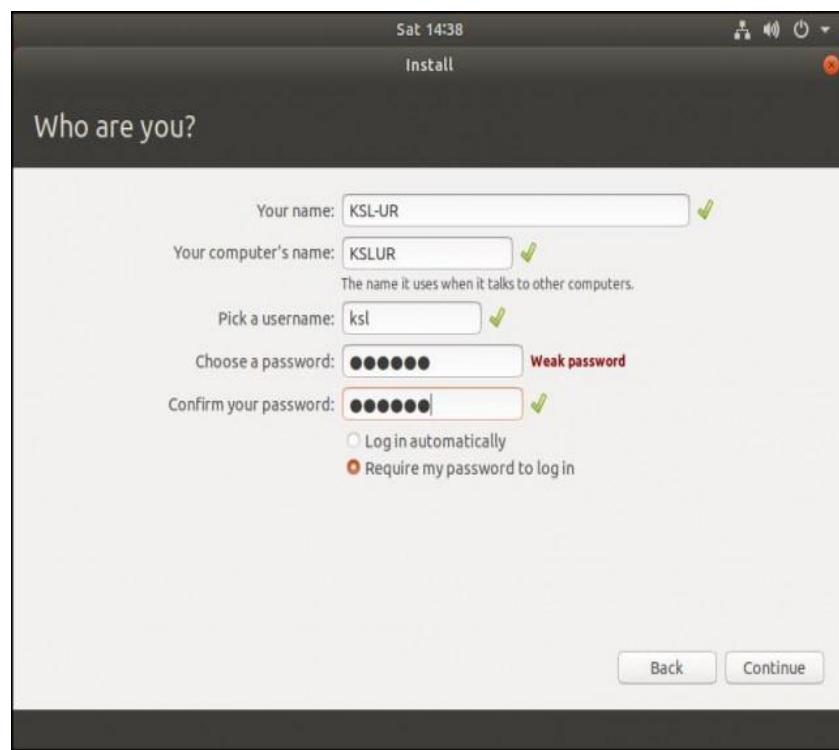


Gambar 6.9 Membuat File System

Proses pembagian hard disk yang terakhir adalah menjadikan sisa ruang hard disk sebagai file system. Pada gambar 6.9 dapat dilihat ukuran sebesar **17379 MB** dengan tipe file **Ext4 journaling file system**. Perhatikan juga pada bagian Mount point terdapat simbol “/” yang mengindikasikan sebagai tempat penyimpanan sistem operasi linux Ubuntu 18.04 LTS Desktop. Lanjutkan dengan menekan tombol OK.

**Gambar 6.10 Format Hard Disk**

Saat ini telah terbentuk dua bagian pada hard disk yang ditunjukkan pada gambar 6.10 di atas, terlihat dua tipe file yaitu **/dev/sda1 swap** dan **/dev/sda2 ext4**. Lanjutkan dengan menekan tombol **Install Now** dan **Continue**. Proses selanjutnya adalah mengatur zona waktu. Pilihlah lokasi Jakarta selanjutnya **continue**.

**Gambar 6.11 Membuat User Sistem**

Proses selanjutnya adalah membuat user sistem yang akan digunakan untuk login pertama kali pada sistem operasi linux. Ada tiga user yang diterapkan pada sistem operasi linux khususnya pada Ubuntu 18.04 LTS Desktop yaitu:

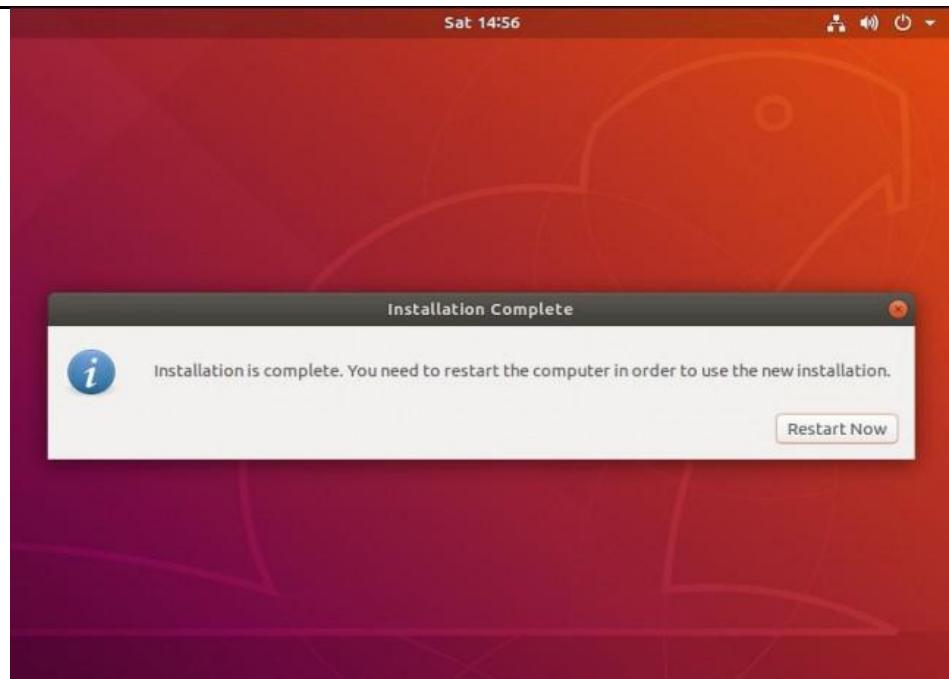
1. user sistem: memiliki kemampuan untuk merubah sistem konfigurasi secara menyeluruh
2. user administrator atau root: memiliki kemampuan penuh terhadap sistem linux, dengan kemampuan mengendalikan semua user dan konfigurasi sistem
3. user biasa: hanya memiliki kemampuan untuk mengelola desktop milik sendiri, tanpa ada kemampuan untuk merubah sistem konfigurasi.

Pada contoh di atas nama user adalah KSL-UR, user name login KSL-UR dan password login 6 karakter. Batasan panjang karakter adalah minimal 6 karakter. Selanjutnya menunggu proses instalasi selesai.



Gambar 6.12 Install Sistem

Proses install sistem dilakukan untuk menuliskan semua berkas dan membuat direktori pada /dev/sda2. Butuh waktu 10 sampa 30 menit, bergantung pada spesifikasi komputer yang digunakan.



Gambar 6.13 Install Sukses dan Restart Komputer

Tekan tombol Restart Now, kemudian cd install Ubuntu 18.04 LTS Desktop akan keluar dari CD ROM drive.



Gambar 6.14 Halaman Login Utama

Selesai sudah proses installasi Ubuntu 18.04 LTS Desktop. Tekan nama user dan masukkan password, kemudian tekan tombol Log In.

Menambah aplikasi

Kemampuan Ubuntu 18.04 LTS Desktop salah satunya adalah kemudahan dalam penambahan aplikasi yang disediakan oleh pengembang Ubuntu yaitu perusahaan Canonical Ltd asal Afrika yang didirikan oleh Mark Shuttleworth pada tahun 2004. Nama Ubuntu diambil dari ideologi di Afrika Selatan, kata Ubuntu berasal dari bahasa kuno Afrika yang artinya “rasa perikemanusiaan terhadap sesama manusia”.

Untuk menambah aplikasi pada Ubuntu 18.04 LTS Desktop, dapat menggunakan perangkat bantu yakni **Ubuntu Software Center** yang terletak pada menu Applications. Tetapi harus menjalankan perintah **sudo apt update** pada terminal.

Tugas.

1. Menggunakan laptop masing-masing instalkan 10 aplikasi yang berbeda pada linux, dan sebutkan fungsinya.

No	Aplikasi	Tampilan	Fungsi
1			
2			
3			
4			
5			
6			
7			

Modul 7 : Manajemen User dan Permission direktori/file pada Linux

Tujuan

- Peserta Praktikum Operasi dapat melakukan manajemen user dan quota.
- Peserta Praktikum Operasi dapat mengenal permission dan ownership file/direktori
- Mengetahui permission dan ownership pada tiap objek file atau direktori pada Linux

Pendahuluan

A. Manajemen User

1. Membuat User Baru

Membuat sebuah account user yang baru sangatlah mudah. Untuk membuat user baru dari

command line, kita dapat menjalankan perintah useradd . Sebagai contoh, untuk membuat sebuah user yang bernama Hana, kita dapat menjalankan perintah sbb:

```
$ useradd hana
```

Untuk memberikan password kepada user baru yang telah dibuat dengan mengetikkan perintah
passwd.

Contoh:

```
$ passwd hana
```

Setelah mengetikkan perintah di atas, maka kita akan diminta untuk memasukkan password yang kita inginkan dua kali. Ingat, untuk mengganti password user jangan sampai lupa menambahkan nama user.

Dalam default system, ketika user baru ditambahkan, maka user baru tersebut akan mempunyai home directory di bawah /home/. Sebagai contoh, ketika kita mengeksekusi perintah di atas (useradd hana), maka user yang bernama hana akan memiliki home directory /home/hana.

Untuk mengubah default system ini, kita dapat memberikan option-option lain setelah perintah useradd. Misalnya kita menginginkan agar si hana diberikan home directory di /www/hana, maka kita dapat memberikan option -d setelah perintah useradd.

Contoh:

```
$ useradd -d /www/hana
```

Sebagai catatan, useradd hanya membuat direktori akhir, bukan semua path. Sehingga bila kita memberikan perintah useradd -d /www/hana, maka useradd akan menciptakan direktori hana apabila telah ada direktori yg bernama /www. Program useradd akan menciptakan suatu group bagi setiap user yg ditambahkan. Setiap user akan tergabung dalam group-group tertentu. Program useradd akan memberikan nama group yang sama seperti nama user yang dibuat. Misalnya ketika kita membuat user baru dengan perintah useradd hana, maka si hana akan berada dalam group hana. Untuk mengubah default ini, maka kita dapat memberikan parameter -g setelah perintah useradd.

Contoh:

```
$ useradd hana -g users
```

Hal tersebut akan membuat user hana tergabung dalam group users.

2. Membuat group baru

Untuk membuat group baru kita dapat mengetikkan perintah groupadd.

Contoh:

```
$ groupadd hacker
```

Perintah ini akan menambah group baru yang bernama hacker di /etc/group dan pada /etc/gshadow jika kita menggunakan shadow password.

3. Memodifikasi account user yang sudah ada

a) Mengubah password

Seorang system administrator akan sering berhubungan dengan masalah user yang lupa akan passwordnya. Perintah di bawah ini akan mengganti password user yang lama dengan password yang baru:

```
$ passwd
```

Misalkan si hana lupa akan passwordnya atau ingin mengubah passwordnya, maka si hana atau system administrator yang bersangkutan dapat mengubahnya dengan mengetikkan perintah:

```
$ passwd hana
```

b) Mengubah home directory

Untuk mengubah home directory dari user yang sudah ada, kita dapat mengetikkan perintah:

```
$ usermod -d
```

Contoh:

```
$ usermod -d /home2/hana hana
```

Jika ternyata home directory hana yang lama telah berisi file-file kepunyaan hana, maka kita dapat memindahkan home directory hana beserta file2nya dengan option -m.

Contoh:

```
$ usermod -d -m /home2/hana hana
```

c) Menghapus account user

Untuk menghapus account user yang sudah ada, kita dapat menggunakan perintah userdel. Contoh:

```
$ userdel hana
```

Perintah tersebut akan menghapus user yang bernama hana. Jika kita ingin menghapus home directory user beserta isi file atau direktori yang berada di bawahnya, maka kita dapat memberikan option -r setelah userdel.

B. Mengenal Permission direktori/file pada Linux

Pada unix (juga Linux), setiap file dan direktori dapat ditentukan hak aksesnya. Hak yang dapat ditentukan : read (r=4), write (w=2), dan execute (x=1). Anda dapat menentukan hak akses setiap direktori dan file sesuai dengan yang diinginkan dengan menggunakan perintah chmod. Melihat informasi yang dimiliki oleh file dan/atau direktori dapat menggunakan perintah :

```
$ ls -l
```

Berdasarkan tampilan yang dihasilkan, bagian kolom pertama akan terbagi dalam 10 sub kolom.

Sub kolom 1 menyatakan tipe file. – adalah file, d adalah direktori, l adalah link atau shortcut ke file/direktori.

Sub kolom 2,3,4 mewakili hak akses rwx untuk pemilik file/direktori.

Sub kolom 5,6,7 mewakili hak akses rwx untuk group/kelompok pemilik file/direktori.

Sub kolom 8,9,10 mewakili hak akses rwx untuk semuanya (everyone) selain pengguna dan group.

Pemilik dari file/direktori dilihat pada kolom tiga dan group dilihat pada kolom empat.

Petunjuk praktikum

1. Nyalakan komputer dan pilih system operasi Linux yang tersedia
2. Tunggu proses booting selesai yaitu pada saat keluar permintaan untuk memasukkan username dan password. Masukkan username kemudian tekan enter.
3. Buka Applications – Accessories – Terminal. Untuk menggunakan command line.
4. Jika menggunakan ubuntu login root dengan menggunakan “sudo su” kemudian ‘enter’, setelah itu masukkan password milik user admin.(tanyakan kepada asisten praktikum)
5. Setelah proses selesai dan berada dalam shell, tuliskan perintah-perintah berikut ini. Perintah harus dijalankan kemudian analisis atau maknai respon yang muncul pada layar monitor. Contoh :

Daftar perintah :

Praktikum 1:

Kerjakan perintah berikut:

1. Buat user baru: # useradd user<nim anda>
2. Masukkan pasword yang diminta sebanyak dua kali:
Enter new UNIX password:
3. Jika sudah berhasil akan muncul pesan
Retype new UNIX password:

```
passwd: password updated successfully
```

Dilanjutkan dengan memasukkan user information untuk user yang anda buat, masukkanlah data-data yang diminta.

Contoh tampilan:

```
Changing the user information for user<nim anda>
```

```
Enter the new value, or press ENTER for the default
```

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n] Y

Masukkanlah data-data anda.

4. Keluarlah dari shell dengan mengetikkan exit kemudian tutup jendela shell
5. Keluarlah dari gnome dengan mengklik tombol Log Out
6. Setelah masuk ke jendela login, cobalah untuk masuk menggunakan user dan pasword yang baru saja anda buat.

Setelah berhasil login, lanjutkan dengan langkah praktikum selanjutnya. Apa bila belum berhasil, anda harus mengulangi langkah paraktikum ini terlebih dahulu.

Praktikum 2:

Kerjakan perintah berikut :

1. Buat file latihan : \$ touch latihan
2. Ketikkan perintah : \$ chmod 666 latihan
3. Lihat hasilnya dengan : \$ ls -l (perhatikan izin aksesnya)
4. Ketikkan perintah : \$ chmod 640 latihan

5. Lihat hasilnya dengan : \$ ls -l (perhatikan izin aksesnya)
6. Ketikkan perintah : \$ chmod 111 latihan
7. Lihat hasilnya dengan : \$ ls -l (perhatikan izin aksesnya)
8. Ketikkan perintah : \$ chmod 222 latihan
9. Lihat hasilnya dengan : \$ ls -l (perhatikan izin aksesnya)
10. Ketikkan perintah : \$ chmod 333 latihan

angka konfigurasi yang dapat dikombinasikan adalah angka 0-7.

