Spesifikasi Tugas Besar II
IF3055 Sistem Operasi

Tujuan

Tujuan dari pemberian tugas ini adalah

- Peserta mengetahui mekanisme *filesystem*.
- Peserta mengetahui mekanisme IPC.

Deskripsi

Pada tugas besar kali ini peserta diminta untuk membuat simulasi *filesystem*. Program yang akan dibuat akan berbentuk *terminal* di mana pengguna dapat mengeksekusi perintah-perintah.
Perintah yang perlu diimplementasikan antara lain

- ***format @[filepath] [size]***
  Perintah ini akan melakukan *formatting* pada sebuah *file* yang sedang dipakai oleh sistem operasi. *File* ini akan menjadi model sebuah *device* untuk *filesystem* yang akan dijalankan. *File* ini akan memiliki besar *size*. Peserta diharapkan mendesain struktur *filesystem* yang akan dibuat.
  Contoh eksekusi
  - `> format @/home/user/filesystem1 65536`

- ***mountfs @[filepath] [mount_point]***
  Perintah ini akan melakukan *mounting* pada *device* yang sudah pernah diformat pada perintah yang disebutkan di atas. Struktur direktori yang terdapat pada *device* akan muncul pada *mount point*. Saat *Terminal* dijalankan, pengguna harus melakukan *mounting* dengan *mount point* "*/*" sebelum dapat melakukan perintah-perintah yang lain.
  Pada implementasinya, perintah ini akan melakukan ***forking*** dan menjalankan sebuah *filesystem manager*. *Filesystem manager* ini bertugas untuk melakukan manajemen *filesystem* yang ada pada *device* serta melayani permintaan dari *Terminal*. *Terminal* hanya menerima input, menampilkan output yang bersesuaian.
  Komunikasi untuk antara *Terminal* dengan *filesystem manager* harus diimplementasikan dengan IPC. IPC yang dapat digunakan adalah *Message queue, Semaphore,* atau *Shared Memory*. *Filesystem manager* dapat diimplementasikan pada *source code* yang sama dengan *Terminal* atau program eksternal yang akan dipanggil dengan menggunakan *system call* **exec().**
  Contoh eksekusi
  - ```
    > mountfs @/home/user/filesystem1 /
    > ls
    dir1 dir2
    ```

- ***umount [mount_point]***
  Perintah ini akan melakukan *unmounting*. *Terminal* akan menterminasi proses *filesystem manager* yang bertanggung jawab atas *mount point* yang bersangkutan.

- ***pwd***
  Perintah ini akan menampilkan *current working directory.*

- ***ls [path]***
    ```

Perintah ini akan menampilkan seluruh *file* atau *folder* yang ada pada *current working directory* atau pada *folder* yang disebutkan pada parameter *path*. Perintah ini juga harus menampilkan ukuran *file*.

- ***cd [path]***
  Perintah ini akan mengganti *current working directory*.

- ***mkdir [path]***
  Perintah ini akan membuat sebuah direktori sesuai dengan *path*.

- ***rm [path]***
  Perintah ini akan menghilangkan sebuah *file* atau *folder* sesuai dengan *path*.

- ***cp [@][path1] [@][path2]***
  Perintah ini akan menyalin *file* atau *folder* dari *path1* ke *path2* selama masih memenuhi *quota* dari *device*. Penyalinan dilakukan antara *file/folder* pada sistem operasi (diberi tanda "@") ke *device* yang ada pada *Terminal*, sebaliknya, dan juga antara *file/folder* yang ada pada *Terminal*. Penyalinan antar *file/folder* yang ada pada sistem operasi tidak perlu diimplementasikan. Penyalinan *folder* harus dilakukan secara rekursif.

- ***mv [@][path1] [@][path2]***
  Perintah ini akan memindahkan *file/folder* dari *path1* ke *path2* selama masih memenuhi *quota device*. Pemindahan dilakukan antara *file/folder* pada sistem operasi (diberi tanda "@") ke *filesystem* yang ada pada *Terminal*, sebaliknya, dan juga antara *file/folder* yang ada pada *Terminal*. Pemindahan antar *file/folder* yang ada pada sistem operasi tidak perlu diimplementasikan. Pemindahan *folder* harus dilakukan secara rekursif. Perintah ini juga dapat digunakan untuk mengganti nama *file/folder*.

- ***cat [path]***
  Menampilkan isi dari *file*

- ***status***
  Program akan menampilkan status dari semua *filesystem* yang ada : *quota, free space, used space, file count, folder count*, dll.

- ***exit***
  Keluar dari program ☺

Catatan
- Argumen yang berupa *path* ke *file*, harus dapat digunakan secara relatif dan absolut.
- Ujilah perintah-perintah dengan *binary file* yang mudah untuk dibandingkan (e.g. mp3, jpg, gif, dll). Misal, coba pindah

Bonus

Aplikasi dapat melakukan *mounting* terhadap beberapa *device*.
Misal terdapat dua buah *device* A dan B. Struktur direktori yang tersimpan pada A adalah

- **dirA1**

    - **dirA11**
    - **dirA12**
- **dirA2**
- **dirA3**

Struktur direktori yang tersimpan pada B adalah

- **dirB1**
- **dirB2**
    - **dirB21**

Pengguna dapat melakukan *mounting* untuk kedua *device* tersebut pada *mount point* yang diinginkan selama *mount point* tersebut berupa direktori kosong.

Contoh :