Praktikum 3:

Hak akses juga dapat ditambahkan, selain diubah seperti cara di atas. Untuk melakukan perubahan hak akses, kita menggunakan operand +x, +r, +w,-r,-w,dan -x.

Kerjakan perintah berikut :

1. \$ chmod 000 latihan (tidak memberikan hak akses)
2. \$ ls -l latihan
3. \$ chmod +r latihan (menambahkan hak akses read)
4. \$ ls -l latihan
5. \$ chmod +w latihan (menambahkan hak akses write)
6. \$ ls -l latihan
7. \$ chmod +x latihan (menambahkan hak akses execute)
8. \$ ls -l latihan
9. \$ chmod -x latihan (menghilangkan hak akses execute)
10. \$ ls -l latihan
11. \$ chmod -w latihan (menghilangkan hak akses write)
12. \$ ls -l latihan
13. \$ chmod -r latihan (menghilangkan hak akses read)
14. \$ ls -l latihan

Perintah +operand/-operand pada chmod akan mengubah hak akses sesuai dengan default set. Kita akan membuat file skrip shell untuk dieksekusi di dalam direktori /bin. Disini kita akan melakukan perubahan hak akses agar file tersebut bisa dieksekusi.

Praktikum 4

Jalankan perintah berikut :

1. Masuklah ke dalam direktori /bin : \$ cd /bin
2. Buat file bernama : info.sh : \$ nano info.sh
3. Ketik isinya sebagai berikut :

```
# !/bin/sh
WAKTU="Tanggal dan jam saat ini : \c"
JMLUSER="Jumlah user : \c"
AKU="Status personal : \c"
echo -e "$WAKTU"
date
echo -e "$JMLUSER"
who | wc -l
echo -e "$AKU"
whoami
exit 0
```

4. Simpan dan keluar dengan menekan **ctrl+o <ENTER>** selanjutnya **ctrl+x**
5. Ketikkan perintah log info.sh : \$ info.sh
6. Ubah tipenya menjadi file yang bisa dieksekusi : \$ chmod 777 info.sh
7. Ketikkan perintah info.sh : \$ info<nim anda>.sh
8. Apa hasilnya?

Modul 8: System Call

Tujuan:

- Membuat sebuah ‘child process’ baru menggunakan system call ‘fork’.
- Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunakan perintah system call ‘wait’.
- Loading sebuah program yang dapat dieksekusi dalam sebuah child proses dengan menggunakan perintah system call ‘exec’.
- Menampilkan status sebuah file menggunakan system call ‘stat’.
- Untuk menampilkan isi direktori menggunakan perintah system call ‘readdir’.

Pendahuluan:

Dalam bidang komputer, istilah ‘system call’ adalah satu cara bagaimana sebuah aplikasi meminta layanan dari kernel (bagian utama) sistem operasi. Beberapa contoh antara lain layanan yang terkait dengan akses hardware seperti membaca hard disk drive, membuat dan mengeksekusi proses baru, dan berkomunikasi dengan layanan terpadu kernel seperti ‘process scheduling’. System call menyediakan antarmuka yang sangat penting antara proses dan sistem operasi. Beberapa fungsi penting dalam system call yang akan dibahas dalam praktikum ini antara lain adalah fork(), getpid(), getppid(), wait(), execl(), exit(), stat(), opendir(), readdir(), dan closedir().

Keterangan untuk setiap fungsi system call diberikan dalam bagian berikut ini:

fork()

system call fork ini digunakan untuk membuat sebuah proses baru disebut ‘child process’.

(a) return value berupa angka NOL (0) jika proses merupakan ‘child process’, (b) return value berupa bilangan NEGATIF jika pemuatan proses baru TIDAK berhasil. (c) Jika proses berupa ‘parent process’ maka return value berupa sebuah angka positif (> 0).

‘Child process’ adalah sebuah proses tiruan yang sama persis dengan ‘Parent process’.

Kedua jenis proses, ‘parent’ dan ‘child’, akan mengeksekusi secara terus menerus semua perintah yang diberikan setelah pemanggilan fungsi fork. ‘child’ dapat memulai eksekusi sebelum proses ‘parent’ dan sebaliknya.

getpid() dan getppid()

System call ‘getpid’ akan memberikan sebuah return value berupa proses ID dari proses yang dipanggil. Sedangkan ‘getppid’ akan memberikan sebuah return value berupa ‘parent’ proses ID dari proses yang dipanggil.

wait()

System call ‘wait’ jika dipanggil akan menyebabkan ‘parent’ proses akan diblok (dihentikan) sampai dengan ‘child’ proses berakhir. Ketika proses ‘child’ berakhir, kernel akan memberitahu kepada proses ‘parent’ dengan cara mengirimkan sinyal ‘SIGCHLD’ kepada proses parent. Jika wait tidak digunakan maka ‘parent’ proses akan berakhir lebih dulu dan akan meninggalkan sebuah ‘sampah’ (berasal dari child proses).

execl()

Merupakan salah satu anggota kelompok fungsi ‘exec’, anggota yang lainnya adalah execv, execle, execve, execlp, execvp. Fungsi ini digunakan oleh ‘child’ proses untuk ‘loading’ sebuah program dan mengeksekusinya. System call ‘exec’ memerlukan ‘path’, nama program dan sebuah pointer null.

exit()

System call ‘exit’ digunakan untuk menghentikan sebuah proses, baik secara dihentikan secara normal atau tidak normal.

stat()

System call stat digunakan untuk memperoleh informasi mengenai sebuah file dan strukturnya.

opendir(), readdir() dan closedir()

System call ‘opendir’ digunakan untuk membuka sebuah direktori, (a) akan menghasilkan sebuah POINTER yang menunjuk ke posisi direktori yang pertama. (b) Pointer akan bernilai NULL jika terjadi sebuah kesalahan (error).

System call ‘readdir’ digunakan untuk membaca direktori sebagai struktur ‘dirent’ (a) Akan menghasilkan sebuah pointer yang menunjuk pada entry direktori selanjutnya. (b) Pointer akan bernilai NULL jika terjadi error atau bertemu dengan tanda ‘END-OF-FILE’

System call ‘closedir’ digunakan untuk menutup direktori hanya dilakukan oleh kernel.

Peralatan

- Software: Sistem operasi Linux, dilengkapi dengan kompiler gcc, dan editor text (nano atau vi)
- Hardware: PC desktop/laptop
- koneksi internet (untuk akses compiler-online)

Langkah kerja

Gunakan aplikasi ‘nano’ atau ‘vi’ atau teks editor yang lain untuk mengedit kode program berikut, Selanjutnya untuk melakukan kompilasi dapat dilakukan dengan perintah berikut:

\$gcc ‘nama_file.c’

Jika tidak ada kesalahan maka akan dihasilkan sebuah program bernama ‘a.out’, dan untuk menjalankan program tersebut dapat dilakukan dengan cara berikut:

\$./a.out

Jika pada PC anda tidak tersedia compiler ‘gcc’ dapat digunakan fasilitas online compiler yang disediakan oleh link berikut:

http://www.tutorialspoint.com/compile_c_online.php

- (a) Membuat sebuah ‘child process’ (proses baru) dengan menggunakan system call ‘fork’ .

Membuat program dengan algoritma sebagai berikut: (contoh program diberikan pada bagian berikutnya).

- (1) Deklarasi sebuah variabel x yang akan diakses bersama antara child proses dan parent proses.
- (2) Membuat sebuah child proses menggunakan system call fork.
- (3) Jika return value bernilai -1, tampilkan teks ‘Pembuatan proses GAGAL’, dilanjutkan dengan keluar program dengan perintah system call ‘exit’.
- (4) Jika return value sama dengan 0 (NOL), Tampilkan teks ‘Child Process’, tampilkan ID proses dari child proses menggunakan perintah system call ‘getpid’ ,

- tampilkan nilai x, dan tampilkan ID proses parent dengan perintah system call ‘getppid’.
- (5) Untuk nilai return value yang lainnya, tampilkan teks ‘Parent process’, tampilkan ID dari parent proses menggunakan perintah system call getpid, tampilkan nilai x, dan tampilkan ID dari proses shell menggunakan perintah system call getppid.
 - (6) Stop

Kode program:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 main() {
6     pid_t pid;
7     int x = 5;
8     pid = fork();
9     x++;
10    if (pid < 0)
11    {
12        printf("Process creation error"); exit(-1);
13    }
14    else if (pid == 0)
15    {
16        printf("Child process:");
17        printf("\nProcess id is %d", getpid());
18        printf("\nValue of x is %d", x);
19        printf("\nProcess id of parent is %d\n\n", getppid());
20    }
21    else
22    {
23        printf("\nParent process:");
24        printf("\nProcess id is %d", getpid());
25        printf("\nValue of x is %d", x);
26        printf("\nProcess id of shell is %d\n", getppid());
27    }
28}

```

- (b) Menghentikan sementara (block) proses parent sampai dengan proses child selesai, menggunakan perintah system call ‘wait’.

Membuat program dengan algoritma sebagai berikut, contoh program diberikan pada bagian berikutnya.

- (1) Membuat sebuah child proses menggunakan sytem call ‘fork’.
- (2) Jika return value bernilai -1, selanjutnya tampilkan teks ‘pembuatan proses gagal’, dan keluar program dengan menggunakan perintah system call ‘exit’.
- (3) Jika return value berupa angka positif (> 0), ‘pause’ hentikan sementara ‘parent’ proses tunggu sampai child proses berakhir dengan menggunakan perintah system call ‘wait’. Tampilkan teks ‘Parent starts’, selanjutnya tampilkan nomor genap mulai dari 0 s/d 10, terakhir tampilkan teks ‘Parent end’.
- (4) Jika return value bernilai 0 (NOL), tampilkan teks ‘Child start’, tampilkan nomor ganjil mulai dari 0 s/d 10, selanjutnya tampilkan teks ‘child ends’
- (5) Stop

Kode program:

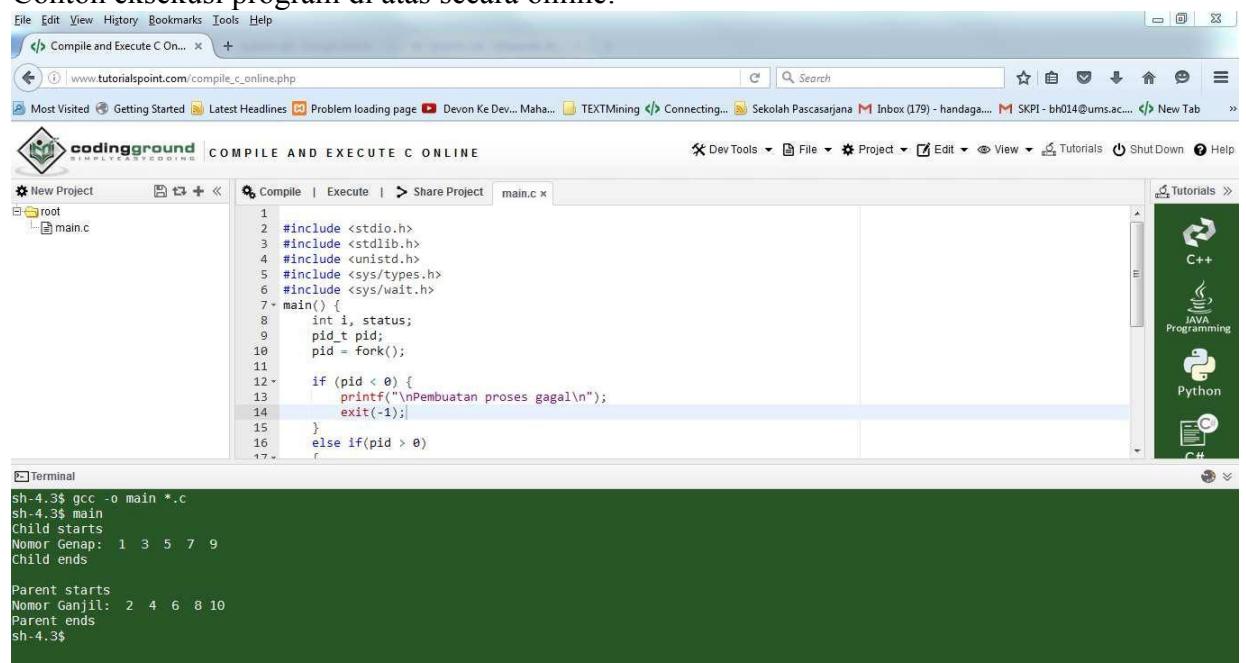
```

wait.c

1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 main()
8 {
9     int i, status;
10    pid_t pid;
11    pid = fork();
12
13    if (pid < 0) {
14        printf("\nPembuatan proses gagal\n");
15        exit(-1);
16    }
17    else if(pid > 0)
18    {
19        wait(NULL);
20        printf ("\nParent starts\nNomor Genap:");
21        for (i=2;i<=10;i+=2)
22            printf ("%3d",i);
23        printf("\nParent ends\n");
24    }
25    else if (pid == 0)
26    {
27        printf ("Child starts\nNomor Ganjil:");
28        for (i=1;i<10;i+=2)
29            printf ("%3d",i);
30        printf ("\nChild ends\n");
31    }
}

```

Contoh eksekusi program di atas secara online:



The screenshot shows the CodingGround online compiler interface. The code editor contains the 'main.c' file with the provided C code. The terminal window at the bottom shows the execution results:

```

sh-4.3$ gcc -o main *.c
sh-4.3$ main
Child starts
Nomor Genap:  1  3  5  7  9
Child ends

Parent starts
Nomor Ganjil: 2  4  6  8 10
Parent ends
sh-4.3$

```

- (c) Loading program yang dapat dieksekusi dalam sebuah ‘child’ proses menggunakan perintah system call ‘exec’.

Membuat program dengan algoritma sebagai berikut: (contoh program diberikan pada bagian berikutnya).

- (1) Jika terdapat 3 argumen dalam command-line berhenti (stop).
- (2) Membuat child proses dengan perintah system call ‘fork’
- (3) Jika return value adalah -1, selanjutnya tampilkan teks ‘Pembuatan proses Gagal’, dan keluar program dengan perintah system call exit.
- (4) Jika return value >0 (positif), selanjutnya hentikan parent-proses sementara hingga child-proses berakhir dengan menggunakan perintah system call wait. Tampilkan teks ‘Child berakhir’, dan hentikan parent-proses.
- (5) Jika return value sama dengan 0 (NOL), selanjutnya tampilkan teks ‘Child starts’, load program dari lokasi yang diberikan dalam ‘path’ ke dalam child-proses, menggunakan perintah system call ‘exec’. Jika return value dari perintah ‘exec’ adalah bilangan negatif, tampilkan error yang terjadi dan stop. Hentikan child-proses.
- (6) Stop

Contoh cara mengkompilasi dan menjalankan program

```
➤      $gcc exec.c  
➤      $ ./a.out /bin/ls ls
```

Kode Program:



```

1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5 main(int argc, char*argv[]) {
6
7     pid_t pid;
8     int i;
9
10    if (argc != 3)
11    {
12        printf("\nInsufficient arguments to load program");
13        printf("\nUsage: ./a.out <path> <cmd>\n"); exit(-1);
14    }
15
16    switch(pid = fork())
17    {
18        case -1:
19            printf("Fork failed");
20            exit(-1);
21        case 0:
22            printf("Child process\n");
23            i = execl(argv[1], argv[2], 0);
24            if (i < 0)
25            {
26                printf("%s program not loaded using exec system call\n", argv[2]);
27                exit(-1);
28            }
29        default:
30            wait(NULL);
31            printf("Child Terminated\n");
32            exit(0);
33    }
34}

```

(d) Menampilkan status file menggunakan perintah system call ‘stat’

Membuat program dengan algoritma sebagai berikut (contoh code ada di bagian berikutnya):

- (1) Gunakan ‘nama file’ yang diberikan melalui argumen dalam perintah command-line.
- (2) Jika ‘nama-file’ tidak ada maka stop disini (keluar program)
- (3) Panggil system call ‘stat’ pada ‘nama-file’ tersebut yang akan mengembalikan sebuah struktur.
- (4) Tampilkan informasi mengenai st_uid, st_blksize, st_block, st_size, st_nlink, etc.
- (5) Ubah waktu dalam st_time, st_mtime dengan menggunakan fungsi ctime.
- (6) Bandingkan st_mode dengan konstanta mode seperti S_IRUSR, S_IWGRP, S_IXOTH dan tampilkan informasi mengenai ‘file-permissions’.
- (7) Stop

Kode Program



```

stat.c * 
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <stdlib.h>
4 #include <time.h>
5 int main(int argc, char*argv[]) {
6     struct stat
7         file; int n;
8     if (argc != 2)
9     {
10         printf("Usage: ./a.out <filename>\n"); exit(-1);
11     }
12     if ((n = stat(argv[1], &file)) == -1)
13     {
14         perror(argv[1]);
15         exit(-1);
16     }
17     printf("User id : %d\n", file.st_uid);
18     printf("Group id : %d\n", file.st_gid);
19     printf("Block size : %d\n", file.st_blksize);
20     printf("Blocks allocated : %d\n", file.st_blocks);
21     printf("Inode no. : %d\n", file.st_ino);
22     printf("Last accessed : %s", ctime(&(file.st_atime)));
23     printf("Last modified : %s", ctime(&(file.st_mtime)));
24     printf("File size : %d bytes\n", file.st_size);
25     printf("No. of links : %d\n", file.st_nlink);
26     printf("Permissions : ");
27     printf( (S_ISDIR(file.st_mode)) ? "d" : "-");
28     printf( (file.st_mode & S_IRUSR) ? "r" : "-");
29     printf( (file.st_mode & S_IWUSR) ? "w" : "-");
30     printf( (file.st_mode & S_IXUSR) ? "x" : "-");
31     printf( (file.st_mode & S_IRGRP) ? "r" : "-");
32     printf( (file.st_mode & S_IWGRP) ? "w" : "-");
33     printf( (file.st_mode & S_IXGRP) ? "x" : "-");
34     printf( (file.st_mode & S_IROTH) ? "r" : "-");
35     printf( (file.st_mode & S_IWOTH) ? "w" : "-");
36     printf( (file.st_mode & S_IXOTH) ? "x" : "-");
37     printf("\n");
38     if(file.st_mode & S_IFREG)
39         printf("File type : Regular\n");
40     if(file.st_mode & S_IFDIR)
41         printf("File type : Directory\n");
42 }

```

(e) Menampilkan isi direktori menggunakan perintah system call ‘readdir’

Membuat program dengan algoritma sebagai berikut (contoh code ada di bagian berikutnya):

- (1) Gunakan ‘nama-direktori’ yang diberikan sebagai argumen pada command-line.
- (2) Jika direktori tidak ditemukan stop, keluar program
- (3) Buka direktori menggunakan perintah system call ‘opendir’ yang akan menghasilkan sebuah struktur.
- (4) Baca direktori menggunakan perintah system call ‘readdir’ yang juga akan menghasilkan struktur data.
- (5) Tampilkan d_name (nama direktori)
- (6) Akhiri pembacaan direktori dengan perintah system call ‘closedir’.
- (7) Stop

Kode Program:

```
dirlist.c      *
1 #include <stdio.h>
2 #include <dirent.h>
3 #include <stdlib.h>
4 main(int argc, char *argv[]){
5     struct dirent *dptr;
6     DIR *dname;
7
8     if (argc != 2)
9     {
10         printf("Usage: ./a.out <dirname>\n");
11         exit(-1);
12     }
13     if((dname = opendir(argv[1])) == NULL)
14     {
15         perror(argv[1]);
16         exit(-1);
17     }
18     while(dptr=readdir(dname))
19         printf("%s\n", dptr->d_name);
20
21     closedir(dname);
22 }
```

Modul-9: File System Call

Tujuan

- (a) Membuat program untuk membuat sebuah file dan menuliskan isinya.
- (b) Membaca file dan menampilkan isinya
- (c) Menambahkan isi sebuah file

Pendahuluan

Terdapat beberapa system call yang berkaitan dengan pembuatan dan pembacaan file, yaitu open, creat, read, write, dan close. Pada bagian ini akan dijelaskan cara membuat dan membaca file dengan menggunakan perintah yang tersedia dalam dalam file-system-call. Penjelasan dari perintah-perintah tersebut adalah sebagai berikut.

open()

Digunakan untuk membuka file yang sudah dibuat sebelumnya (ada dalam sistem) untuk dibaca atau ditulis atau dapat juga digunakan untuk membuat sebuah file baru. Fungsi ini memiliki return value berupa angka NEGATIF jika terjadi kesalahan. Pilihan (opsi) yang disarankan ketika menggunakan fungsi open antara lain adalah O_RDONLY, O_WRONLY, dan O_RDWR, selain itu juga tersedi opsi O_APPEND, O_CREAT, O_TRUNC dan lain-lain. Gabungan beberapa opsi dapat dilakukan dengan menggunakan operasi digital OR. Sedangkan MODE (pada parameter ketiga) digunakan untuk menentukan hak akses.

creat()

Digunakan untuk membuat file baru dan membuka file tersebut untuk ditulis. Fungsinya dapat digantikan oleh fungsi open() dengan opsi O_WRONLY|O_CREAT|O_TRUNC

read()

Membaca sejumlah byte tertentu dari sebuah file atau terminal. Jika proses pembacaan berhasil, fungsi tersebut akan memberi informasi (melalui return value) jumlah byte yang berhasil dibaca. Pembacaan berikutnya akan dimulai dari data selanjutnya sampai akhirnya bertemu dengan tanda END-OF-FILE yang menghasilkan return value bernilai 0 (NOL).

write()

Menulis sejumlah byte pada sebuah file. Setelah proses penulisan berhasil, penulisan selanjutnya akan dimulai dari posisi berikutnya. Jika memory tidak cukup akan menghasilkan error.

close()

Mengakhiri status file yang terbuka (sedang dibuka oleh fungsi creat atau open).

Peralatan

- Software: Sistem operasi Linux, dilengkapi dengan kompiler gcc, dan editor text (nano atau vi)
- Hardware: PC desktop/laptop
- koneksi internet (untuk akses compiler-online)

Langkah Kerja

A. Membuat sebuah file dan menuliskan data di dalamnya.

Membuat program dengan algoritma sebagai berikut (Contoh program diberikan pada bagian berikutnya):

- (1) Deklarasi variabel untuk buffer ‘buf’ untuk menyimpan 100 byte data.
- (2) Gunakan nama file yang diberikan sebagai argumen pada command-line
- (3) Buat sebuah file baru dengan nama file seperti pada (2) menggunakan fungsi system-call open() dengan opsi O_CREAT dan O_TRUNC.
- (4) Periksa ‘file-descriptor’, jika pembuatan file tidak berhasil, selanjutnya berhenti (stop) dan keluar dari program.
- (5) Baca input dari console sampai user menekan tombol Ctrl+D. Membaca 100 byte dari data yang dimasukan melalui console dan menyimpannya ke dalam variabel ‘buf’ menggunakan perintah system-call read(). Memindahkan isi variabel ‘buf’ ke dalam file menggunakan perintah ‘write’.
- (6) Menutup file dengan menggunakan fungsi ‘close’
- (7) Stop

Kode Program

```
fcreate.c *
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <fcntl.h>
5
6 main(int argc, char *argv[])
7 {
8     int fd, n, len;
9     char buf[100];
10    if (argc != 2)
11    {
12        printf("Usage: ./a.out <filename>\n");
13        exit(-1);
14    }
15    fd = open(argv[1], O_WRONLY|O_CREAT|O_TRUNC, 0644);
16    if(fd < 0)
17    {
18        printf("File creation problem\n");
19        exit(-1);
20    }
21
22    printf("Tekan Ctrl+D di akhir baris:\n");
23
24    while((n = read(0, buf, sizeof(buf))) > 0)
25    {
26        len = strlen(buf);
27        write(fd, buf, len);
28    }
29
30 }
```

Compilasi program dengan perintah \$gcc -o fcreate fcreate.c , selanjutnya jika proses compilasi berhasil jalankan program dengan memanggil nama file \$./fcreate test.txt, ketikan text yang akan disimpan dalam file, tekan Ctrl + D untuk menandai baris baru, dan tekan Ctrl + C jika sudah selesai.

B. Membaca sebuah file dan menampilkan isinya di layar.

Membuat kode program dengan algorithm sebagai berikut:

- (1) Deklarasi sebuah variabel buffer tipe character untuk menyimpan 100 byte data.
- (2) Gunakan nama-file sesuai dengan argumen yang diberikan dalam perintah command-line.
- (3) Buka file untuk dibaca menggunakan perintah ‘open’ dengan opsi O_RDONLY.
- (4) Periksa isi file-descriptor, Jika file tidak ada maka program berhenti, stop
- (5) Baca isi file per 100 byte data menggunakan perintah ‘read’ sampai ketemu dengan tanda akhir file, ‘END-OF-FILE’.
- (6) Tutup file menggunakan perintah ‘close’
- (7) Stop

Kode Program:

```
fread.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 main(int argc, char *argv[])
5 {
6     int fd,i;
7     char buf[100];
8     if (argc < 2)
9     {
10         printf("Usage: ./a.out <filename>\n");
11         exit(-1);
12     }
13     fd = open(argv[1], O_RDONLY);
14     if(fd == -1)
15     {
16         printf("%s file does not exist\n", argv[1]);
17         exit(-1);
18     }
19     printf("Isi dari file %s adalah : \n", argv[1]);
20     while(read(fd, buf, sizeof(buf)) > 0)
21         printf("%s", buf);
22     close(fd);
23 }
24 }
```

C. Menambah isi file.

Membuat kode program dengan algoritma sebagai berikut:

- (1) Deklarasi sebuah variabel buffer tipe character untuk menyimpan 100 byte data.
- (2) Gunakan nama-file sesuai dengan argumen yang diberikan dalam perintah command-line.
- (3) Buka file di atas dengan menggunakan perintah open dengan opsi O_APPEND.
- (4) Periksa file-descriptor, jika nilainya berupa angka negatif, stop program
- (5) Baca input user dari console sampai user menekan tombol Ctrl + D. Jika user sudah menekan tombol Ctrl + D, baca 100 byte dari console dan simpan ke dalam variabel ‘buf’ menggunakan perintah read. Selanjutnya tuliskan isi variabel ‘buf’ ke dalam file menggunakan perintah ‘write’.
- (6) Tutup file dengan menggunakan fungsi ‘close’.

Kode Program:

```
fappend.c * 
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <fcntl.h>
5 main(int argc, char *argv[])
6 {
7     int fd, n, len;
8     char buf[100];
9     if (argc != 2)
10    {
11        printf("Usage: ./a.out <filename>\n"); exit(-1);
12    }
13    fd = open(argv[1], O_APPEND|O_WRONLY|O_CREAT, 0644);
14    if (fd < 0)
15    {
16        perror(argv[1]);
17        exit(-1);
18    }
19    while((n = read(0, buf, sizeof(buf))) > 0)
20    {
21        len = strlen(buf);
22        write(fd, buf, len);
23    }
24    close(fd);
25 }
```

Hasil:

Telah dibuat program untuk membaca, menulis dan menambah isi sebuah file.

Kesimpulan:

File System call merupakan kelompok fungsi system-call yang berkaitan dengan proses pembuatan, pembacaan dan penulisan file.

Modul 10 : Simulasi Command (Perintah)

Tujuan

Membuat program untuk mensimulasi beberapa perintah dalam sistem operasi linux menggunakan kelompok fungsi yang tersedia dalam system-call. Beberapa command yang akan disimulasi antara lain adalah

- (a) Simulasi perintah ‘ls’
- (b) Simulasi perintah ‘grep’
- (c) Simulasi perintah ‘cp’
- (d) Simulasi perintah ‘rm’

Pendahuluan

Fungsi-fungsi yang tersedia dalam system call dapat digunakan untuk mensimulasikan sebagian besar perintah yang terdapat dalam sistem operasi linux, seperti perintah ‘ls’, ‘grep’, ‘cp’, dan ‘rm’. Mensimulasi perintah dengan seluruh opsi yang disediakan dalam perintah tersebut merupakan pekerjaan yang sangat melelahkan. Tetapi membuat program untuk mensimulasi perintah dapat meningkatkan ketrampilan dalam membuat program dan memanfaatkan fasilitas dalam sistem operasi. Mensimulasi perintah juga dapat membantu dalam meng-customize perintah standar dalam sistem operasi. Secara umum perintah-perintah yang berhubungan dengan file I/O dapat disemulasikan. Pada modul ini akan dibuat program untuk mensimulasi empat buah perintah yaitu ‘ls’, ‘grep’, ‘cp’ dan ‘rm’.

Peralatan

- Software: Sistem operasi Linux, dilengkapi dengan kompiler gcc, dan editor text (nano atau vi)
- Hardware: PC desktop/laptop
- koneksi internet (untuk akses compiler-online)

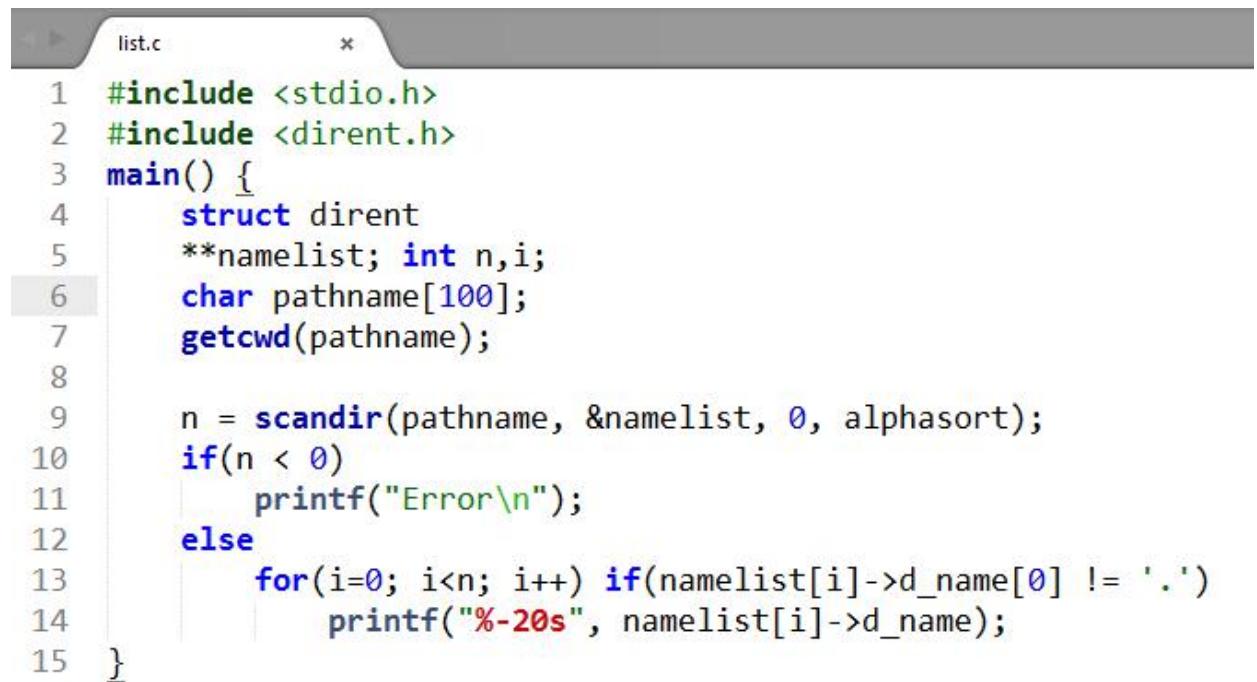
Langkah Kerja

A. Program untuk mensimulasi perintah ‘ls’.

Membuat kode program dengan algoritm sebagai barikut:

- (1) Menyimpan ‘path’ dari direktori kerja saat ini menggunakan perintah system call ‘getcwd’
- (2) Membaca isi direktori dari path di atas menggunakan perintah system call ‘scandir’ dan mengurutkan hasil pembacaannya dan menyimpannya dalam sebuah variabel array.
- (3) Menampilkan nama direktori (dname) dan nama file didalamnya jika file atau direktori tersebut tidak memiliki properti ‘HIDE’.
- (4) Stop

Kode Program



```

1 #include <stdio.h>
2 #include <dirent.h>
3 main() {
4     struct dirent
5         **namelist; int n,i;
6     char pathname[100];
7     getcwd(pathname);
8
9     n = scandir(pathname, &namelist, 0, alphasort);
10    if(n < 0)
11        printf("Error\n");
12    else
13        for(i=0; i<n; i++) if(namelist[i]->d_name[0] != '.')
14            printf("%-20s", namelist[i]->d_name);
15 }

```

A. Program untuk mensimulasi perintah ‘grep’.

Membuat kode program dengan algoritma sebagai berikut:

- (1) Gunakan nama file yang diberikan dalam argumen command-line
- (2) Buka file dalam mode ‘read-only’ menggunakan perintah system call ‘open’
- (3) Jika file tidak ada, keluar program, stop
- (4) Misal panjang string yang dicari adalah n .
- (5) Baca file perbaris sampai akhir file (END-OF-FILE), untuk setiap baris lakukan hal-hal berikut: (a) Periksa untuk mencari string dalam baris tersebut dengan dalam range 1-n, 2-n+1, dan seterusnya, (b) Jika string ditemukan tampilan baris tersebut di layar.
- (6) Tutup file menggunakan perintah ‘close’.
- (7) Stop

Kode Program

```

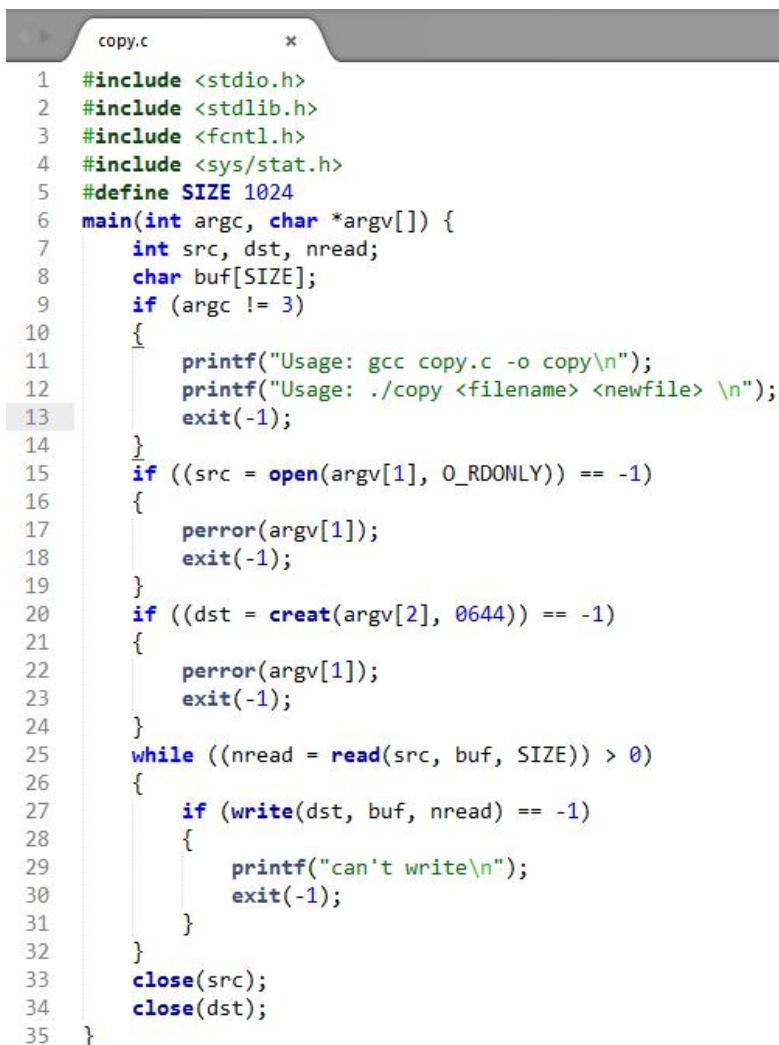
mygrep.c      *
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 main(int argc,char *argv[])
5 {
6     FILE *fd;
7     char str[100];
8     char c;
9     int i, flag, j, m, k;
10    char temp[30];
11
12    if (argc != 3)
13    {
14        printf("Usage: gcc mygrep.c -o mygrep\n");
15        printf("Usage: ./mygrep <search_text> <filename>\n");
16        exit(-1);
17    }
18
19    fd = fopen(argv[2],"r");
20    if(fd == NULL)
21    {
22        printf("%s is not exist\n",argv[2]);
23        exit(-1);
24    }
25
26    while(!feof(fd))
27    {
28        i = 0;
29        while(1)
30        {
31            c = fgetc(fd);
32            if(feof(fd))
33            {
34                str[i++] = '\0'; break;
35            }
36            if(c == '\n')
37            {
38                str[i++] = '\0'; break;
39            }
40            str[i++] = c;
41        }
42
43        if(strlen(str) >= strlen(argv[1]))
44        {
45            for(k=0; k<strlen(str)-strlen(argv[1]); k++)
46            {
47                for(m=0; m<strlen(argv[1]); m++)
48                {
49                    temp[m] = str[k+m];
50                    temp[m] = '\0';
51                    if(strcmp(temp,argv[1]) == 0)
52                    {
53                        printf("%s\n",str);
54                        break;
55                    }
56                }
57            }
58        }
59    }
60 }
```

C. Program untuk mensimulasikan perintah ‘cp’.

Membuat kode program dengan algoritm sebagai berikut:

- (1) Gunakan nama file untuk sumber dan tujuan dari argumen yang diberikan dalam command line.
- (2) Deklarasi sebuah buffer berukuran 1 KB
- (3) Buka file sumber dalam mode ‘read-only’ menggunakan fungsi ‘open’
- (4) Jika file sumber tidak ditemukan, stop keluar dari program
- (5) Membuat file baru sebagai file target dengan menggunakan perintah ‘creat’.
- (6) Jika proses pembuatan file gagal, stop keluar dari program.
- (7) Proses penyalinan (copy) file dilakukan dengan cara berikut: (a) Membaca 1KB data dari file sumber dan menyimpan hasilnya dalam buffer menggunakan perintah ‘read’. (b) Menuliskan isi buffer dalam file target menggunakan perintah ‘write’. (c) Jika bertemu dengan kode ‘END-OF-FILE’ lanjut ke nomor 8, yang lain kembali ke perintah (a)
- (8) Tutup file sumber dan target menggunakan perintah ‘close’.
- (9) Stop

Kode Program



```

copy.c

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <sys/stat.h>
5 #define SIZE 1024
6 main(int argc, char *argv[])
7 {
8     int src, dst, nread;
9     char buf[SIZE];
10    if (argc != 3)
11    {
12        printf("Usage: gcc copy.c -o copy\n");
13        printf("Usage: ./copy <filename> <newfile> \n");
14        exit(-1);
15    }
16    if ((src = open(argv[1], O_RDONLY)) == -1)
17    {
18        perror(argv[1]);
19        exit(-1);
20    }
21    if ((dst = creat(argv[2], 0644)) == -1)
22    {
23        perror(argv[2]);
24        exit(-1);
25    }
26    while ((nread = read(src, buf, SIZE)) > 0)
27    {
28        if (write(dst, buf, nread) == -1)
29        {
30            printf("can't write\n");
31            exit(-1);
32        }
33    }
34    close(src);
35    close(dst);
}

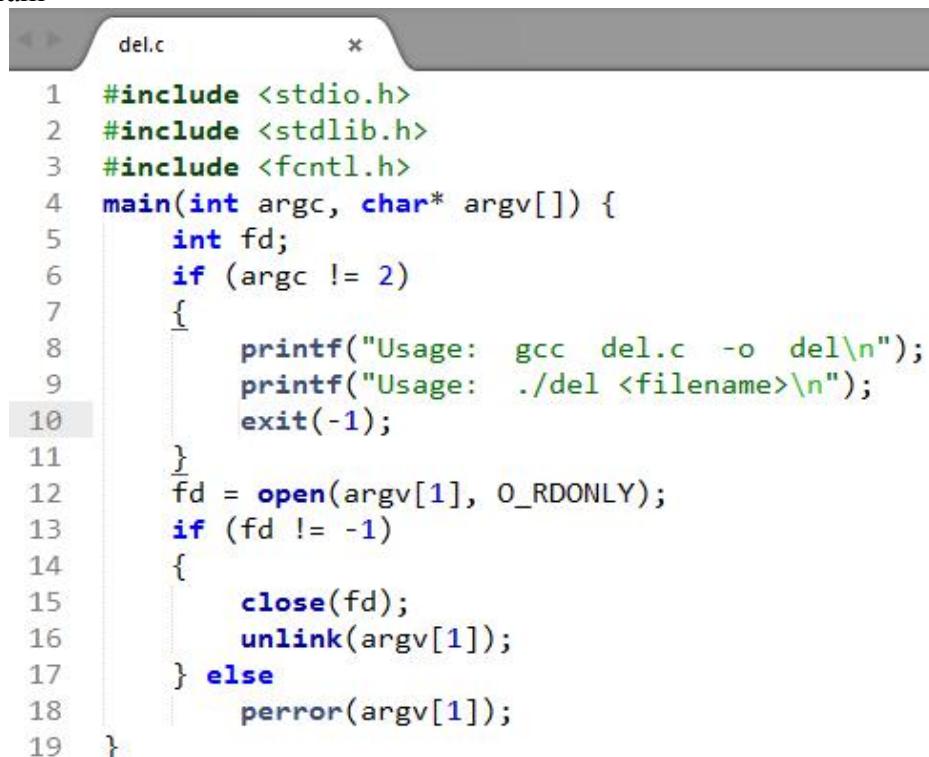
```

C. Program untuk mensimulasi perintah ‘rm’.

Membuat kode program dengan algorihm sebagai barikut:

- (1) Gunakan nama file yang diberikan dalam argumen command line
- (2) Buka file dalam mode ‘read-only’ menggunakan perintah ‘read’
- (3) Jika file tidak ditemukan, stop keluar program
- (4) Tutup file menggunakan perintah ‘close’
- (5) Menghapus file menggunakan perintah ‘unlink’
- (6) Stop Kode

Program



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 main(int argc, char* argv[]) {
5     int fd;
6     if (argc != 2)
7     {
8         printf("Usage: gcc del.c -o del\n");
9         printf("Usage: ./del <filename>\n");
10        exit(-1);
11    }
12    fd = open(argv[1], O_RDONLY);
13    if (fd != -1)
14    {
15        close(fd);
16        unlink(argv[1]);
17    } else
18        perror(argv[1]);
19 }

```

Hasil

Telah dibuat dan diperiksa empat program simulasi dengan menggunakan perintah dalam system-call, terdiri atas perintah ‘ls’, ‘grep’, ‘cp’ dan ‘rm’.

10.4. Kesimpulan

Perintah-perintah dalam system call, dapat digunakan untuk mensimulasi sebagian besar perintah dalam sistem operasi linux. Namun demikian mensimulasi sebuah perintah dengan semua opsi yang terdapat didalamnya merupakan pekerjaan yang melelahkan.

Modul 11 : Penjadwalan Proses dan Manajemen Memori (OS SIM)

Tujuan

- a. Mahasiswa mampu memahami proses penjadwalan suatu sistem operasi
- b. Mahasiswa mengenal dan menguasai berbagai macam jenis proses penjadwalan
- c. Mahasiswa mampu memahami proses manajemen memori
- d. Mahasiswa mampu mensetting parameter dalam manajemen memori

Pendahuluan

Pada praktikum ini akan dikenalkan beberapa hal penting yang terkait dengan algoritma penjadwalan proses dan proses manajemen memori. Adapun tool yang akan digunakan yaitu OS Simulator atau disingkat dengan OS Sim.

OS Sim merupakan aplikasi free yang berisi tentang simulasi sistem operasi (SO) secara grafis. Aplikasi ini bertujuan sebagai media pembelajaran yang berfokus pada empat komponen SO diantaranya penjadwalan proses, manajemen memori, manajemen file sistem dan penjadwalan disk



Penjadwalan Proses

Penjadwalan berkaitan dengan permasalahan memutuskan proses mana yang akan dilaksanakan dalam suatu sistem. Proses yang belum mendapat jatah alokasi dari CPU akan mengantre di *ready queue*. Algoritma penjadwalan berfungsi untuk menentukan proses manakah yang ada di *ready queue* yang akan dieksekusi oleh CPU.

Algoritma penjadwalan terdiri dari preemptive dan non-preemptive. Algoritma non-preemptive dibuat agar jika ada proses yang masuk ke running state, maka proses itu tidak bisa diganggu sampai menyelesaikan prosesnya, sedangkan penjadwalan preemptive menggunakan prioritas dimana jadwal dapat mengganggu proses dengan prioritas rendah ketika proses dengan prioritas tinggi sedang running.

Ada 4 algoritma yang akan dibahas yaitu:

a. **FCFS (First Come First Served)**

Algoritma ini merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status *ready* dimasukkan kedalam *FIFO queue* atau antrian dengan prinsip *first in first out*, sesuai dengan waktu kedadangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi.

Contoh:

Ada tiga buah proses yang datang secara bersamaan yaitu pada 0 ms, P1 memiliki burst time 24 ms, P2 memiliki burst time 3 ms, dan P3 memiliki burst time 3 ms. Hitunglah *waiting time* rata-rata dan *turnaround time* (*burst time* + *waiting time*) dari ketiga proses tersebut dengan menggunakan algoritma FCFS. *Waiting time* untuk P1 adalah 0 ms (P1 tidak perlu menunggu), sedangkan untuk P2 adalah sebesar 24 ms (menunggu P1 selesai), dan untuk P3 sebesar 27 ms (menunggu P1 dan P2 selesai).

P1	P2	P3
0	24	27

Urutan kedadangan adalah P1, P2 , P3; gantt chart untuk urutan ini adalah:

Waiting time rata-ratanya adalah sebesar $(0+24+27)/3 = 17$ ms. Turnaround time untuk P1 sebesar 24 ms, sedangkan untuk P2 sebesar 27 ms (dihitung dari awal kedadangan P2 hingga selesai dieksekusi), untuk P3 sebesar 30 ms. Turnaround time rata-rata untuk ketiga proses tersebut adalah $(24+27+30)/3 = 27$ ms.

b. **Shortest-Job-First(SJF)**

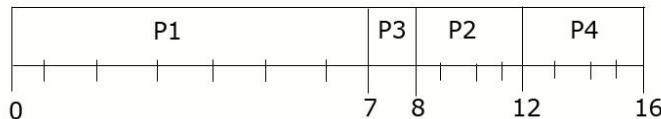
Pada algoritma ini setiap proses yang ada di *ready queue* akan dieksekusi berdasarkan *burst time* terkecil. Hal ini mengakibatkan *waiting time* yang pendek untuk setiap proses dan karena hal tersebut maka *waiting time* rata-ratanya juga menjadi pendek, sehingga dapat dikatakan bahwa algoritma ini adalah algoritma yang optimal.

Tabel 14.1. Contoh Shortest Job First

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Contoh: Ada 4 buah proses yang datang berurutan yaitu P1 dengan *arrival time* pada 0.0 ms dan *burst time* 7 ms, P2 dengan *arrival time* pada 2.0 ms dan *burst time* 4 ms, P3 dengan *arrival time* pada 4.0 ms dan *burst time* 1 ms, P4 dengan *arrival time* pada 5.0 ms dan *burst time* 4 ms. Hitunglah *waiting time* rata-rata dan *turnaround time* dari keempat proses tersebut dengan menggunakan algoritma SJF.

Average waiting time rata-rata untuk ketiga proses tersebut adalah sebesar $(0 + 6 + 3 + 7)/4 = 4$ ms.

Gambar 14.3. Shortest Job First (Non-Preemptive)

Average waiting time rata-rata untuk ketiga proses tersebut adalah sebesar $(9 + 1 + 0 + 2)/4 = 3$ ms

c. **Priority**

Priority Scheduling merupakan algoritma penjadwalan yang mendahulukan proses yang memiliki prioritas tertinggi. Setiap proses memiliki prioritasnya masing-masing.

Prioritas suatu proses dapat ditentukan melalui beberapa karakteristik antara lain:

1. Time limit.
2. Memory requirement.
3. Akses file.
4. Perbandingan antara burst M/K dengan CPU burst.
5. Tingkat kepentingan proses.

Priority scheduling juga dapat dijalankan secara preemptive maupun non-preemptive. Pada preemptive, jika ada suatu proses yang baru datang memiliki prioritas yang lebih tinggi daripada proses yang sedang dijalankan, maka proses yang sedang berjalan tersebut dihentikan, lalu CPU dialihkan untuk proses yang baru datang tersebut. Sementara itu, pada non-preemptive, proses yang baru datang tidak dapat menganggu proses yang sedang berjalan, tetapi hanya diletakkan di depan queue.

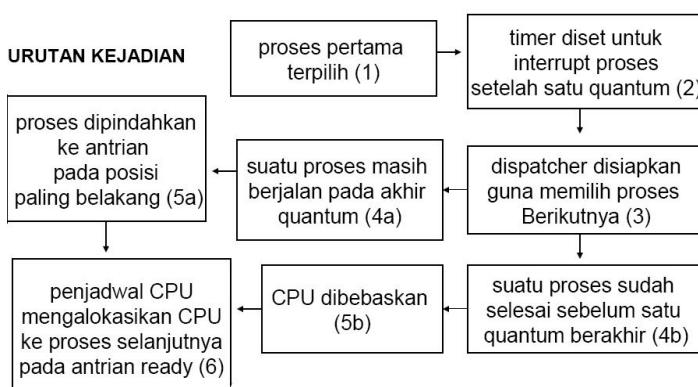
d. **Round Robin (RR)**

Algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar *time quantum*. Jika *time quantum*-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu $(1/n)$, dan tak akan menunggu lebih lama dari $(n-1)q$ dengan q adalah lama 1 *quantum*.

Algoritma ini sepenuhnya bergantung besarnya *time quantum*. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma *first come first served*. Jika terlalu kecil, akan semakin banyak peralihan proses sehingga banyak waktu terbuang.

Permasalahan utama pada *Round Robin* adalah menentukan besarnya *time quantum*. Jika *time quantum* yang ditentukan terlalu kecil, maka sebagian besar proses tidak akan selesai dalam 1 *quantum*. Hal ini tidak baik karena akan terjadi banyak *switch*, padahal CPU memerlukan waktu untuk beralih dari suatu proses ke proses lain (disebut dengan context switches time). Sebaliknya, jika *time quantum* terlalu besar, algoritma *Round Robin* akan berjalan seperti algoritma *first come first served*. *Time quantum* yang ideal adalah jika 80% dari total proses memiliki CPU burst time yang lebih kecil dari 1 *time quantum*.

Gambar 14.4. Urutan Kejadian Algoritma Round Robin



Manajemen Memori

Algoritma manajemen memori dibagi menjadi dua grup, yaitu:

- *Contiguous memory management*, yaitu seluruh proses dialokasikan ke memori. Untuk setiap proses, partisi yang sedang tidak dipakai dan cukup besar untuk mengakomodasi proses yang membutuhkan akan dipilih.

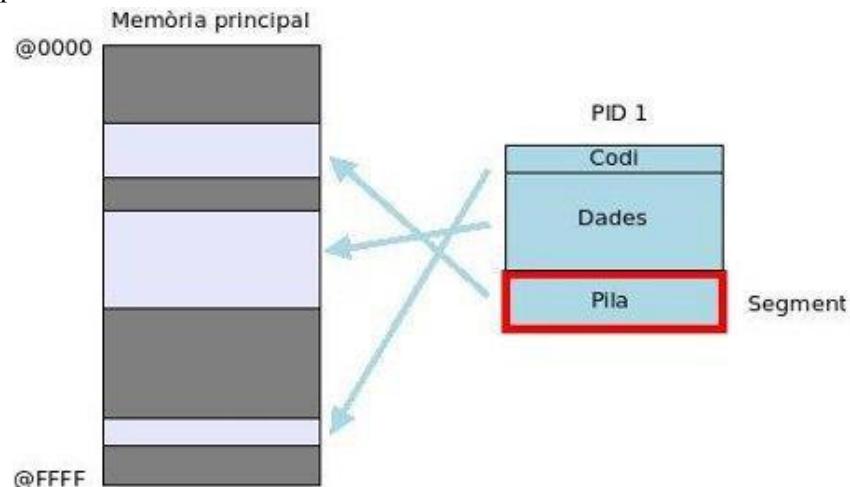
Terdapat dua algoritma manajemen memori:

- Partisi tetap (*fixed-sized partitions*), memori awalnya dibagi menjadi beberapa partisi dengan ukuran tetap (pengguna yang membuat). Pada umumnya proses yang dialokasikan kedalam sebuah partisi memiliki ukuran yang lebih kecil daripada ukuran partisinya, dan oleh karena itu ada sebagian ukuran partisi yang tidak terpakai, yang disebut sebagai *internal fragmentation*.
- Partisi tidak tetap (*variable-sized partitions*), memori awalnya merupakan sebuah partisi besar. Lalu untuk setiap proses akan dialokasikan ukuran partisi yang sesuai dengan ukuran proses, dan setelah selesai maka partisi tersebut dapat digunakan ulang. Partisi yang telah selesai digunakan tersebut dapat disebut sebagai *external fragmentation*.

Konsep lain yang menjadi perhatian adalah aturan alokasi memori, yaitu bagaimana sistem operasi memilih satu dari beberapa partisi yang tersedia. Beberapa diantaranya adalah:

- *First fit*, memilih partisi pertama kali yang dapat memuat proses.
- *Best fit*, memilih partisi yang dapat memuat proses secara optimal.
- *Worst fit*, memilih partisi yang dapat memuat proses dengan pemilihan paling buruk.
- *Non-contiguous memory management*, yaitu seluruh proses dibagi menjadi bagian-bagian yang dapat dialokasikan secara terpisah di memori, antara lain:
 - *Pagination*, memori dan proses dibagi menjadi bagian-bagian yang berukuran sama (frame dan page), dimana page dari proses akan dialokasikan pada frame dari memori.

- *Segmentation*, proses dibagi menjadi bagian-bagian secara logical (misalnya code, data, atau stack) dan masing-masing akan dialokasikan ke partisi memori secara independen.



Tampilan awal

Main memory

Proses-proses nantinya akan dialokasikan disini, baik itu secara keseluruhan maupun per bagian page atau segmen.

Secara definisi, pada setiap sistem manajemen memori, akan ada selalu terdapat satu proses pertama yang akan dialokasikan ke memori yang mendapatkan alamat dasar (base addresses), yaitu Operating System, yang ukurannya bervariasi dapat diantara 1, 2, atau 4 unit.

Alamat, proses, partisi yang tersedia maupun partisi yang sedang digunakan akan selalu terlihat, dan juga fragmentasi mengikuti kode warna yang memungkinkan pemahaman mahasiswa lebih cepat.

Warna	Keterangan
Abu-abu	Operating System
Putih	Memori belum digunakan
Titik biru	Fragmentasi internal
Titik merah muda	Fragmentasi eksternal
Warna lain...	Proses (masing-masing)

Ukuran memori dari simulasi ini bervariasi antara 64 sampai dengan 256 unit.

Klik kanan pada memori untuk menampilkan **pop up menu** yang memungkinkan untuk melakukan beberapa pilihan:

- Saat simulasinya berhenti (hanya untuk partisi tetap) >> untuk update dan delete partisi
- Saat waktu simulasinya untuk setiap proses yang dialokasikan ke memori >> delete proses, swap proses, melihat address translation, melihat informasi detail (hanya untuk pagination dan segmentation) dan untuk melakukan defragmentasi memori (hanya untuk partisi tidak tetap dan segmentation).

Process queue

Memuat proses yang tersedia untuk dialokasikan ke memori diurutkan berdasarkan waktu kedatangan proses.

Untuk setiap proses menunjukkan informasi yang relevan (PID, nama, dan durasi), disamping dari ukuran dan distribusinya, dimana setiap kotak bersesuaian ke satu unit space, yang dapat digunakan oleh satu alamat memori.

Pada pagination dan segmentation, proses-proses dibedakan menjadi komponen-komponennya, dan di-highlight oleh warna abu-abu untuk komponen yang belum dialokasikan ke memori tetapi dialokasikan ke swap area.

Saat simulasi berhenti, klik kanan pada sebarang proses untuk melihat pop up menu yang memungkinkan untuk meng-update dan delete.

Settings

- *Memory Size* (Ukuran memori, bervariasi antara 64 s/d 256 unit).
- *Operating System Size* (Ukuran memori untuk OS, bervariasi dari 1, 2 atau 4 unit).
- Algorithm:
 - *Fixed-sized partitions* (contiguous memory management) >> required untuk mempartisi seluruh memori.
 - *Variable-sized partitions* (contiguous memory management).
 - *Pagination* (non-contiguous memory management) >> harus menentukan ukuran page sistem (antara 1, 2 atau 4).
 - *Segmentation* (non-contiguous memory management).
- Allocation policy, memilih algoritma pemilihan partisi, yaitu:
 - *First fit*
 - *Best fit*
 - *Worst fit*

Perubahan pada algoritma, ukuran memori, atau ukuran OS membutuhkan untuk me-restart simulasi dan menghapus seluruh proses yang ada.

Menambah proses

Proses hanya dapat ditambahkan lagi simulasi berhenti, untuk setiap proses baru dapat ditambahkan dengan cara berikut:

- *PID* >> proses identifier, bersifat *automatically calculated* dan *incremental*.
- *Name (required)* >> nama proses.
- *Size* >> ukuran proses (nilai: 1 s/d 64 unit).
- *Duration* >> proses dapat mempunyai durasi tak terhingga (diisi -1) atau terhingga (1-100), tapi tidak boleh 0.
- *Color* >> warna yang akan digunakan untuk menandai proses yang dimaksud.

Tambahan untuk pagination:

- *Page table* >> jumlah page yang tergantung pada ukuran proses, untuk setiap page mahasiswa harus menentukan apakah sudah dialokasikan ke memori secara langsung (initially allocated) atau tidak (swapped).

Tambahan untuk segmentation:

- *Segment table* >> setiap proses mempunyai tiga segmen: code, data, dan stack. Untuk setiap segmen, mahasiswa harus menentukan ukuran dan apakah sudah dialokasikan ke memori (initially allocated) atau tidak (swapped). Ukuran dari jumlahan segmen harus sama dengan ukuran dari proses.

Data dan statistik

Menunjukkan informasi alokasi memori pada setiap waktu simulasi. Informasi ini beragam antara satu algoritma dengan algoritma yang lain.

Contiguous memory management, tabel dengan informasi berikut:

- *Direction*, alamat partisi mula-mula.
- *Size*, ukuran dari partisi.

(jika pada partisi terdapat sebuah proses)

- *PID*, proses ID (warna background menunjukkan warna proses).
- *Name*, nama proses.
- *Size*, ukuran proses.
- *Duration*, durasi proses.

Pada pagination, tabel dengan informasi berikut:

- *Address*, alamat frame mula-mula.
- *Frame*, nomor frame.

(jika partisi sedang digunakan)

- *PID*, proses ID (warna background menunjukkan warna proses).
- *Page*, proses yang menggunakan frame yang bersesuaian.
- *Name*, nama proses.
- *Dimensions*, total proses.
- *Duration*, durasi proses.

Pada segmentation, tabel dengan informasi berikut:

- *Address*, alamat frame mula-mula.
- *Size*, ukuran partisi.

(jika pada partisi terdapat sebuah proses)

- *PID*, proses ID (warna background menunjukkan warna proses).
- *Segment*, segmen proses, yaitu: code, data atau stack.
- *Name*, nama proses.
- *Size*, ukuran proses.
- *Duration*, durasi proses.

Tabel page dan tabel segment

Menunjukkan informasi alokasi memori dari sebuah proses pada non-contiguous memory management.

Pada pagination, konten tabel adalah sebagai berikut.

- *Page*, proses yang menggunakan frame yang bersesuaian.

- *Frame*, nomor frame.
- *Valid*, bit valid dari page (v valid, i invalid). Sebuah page dikatakan tidak valid ketika page tersebut tidak dialokasikan ke memori.

Pada segmentation, konten tabel adalah sebagai berikut.

- *Segment*, segmen dari proses: code, data, atau stack.
- *Size*, ukuran dari segmen.
- *Address*, alamat frame mula-mula.
- *Valid*, bit valid dari segmen (v valid, i invalid). Sebuah segmen dikatakan tidak valid ketika segmen tersebut tidak dialokasikan ke memori.

Membuat partisi (untuk partisi berukuran tetap)

Menentukan alamat mula-mula dan ukuran partisi.

Partisi tidak bisa saling overlap, dan tidak bisa melebihi batas akhir memori.

Defragment memori (untuk partisi berukuran tidak tetap dan segmentation)

Paritisi berukuran tidak tetap dan segmentation mempunyai kesamaan pada partisi yang dibuat secara dinamis dan diukur secara otomatis untuk mengakomodasi proses atau segmen secara tepat. Operating System memilih sebuah partisi yang sama atau sedikit lebih besar dan dibagi menjadi dua bagian, satu untuk mengalokasikan proses atau segmen, satu untuk yang masih tersedia.

Setelah selesai digunakan (proses sudah selesai atau swapped out), partisi tersebut menjadi kosong dan berubah menjadi lebih kecil (memorinya ter-degradasi).

Proses defragmentasi dapat menanggulangi untuk me-reorganize memori, mengatur semua proses menjadi menggunakan memori dengan alamat rendah (lower memory addresses) dan menggabungkan seluruh partisi kosong menjadi satu partisi kosong yang besar.

Ini adalah tanggung jawab dari OS, dan ini diserahkan kepada mahasiswa untuk simulasi (pop up menu dari memori).

Swap

Melakukan swap dapat memungkinkan kita untuk menjalankan proses yang lebih banyak dari kapasitas memori utama, dan biasanya diimplementasikan pada memori sekunder.

Hal ini muncul atas dasar untuk mempunyai space ekstra untuk memindah proses atau bagian darinya (page atau segmen) yang sedang kurang aktif dan mengalokasikan ulang ke memori saat sedang dibutuhkan.

Simulasi digunakan untuk menyederhanakan proses ini, mensimulasikan swap in dan swap out antar memori dan swap dilakukan oleh mahasiswa secara manual.

Swap out, dari pop up menu memori (klik kanan pada memori). Melakukan swap out sebuah proses atau bagian darinya.

Swap in, dari pop up menu memori (klik kanan pada memori). Mencoba untuk melakukan swap in pada sebuah proses atau bagian darinya kepada memori, jika memori penuh akan menghasilkan sebuah error dan tidak akan bisa swap in.

Algoritma pagination dan segmentation memungkinkan untuk melakukan swap out page atau segmen pada awal (initial). Area ini tidak dibatasi pada jumlah atau ukuran dari proses yang dimuat.

Error pada simulasi

1. **Memory full**, karena sedang memproses⁹⁸ antrian proses pada simulasi, memori mengalokasikan untuk proses dan selesai mengalokasikannya. Saat proses tidak bisa dialokasikan karena kurang space-nya, maka menyebabkan tidak dapat dilanjutkan ke proses

selanjutnya. Memori dapat dilepaskan (released) secara otomatis (karena proses sudah selesai menggunakan memori) atau oleh campur tangan user (delete atau swap out proses).

2. **Memory not fully partitioned**, algoritma paritisi berukuran tetap (fixed-size partition) membutuhkan semua memori untuk dipartisi (alamat akhir satu partisi adalah alamat awal partisi yang lain).

Peralatan

- Software: OSSIM
- Hardware: PC desktop/laptop
- Modul Praktikum

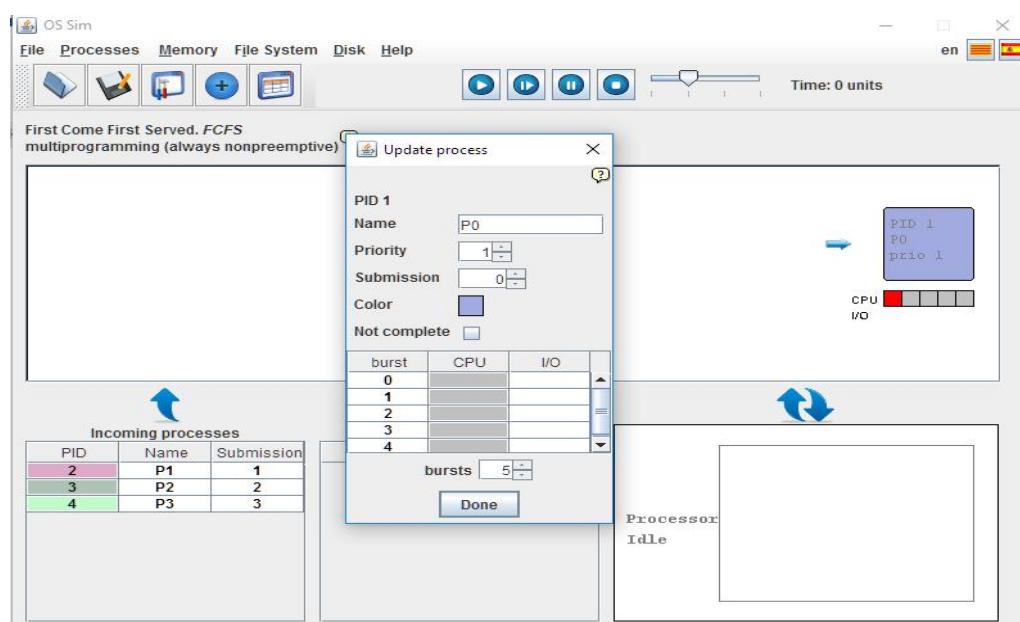
Langkah Kerja

Kegiatan 1. Penjadwalan Proses

1.1 First-Come, First-Served (FCFS)

- a. Buka program OSSim, selanjutnya pilih menu processes -> process scheduling
- b. Selanjutnya pilihlah setting dan pilih algoritma **First-Come, First-Served (FCFS)**
- c. Lakukan input proses sesuai dengan tabel berikut dengan memulai dengan P0 sebagai input proses yang pertama.

Process	Arrival Time	Burst Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



- d. Jika input sudah selesai dilakukan. Pilih tombol start pada bagian atas. Amati dan analisa proses yang terjadi.
- e. Isilah tabel berikut

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.2 Shortest Job First (SJF)

- Bukalah program OS Sim, selanjutnya pilih menu processes -> process scheduling
- Selanjutnya pilihlah setting dan pilih algoritma **Shortest Job First (SJF)**. algoritma ini terdiri dari 2 jenis yaitu non-preemptive dan preemptive. Untuk mengaktifkan preemptive dengan mencentang menu tersebut. Sebaliknya jika menonaktifkan maka hanya cukup menghilangkan centangnya saja.
- Selanjutnya klik tombol start. Amati dan analisa proses yang terjadi. Lakukan perbandingan dari hasil keduanya.
- Isilah tabel berikut:

Non-Preemptive

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

Preemptive

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.3 Priority

- Pilihlah menu setting dan pilih algoritma **Priority**. Selanjutnya tambahkan priority pada setiap proses.

Process	Arrival Time	Burst Time	Priority	Service Time

P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

- b. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi. Lengkapilah tabel berikut!

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

1.4 Round Robin

- a. Pilihlah menu setting dan pilih algoritma **Round Robin**. Selanjutnya tambahkan quantum time sebesar 3.
- b. Selanjutnya klik tombol start. Lakukan pengamatan dan analisa proses yang terjadi. Lengkapilah tabel berikut:

Process	Wait time : Service Time – Arrival Time
P0	
P1	
P2	
P3	
Av wait time	

Kegiatan 2: Manajemen Memori

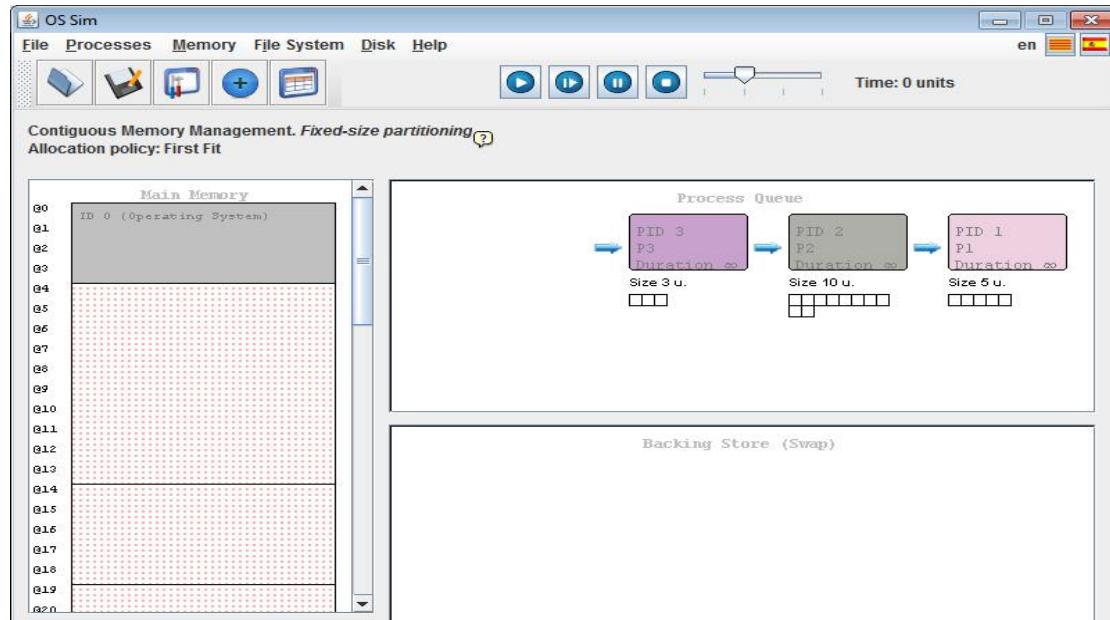
2.1 Contiguous memory management dengan menggunakan partisi berukuran tetap (fixed-size partition) dan aturan first fit

Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan first fit, partisi pertama yang dapat memuat proses akan dipilih.

Partisi		Proses	
Alamat awal	Unit ukuran	Proses	Unit ukuran
@4	10	P1	5
@14	5	P2	10
@19	4	P3	3
@23	15		
@38	10		
@48	14		

Langkah:

- Pilih menu help >> examples...>> memory management >> Contiguous memory management with fixed-size partitions (first adjustment). Pilihlah folder hingga menampilkan tampilan berikut.



- Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
- Lakukan analisa proses dengan melakukan klik pada tombol next.

2.2 Contiguous memory management dengan menggunakan partisi berukuran tetap (fixed-size partition) dan aturan best fit

Memori dibagi menjadi partisi-partisi berukuran tetap, lalu seluruh proses akan dialokasikan pada satu dari partisi yang tersedia. Pada aturan best fit, partisi terbaik (yang ukurannya sama atau hampir sama) yang dapat memuat proses akan dipilih.

Partisi		Proses	
Alamat awal	Unit ukuran	Proses	Unit ukuran
@4	10	P1	5
@14	5	P2	10
@19	4	P3	3
@23	15		
@38	10		
@48	14		

Langkah:

1. Pilih menu help >> examples...>> memory management >> Contiguous memory management with fixed-size partitions (best fit)>> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.3 Contiguous memory management dengan menggunakan partisi berukuran tidak tetap (variable-size partition) >> defragmentasi

Memori pertama tidak dipartisi, lalu partisi akan dibuat saat proses dialokasikan dan mempunyai ukuran yang sama dengan ukuran proses. Selanjutnya partisi akan dibebaskan (saat proses sudah selesai menggunakan memori) dan digunakan oleh proses lain. Jika ukuran proses lebih kecil dari ukuran partisi, maka partisi akan dibagi menjadi dua bagian, bagian pertama sama besarnya dengan proses dan bagian kedua adalah sisanya.

Proses	Unit ukuran	Durasi
P1	5	5
P2	8	4
P3	3	3
P4	15	∞
P5	2	1
P6	20	∞
P7	3	∞
P8	10	∞

Langkah:

1. Pilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Defragmentation) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.4 Contiguous memory management dengan menggunakan partisi berukuran tidak tetap (variable-size partition) >> swap

Memori pertama tidak dipartisi, lalu partisi akan dibuat saat proses dialokasikan dan mempunyai ukuran yang sama dengan ukuran proses. Selanjutnya partisi akan dibebaskan (saat proses sudah selesai menggunakan memori) dan digunakan oleh proses lain. Jika ukuran proses lebih kecil dari ukuran partisi, maka partisi akan dibagi menjadi dua bagian, bagian pertama sama besarnya dengan proses dan bagian kedua adalah sisanya.

Memori mempunyai ukuran yang berhingga, hanya beberapa proses saja yang bisa dimuat di memori pada saat yang sama, tetapi space swap memungkinkan kita untuk menyimpan beberapa proses yang sedang kurang aktif, sehingga space pada memori dapat dialokasikan untuk proses yang lain yang lebih aktif. Proses yang telah di swap out akan dialokasikan kembali ke memori saat dibutuhkan.

Proses	Unit ukuran
P1	5
P2	8
P3	3
P4	15
P5	2

P6	20
P7	3
P8	10

Langkah:

1. Pilih menu help >> examples...>> memory management >>Contiguous memory management with variable-sized partitions (Swap) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.5 Pagination (ukuran page 2 unit)

Proses akan dibagi menjadi beberapa page dengan ukuran tetap (contohnya adalah 2 unit), memori lalu dibagi menjadi beberapa frame berukuran sama. Page dari proses lalu dialokasikan pada frame memori yang kosong.

Proses	Ukuran	Durasi
P1	5 unit (3 page)	4
P2	4 unit (2 page)	∞
P3	11 unit (6 page)	2
P4	5 unit (3 page)	∞
P5	7 unit (4 page)	∞
P6	3 unit (2 page)	∞

Langkah:

1. Pilih menu help >> examples...>> memory management >>Pagination) >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

2.6 Segmentation (alokasi parsial)

Proses dibagi menjadi beberapa segmen, tiap segmen bisa mempunyai ukuran yang berbeda. Segmen ini akan dialokasikan ke memori secara independen, lalu partisi akan dibuat untuk menampung segmen tersebut dengan ukuran yang sama. Pembagian ini adalah secara fungsional dan berdasarkan pada logical structure dari proses tersebut (data, stack, code, dll).

Tidak semua segmen harus dimuat pada memori agar proses dapat berjalan, selama sebuah segmen tidak dipakai maka dapat di-swap out.

Proses	Ukuran total	Segmen	Ukuran (unit)	Dialokasikan diawal
P1	20	Code	2	Yes
		Data	10	Yes
		Stack	8	No
P2	20	Code	5	Yes
		Data	14	No
		Stack	1	Yes
P3	40	Code	10	Yes
		Data	20	Yes
		Stack	10	Yes

Langkah:

1. Pilih menu help >> examples...>> memory management >>Segmentasi >> pilih folder
2. Selanjutnya klik tombol play untuk melihat proses yang terjadi dan klik tombol stop untuk berhenti .
3. Lakukan analisa proses dengan melakukan klik pada tombol next.

Tugas:

1. Jawab pertanyaan pada latihan OS-SIM scheduling algorithm performance comparison
2. Jawab pertanyaan pada latihan OS-SIM management memory

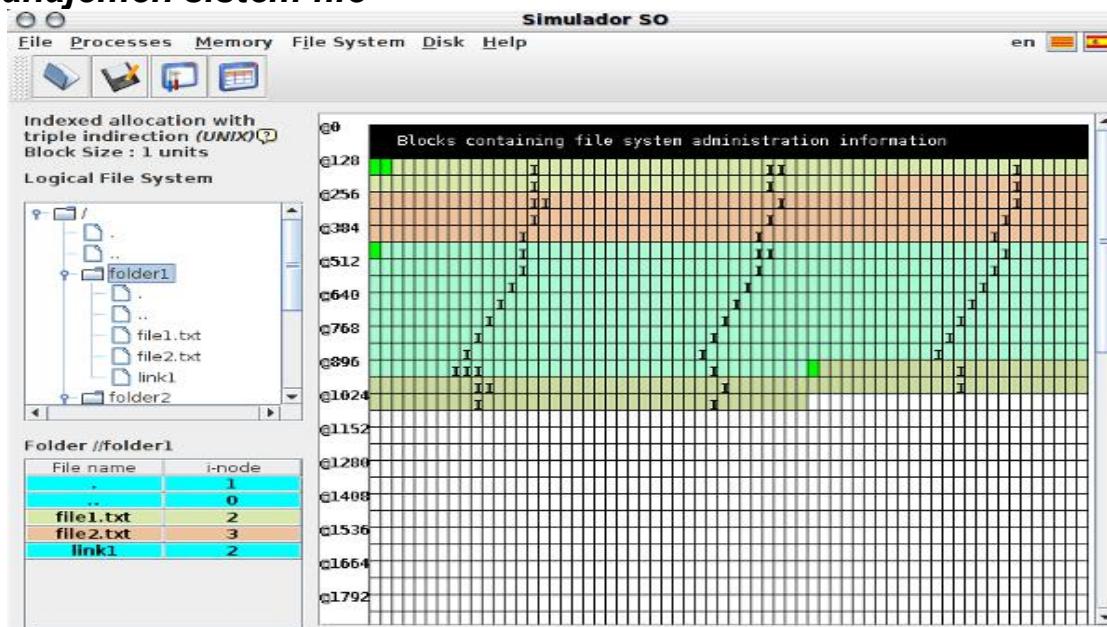
Tujuan

- Mahasiswa mampu memahami proses manajemen file system suatu sistem operasi
- Mahasiswa mengenal dan menguasai berbagai macam jenis proses penjadwalan
- Mahasiswa mampu memahami proses penjadwalan disk
- Mahasiswa mampu mensetting parameter dalam manajemen memori

Pendahuluan

Pada modul ini akan dipelajari beberapa hal penting yang terkait dengan manajemen sistem file dan algoritma penjadwalan suatu disk.

Manajemen sistem file



Sistem file mengatur hubungan antara logical space sebagai pengguna melihat (file, direktori, link) dengan struktur yang sesuai dan lokasi fisik (nyata) obyek dalam memori sekunder.

Unit kerja minimal sistem file adalah blok, memori dan sistem file objek dibagi menjadi blok dengan ukuran yang sama, dan sistem manajemen adalah untuk mengalokasikan blok-blok objek ke dalam blok yang tersedia pada memori sekunder dan menjaga kontrol.

Simulasi menerapkan dua teknik utama untuk Manajemen File System:

- Alokasi Link dengan tabel alokasi file (FAT, MS-DOS environments).
- Alokasi Index dengan tiga tingkat tidak berarah (sistem UNIX).

Parameter simulasi adalah:

- Ukuran blok (1, 2 or 4 unit).
- Total ukuran disk (4, 6 or 8 Mega Unit).

128 blok disk pertama tidak digunakan untuk mensimulasikan tugas-tugas administratif OS dan tidak digunakan.

Dua algoritma Manajemen File System adalah:

- **Alokasi dilink dengan tabel alokasi file**, sistem mempertahankan tabel dengan entri sebanyak blok yang dimiliki sistem ((File Allocation Table, FAT), untuk setiap objek hanya referensi ke blok pertama yang disimpan. Referensi ini berada di folder yang berisi objek.

Di tabel setiap blok memiliki referensi ke blok berikutnya sehingga mereka bisa pergi bersama secara blok file berantai dari awal sampai akhir.

Ukuran tabel FAT = ukuran disk / ukuran blok

Blok yang dibutuhkan untuk setiap file = ukuran file / ukuran blok

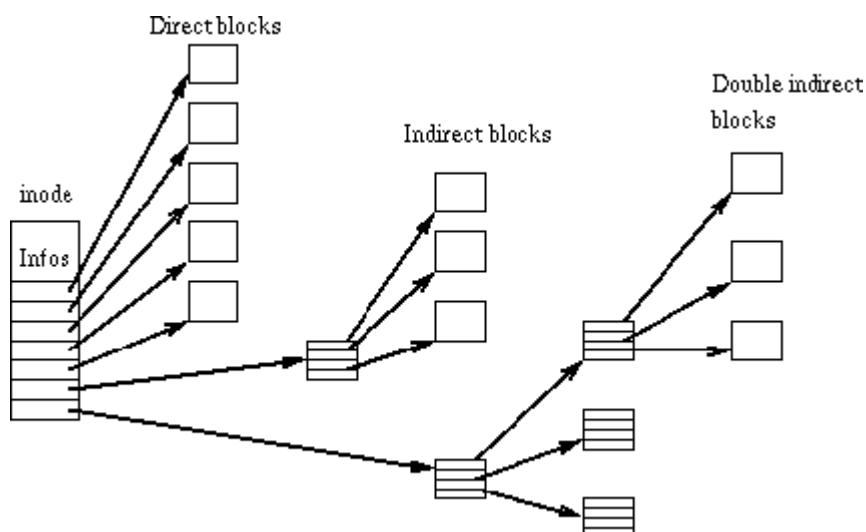
Untuk mempermudah dalam simulasi folder menempati hanya satu blok, dan link tidak (poin ke item yang ada dan informasi yang hanya disimpan di direktori yang mengandung link)

- **Alokasi diindex dengan tiga tingkat tidak berarah**, alokasi diindeks didasarkan pada struktur yang disebut i-node, dan masing-masing objek terkait dengan i-node. Struktur dari i-node adalah:

- information fields.
- 12 pointers to data blocks.
- 1 pointer to 1 indirect block.
- 1 pointer to 1 double indirect block.
- 1 pointer to 1 triple indirect block.

Titik blok tunggal tidak langsung untuk blok data, titik blok ganda tidak langsung untuk blok yang pada titik gilirannya untuk blok data titik blok triple menunjuk ke blok saat itu untuk memblokir yang mengarah ke blok data.

Untuk membuat file baru pertama blok data i-node yang digunakan, maka blok tunggal tidak langsung, double dan triple untuk mengisi informasi yang diperlukan tergantung pada ukuran file. Untuk penyederhanaan folder menempati blok (data), dan link tidak.



Dalam simulasi jumlah maksimum i-node yang dapat dibuat adalah 128, blok tidak langsung memiliki 20 pointer. Dalam tabel di bawah ini menunjukkan bagaimana menghitung blok yang dibutuhkan untuk membuat sebuah file.

	Available blocks	Data	Administration Blocks	Total Blocks	data	Total Blocks	Adm.	Total Blocks
One node	i-12 blocks		Head	12	0			12
First indirection	Past more 20 blocks	...	1 indirect block	32	1			33
Second indirection	Past more 20 * 20 blocks	...	1 indirect block 1 +20 Indirect b.	432	22			454
Third indirection	Past more 20 * 20 * 20 blocks	...	1 indirect block 1 +20 Indirect b. 1 +20 +400 Indirect b.	8.432	433			8.875

Sistem file Logical:

Menampilkan sistem file seperti yang terlihat oleh pengguna, pohon folder, dan file, link dan subfolder dalam setiap folder. Root tergantung pada manajer sistem file, UNIX adalah "/" MD-DOS adalah "C:\".

Klik kanan pada objek apapun untuk menampilkan menu pop up yang memungkinkan anda melakukan berbagai tindakan:

- Tambahkan sebuah file pada folder objek yang sama.
- Tambahkan sebuah subfolder pada folder objek yang sama.
- Tambahkan sebuah link pada folder objek yang sama.
- Update objek.
- Hapus objek.

(Device) Sistem file Physical:

Menunjukkan perangkat (sistem file fisik) dibagi menjadi blok, dan pada alamat disk yang tepat. Alamat awal (128 unit) dicadangkan untuk tugas-tugas administrasi OS dan tidak digunakan.

Blok diberi warna dengan warna yang berbeda tergantung pada jenis:

Colors:	Block use
black	Administration (File System)
white	Available
Green	Folder
Blue	Soft link
[Other colors]	Occupied by a file UNIX "I" means it is an indirect block

Melihat folder terpilih:

Tampilan ini tergantung pada folder yang dipilih sistem file logis menampilkan isi dari folder ini, setiap objek digambar dengan warna yang sesuai: berkas (file bervariasi sesuai dengan warna), link (biru), folder (hijau).

Untuk setiap objek itu menunjukkan referensi ke sistem file fisik. Untuk i-node UNIX objek, untuk MS-DOS awal entri dalam tabel FAT.

Klik kanan pada setiap objek sistem berkas untuk menampilkan menu pop up yang memungkinkan Anda untuk melihat informasi rinci objek.

Pengaturan.

Mengatur perilaku simulasi ini:

- *Block Size* (Values 1, 2 or 4 pieces) (Values between 64 and 256 units).
- *Memory size* (values 4, 6 or 8 x 1024 units).
- Algoritma:
 - Alokasi di link dengan file allocation table (*FAT, MS-DOS*).
 - Alokasi di index dengan triple indirection (*UNIX*).

Setiap perubahan algoritma, ukuran blok atau ukuran memori membutuhkan restart simulasi dan menghapus semua objek yang ada.

Menambah sebuah file:

File tersebut akan ditambahkan ke folder yang sedang dipilih, untuk setiap file baru anda harus memasukkan informasi berikut:

- Nama (*dibutuhkan*)teks apapun.
- Ukuran file (Range: 1 to 4096 units).
- Warna, warna yang akan menarik elemen grafis yang terkait dengan file tersebut.

Menambah sebuah folder:

Folder ditambahkan ke folder yang sedang dipilih,

Hal ini hanya diperlukan untuk menunjukkan nama folder baru (diperlukan).

Menambah sebuah link:

Link tersebut ditambahkan ke folder yang sedang dipilih untuk setiap link baru perlu untuk memasukkan informasi berikut:

- Nama (*dibutuhkan*)teks apapun.
- *Soft*, link type, soft link (terpilih) atau hard link (tidak terpilih). MS-DOS hanya membolehkan soft links
- *Target*, link target.

Informasi rinci dari sebuah objek:

Dari tampilan folder, pengguna dapat membuka informasi rinci dari objek apapun dalam folder, informasi ini tergantung pada sistem file.

UNIX menunjukkan struktur i-node:

- Informasi, dalam hal ini hanya jumlah link yang menunjuk ke i-node.
- 12 blok data, jumlah blok di mana data tersebut, atau "nihil" jika tidak digunakan.

- Indirections, ke-1, ke-2 dan ke-3, nomor blok dimana blok tidak langsung, atau "nihil" jika tidak digunakan.
 - Untuk setiap indirection juga bisa mendapatkan informasi konkret:
 - Index pointer.
 - Nomor blok Pointed, atau "nihil" jika tidak digunakan.
 - Type: "blok" jika data atau "indirection" jika tidak langsung.

Untuk MS-DOS menampilkan entri tabel FAT untuk objek:

- Nomor blok (FAT tabel entry).
- Objek selanjutnya nomor blok atau "nihil" jika terakhir.

Data dan statistik:

Menampilkan informasi perangkat. Informasi ini bervariasi dari satu algoritma yang lain. Dalam UNIX, menunjukkan tabel dengan semua blok perangkat data, untuk setiap blok:

- Blok, nomor blok (dimulai dari data blok pertama, @128).
- Tipe "data blok" atau "indirect blok."

Dalam MS-DOS, menunjukkan tabel FAT, yang merupakan snapshot dari semua blok perangkat, setiap entri tabel berisi informasi berikut:

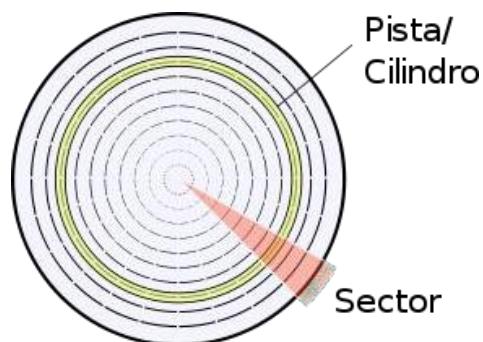
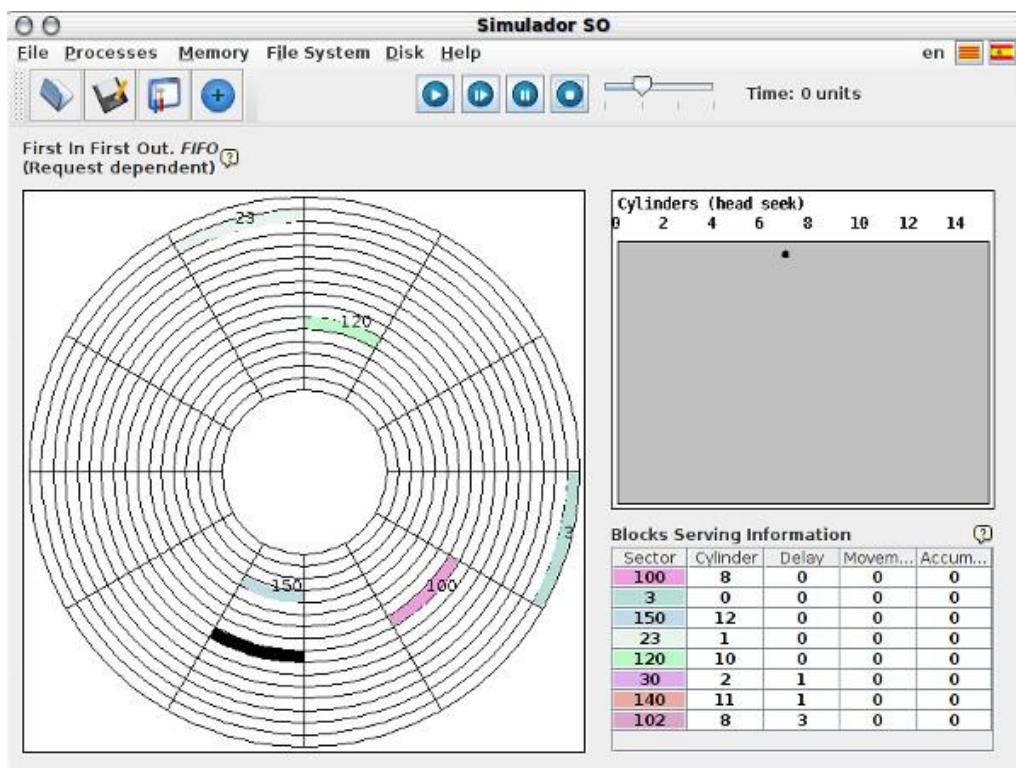
- Blok, nomor blok (dimulai dari data blok pertama, @128).
- Next, nomor blok objek berikutnya ketika blok berisi data obyek, "nihil" jika blok berisi blok terakhir dari suatu obyek atau 0 jika tersedia.
- Kondisi dari blok, "used" atau "free."

Kesalahan dalam simulasi:

1. **Memori penuh**, sebagai objek baru yang dibuat blok perangkat ditempati tetapi ada batasan jumlahnya. Jika tidak mungkin untuk membuat objek baru karena kurangnya ruang perlu untuk mengurangi ukuran yang sudah ada atau menghapusnya.
2. **Ada objek lain dengan nama yang sama** dalam direktori tidak mungkin ada beberapa objek (file, link atau subdirektori) dengan nama yang sama.

Penjadwalan Disk

Berikut adalah kebijakan penjadwalan disk di dalam Sistem Operasi I / O manajemen perangkat.



Meskipun heterogenitas mereka, optical disk atau magnet, kaku atau fleksibel, read-only atau membaca / menulis CD, DVD atau HHD misalnya, keseluruhan dapat diperlakukan sama.

Disk diatur dalam sector (**sector**), mereka dikelompokkan ke dalam trek(**tracks**), trek yang di satu sisi atau permukaan di piring, dan setiap disk dapat memiliki beberapa piring-piring (**platters**).

Himpunan trek yang berada dalam posisi kepala yang sama disebut silinder. Piring-piring berputar terus-menerus dan masing-masing memiliki satu kepala tepat di atas, yang membaca dan menulis sektor, kepala bergerak lurus ke trek.

Sektor disk yang direferensikan berurutan, sektor pertama adalah 0, dan itu adalah di lagu pertama dari piring pertama dalam silinder luar.

Dalam geometri simulasi disk: piring, 16 silinder, 12 sektor per silinder, benar-benar 192 sektor atau blok data.

Sistem Operasi terus menerima permintaan untuk menulis dan membaca dari disk, bagaimana melayani permintaan tersebut ditentukan oleh kebijakan penjadwalan disk.

Sebuah kebijakan penjadwalan yang baik harus meminimalkan gerakan head, yang berarti kehidupan yang lebih panjang untuk perangkat, dan permintaan-to-respon waktu.

Dalam simulasi ini ada tersedia kebijakan penjadwalan berikut:

- **FIFO.** *Fist In First Out* penjadwalan ini cukup sederhana, melayani permintaan agar tepat dari kedatangan. Head selalu bergerak mencari permintaan berikutnya di mana pun itu, jangan mencoba untuk meminimalkan gerakan head tapi itu benar karena tidak memprioritaskan permintaan apapun.
- **LIFO.** *Last In First Out* tidak seperti FIFO, melayani permintaan dalam urutan terbalik kedatangan, yang terakhir datang pertama dilayani. Permintaan terbaru dilayani segera, tetapi dapat menyebabkan tidak dilayani, permintaan pertama tidak akan pernah dilayani jika yang baru tiba terus-menerus.
- **STF.** *Shortest (Seek) First* selalu melayani permintaan terdekat pertama yang posisi head ini. penjadwalan seperti meminimalkan gerakan kepala, tetapi juga dapat menyebabkan tidak dilayani.
- **SCAN.** Juga disebut "Elevator", head bergerak dari silinder terluar untuk terdalam ketika mencapai akhir itu berubah arah dan kembali ke awal, dan seterusnya tanpa batas. Menangani permintaan ketika mereka muncul selama gerakan.
- **C-SCAN.** *Circular SCAN*. Sama dengan SCAN, Perbedaannya adalah bahwa ketika mencapai silinder bagian di disk akhir ini, bergerak langsung ke awal dan dimulai lagi, jadi selalu menangani permintaan dalam arah yang sama. Meningkatkan salah satu kelemahan SCAN, yang berjalan di atas track yang baru saja dilihat ketika perubahan arah.
- **LOOK.** Adalah variasi yang meningkatkan penjadwalan SCAN, perbedaannya adalah bahwa hal itu hanya naik ke permintaan terakhir dan tidak sampai silinder terakhir. Pada saat itu mendeteksi bahwa tidak ada lagi permintaan ke depan, maka perubahan arah.
- **C-LOOK.** *Circular LOOK*. Termasuk kedua LOOK dan perbaikan C-SCAN hanya pergi ke permintaan terakhir dan juga saat mencapai luar, bergerak ke yang pertama dan mulai lagi, itu selalu melayani permintaan dalam arah yang sama.

Disk:

Menunjukkan sektor dan silinder divisi disk. Sektor 0 (pertama) adalah di luar silinder, atas dan kanan vertikal. Setiap saat simulasi posisi kepala ditampilkan hitam, dan permintaan dalam warna yang sesuai Sementara simulasi dihentikan, klik kanan pada permintaan untuk menampilkan menu pop up yang memungkinkan Anda untuk mengubah dan menghapus.

Grafik gerakan head:

grafik mewakili gerakan head melalui silinder sebagai permintaan dilayani. Dalam silinder sumbu x dan ordinat waktu. Setiap permintaan disajikan ditandai dengan titik (warna yang sesuai) menunjukkan silinder di mana sektor ini disajikan dan waktu, poin bergabung dengan garis-garis yang mewakili gerakan head melalui silinder.

Data dan statistik:

Lihat semua permintaan, informasi dalam tabel adalah sebagai berikut:

- *Sector*, jumlah sektor yang diminta.
- *Cylinder*, di mana sektor ini.
- *Delay*, waktu yang dibutuhkan untuk permintaan tiba. Misalnya, permintaan 3 unit tertunda, masuk ke dalam antrian permintaan pada simulasi waktu 3.
- *Movement*, hanya untuk permintaan dilayani, menunjukkan gerakan head sejak permintaan sebelumnya, yaitu jumlah total silinder yang sudah naik sampai permintaan saat silinder.
- *Accumulated* Gerakan Jumlah silinder akumulasi sejak simulasi dimulai.

Permintaan dapat di tiga negara yang berbeda:

- *Served*, telah disajikan.
- *Waiting*, mereka antri menunggu untuk dilayani.
- *Delayed*, permintaan yang datang kemudian.

Disajikan ditampilkan pertama, maka mereka yang menunggu dan akhirnya mereka yang belum tiba.

Sementara simulasi dihentikan, klik kanan pada permintaan untuk menampilkan menu pop up yang memungkinkan Anda untuk mengubah dan menghapus.

Pengaturan:

Set perilaku simulasi:

- *Head Start position*: pengguna dapat memilih mana head awalnya (Range: 0-192 sektor).
- Algoritma:
 - *FIFO*, First in first out.
 - *LIFO*, Last in First Out.
 - *STF*, Shortest Seek Time First.
 - *SCAN*.
 - *C-SCAN*, Circular SCAN.
 - *LOOK*.
 - *C-LOOK*, LOOK circular.

Pengaturan hanya tersedia saat simulasi dihentikan

Menambahkan permintaan:

Permintaan dapat ditambahkan hanya sementara simulasi dihentikan, untuk setiap permintaan baru harus memasukkan informasi berikut:

- *Sector*, nomor sektor di mana permintaan dihasilkan (Nilai: 1 - 192).
- *Delay*, ketika permintaan akan memasuki antrian permintaan, jika 0 ditambahkan langsung ke antrian, jika tidak maka ditambahkan kemudian (Range: 1-100).

- *Color*, warna yang akan menarik elemen grafis yang terkait dengan permintaan.

Selain itu, sistem menghitung dan menunjukkan silinder yang sesuai dengan sektor yang diminta.

Common Tasks

Membuka dan menyimpan simulasi:

Sementara simulasi dihentikan, pengguna dapat menyimpan simulasi mereka dan membuka mereka nanti untuk menyelesaikan, review, dll ...

Menyimpan semua objek simulasi dan pengaturan.

Hal ini tidak mungkin untuk membuka atau menyimpan simulasi saat menjalankan OS Sim sebagai Applet memiliki beberapa keterbatasan karena masalah keamanan java.

Kontrol waktu

Untuk mengelola kemajuan simulasi, run normal, langkah demi langkah (satuan waktu setiap langkah), stop dan restart simulasi.

Selain itu Anda dapat bervariasi tingkat kecepatan dari 0,5 detik untuk 2 detik per satuan waktu. simulasi waktu selalu ditampilkan.

Peralatan

- Software: OSSIM
- Hardware: PC desktop/laptop
- Modul Praktikum

Langkah Kerja

A. Manajemen sistem file

Dalam kegiatan praktikum ini pohon folder dibuat dengan berbagai jenis objek sistem file: folder, file dan link. Pada UNIX setiap objek terkait dengan struktur yang disebut i-node, referensi ini berada di entri folder di mana objek berada.

Peringatan 1: Warna folder adalah hijau dan link berwarna biru. File warna didefinisikan ketika dibuat.

Peringatan 2: Dalam simulasi ukuran folder adalah 1 blok, sedangkan link tidak menempati ruang disk, mereka hanya entri data folder nya.

Peringatan 3: Struktur dari i-node

- Informasi, yang misalnya berisi sejumlah referensi ke i-node
- 12 pointer ke blok data
- 1 pointer ke blok tidak langsung.
- 1 pointer ke blok tidak langsung ganda.
- 1 pointer ke blok tidak langsung tiga.

Kegiatan 1: Alokasi Linked dengan FAT. Sistem File Objek

Tree				Type	Size Units
C:				Folder	1
	folder1			Folder	1
		file1.txt		File	101

		file2.txt		File	200
	folder2			Folder	1
		file3.txt		File	35
		Folder3		Folder	1
			file4.txt	File	120
		link1		Soft link (to file1.txt)	0

Kegiatan 2: Alokasi Linked dengan FAT (ukuran blok: 4 units)

Tree				Type	Size Units	Blocks
C:				Folder	1	1
	folder1			Folder	1	1
		file1.txt		File	101	26
		file2.txt		File	200	50
	folder2			Folder	1	1
		file3.txt		File	35	9.
		Folder3		Folder	1	1
			file4.txt	File	120	30
		link1		Soft link (to file1.txt)	0	0

Kegiatan 3: Alokasi Indexed (UNIX). Referensi ke i-nodes

Tree				Type	Size Units
/				Folder	1
	folder1			Folder	1
		file1.txt		File	101
		file2.txt		File	200
		link1		Hard link (to file1.txt)	1
	folder2			Folder	1
		file3.txt		File	460
		Folder3		Folder	1
			file4.txt	File	120
		link2		Hard link (to link1)	0

Kegiatan 4: Alokasi Indexed (UNIX). Indirections

Tree				Type	Size Units
/				Folder	1
	folder1			Folder	1
		file1.txt		File	101
		file2.txt		File	200
		link1		Hard link (to file1.txt)	1
	folder2			Folder	1
		file3.txt		File	460
		Folder3		Folder	1
			file4.txt	File	120
		link2		Hard link (to link1)	0

B. Penjadwalan disk

Peringatan: geometri harddisk adalah salah satu piring, 16 silinder, 12 sektor per silinder, 192 sektor total.

Kegiatan 1: Fair scheduling, FIFO

Permintaan untuk sektor disk yang disajikan dalam rangka kedatangan tepat.

Sector	Cylinder
100	8
3	0
150	12
23	1
120	10

Head awalnya diposisikan di sektor 0(Cylinder 0). Semua permintaan tiba di urutan tabel pada saat 0

kegiatan 2: STF scheduling contoh Starvation

Permintaan paling dekat dengan head dilayani pertama, hal ini dapat menyebabkan bahwa sementara permintaan tiba dekat dengan head, yang lebih jauh tidak dilayani (Starvation)

Sector	Cylinder	Delay
170	14	0

150	12	0
120	10	0
40	3	1
20	1	2
35	2	5
2	0	6
10	0	8
50	4	15
0	0	15

Head awalnya diposisikan di sektor 50 (Cylinder 4)

Kegiatan 3: Elevator (SCAN)

Gerakan head tidak tergantung pada permintaan, scan permintaan di kedua arah, permintaan dilayani seperti yang ditemukan.

Sector	Cylinder
100	8
3	0
150	12
23	1
120	10

Kegiatan 4: Perbaikan elevator 1. Circular SCAN

Gerakan head tidak tergantung pada permintaan, scan dalam satu arah, dari luar ke silinder bagian

dalam, permintaan dilayani seperti yang ditemukan.

Sector	Cylinder
100	8
3	0
150	12
23	1
120	10

Head awalnya diposisikan di sektor 90 (Cylinder 7), bergerak maju silinder terdalam.

Semua permintaan tiba di urutan meja pada saat 0.

Kegiatan 5: Perbaikan elevator 2. Look circular

Gerakan head tidak tergantung pada permintaan, dan scan dalam satu arah, dari luar ke silinder bagian dalam, dan hanya mencapai permintaan terdalam, permintaan dilayani seperti yang ditemukan.

Sector	Cylinder
100	8
3	0
150	12
23	1
120	10

Head awalnya diposisikan di sektor 90 (Cylinder 7), bergerak maju silinder terdalam.

Semua permintaan tiba di urutan meja pada saat 0.

C. Tugas

- a.** Jawab pertanyaan pada latihan OS-SIM file system
- b.** Jawab pertanyaan pada latihan OS-SIM disk scheduling