```
> mountfs @/home/user/A /
> ls
dirA1 dirA2 dirA3
> mountfs @/home/user/B /dirA2
> cd dirA2
> ls
dirB1 dirB2
```

Pengguna juga dapat melakukan manipulasi (salin, pindah, hilang, dll) antar kedua *filesystem*. Contoh kasus yang perlu dicoba : Pengguna melakukan *mounting* kedua *filesystem* di atas. Kemudian pengguna menyalin direktori /dirA2/dirB2/dirB21 ke /dirA3. Direktori dirB21 ini haruslah muncul pada dirA3 (beserta isinya) jika *device* A dipakai lagi tanpa *device* B.

Ketentuan

A. Pengerjaan
    - Program dikembangkan dengan bahasa C/C++ dan *compiler gcc/g++* untuk lingkungan sistem operasi Linux. Jangan gunakan pustaka selain dari pustaka standard POSIX dan STL.
    - Program dapat dikembangkan dengan menggunakan IDE apapun (*Netbeans, CodeBlock,* dll) selama kompilasi dapat dilakukan dengan Makefile sederhana dan dapat dijalankan tanpa menggunakan IDE.

B. Deliverables

Pengiriman program dilakukan pada media CD dan juga lewat *email* (dikompresi *.tar.gz)

Struktur direktori

- bin, berisi *Terminal*
- src, berisi kode program
- doc, berisi laporan
- Makefile, *file* untuk melakukan kompilasi.

C. Laporan

Laporan ditulis dengan menggunakan format *font* Times New Roman 11. Semua kelompok harus menggunakan *template* yang sama.

Struktur laporan

D. Cover (lambang Ganesha hitam)

E. Bab I Latar Belakang Masalah

Tuliskan deskripsi permasalahan. Tulislah pemahaman masing-masing terhadap permasalahannya, apa saja yang menjadi permasalahan inti dari tugas. Jangan salin spesifikasi yang ada di dokumen ini.

F. Bab II Dasar Teori

Tulislah dasar-dasar teori yang diperlukan untuk mengerjakan tugas ini.

G. Bab III Pembahasan

Bab ini berisi

- Analisis.
  Jelaskan solusi yang digunakan : IPC serta desain *filesystem* beserta kelebihan dan kekurangannya.
- Pembagian Kerja
- Langkah pengerjaanx

H. Bab IV Pengujian
   Berisi pengujian dari program. Tuliskan juga kasus-kasus penanganan-penanganan kesalahan.
I. Bab V Kesimpulan dan Saran
   Tuliskan kesimpulan dan saran.
J. Daftar Pustaka
K. Lampiran
   Lampirkan kode program keseluruhan

Program beserta *hardcopy* laporan dikumpulkan tanggal 2 Desember 2008 pukul 16:59 WIB.

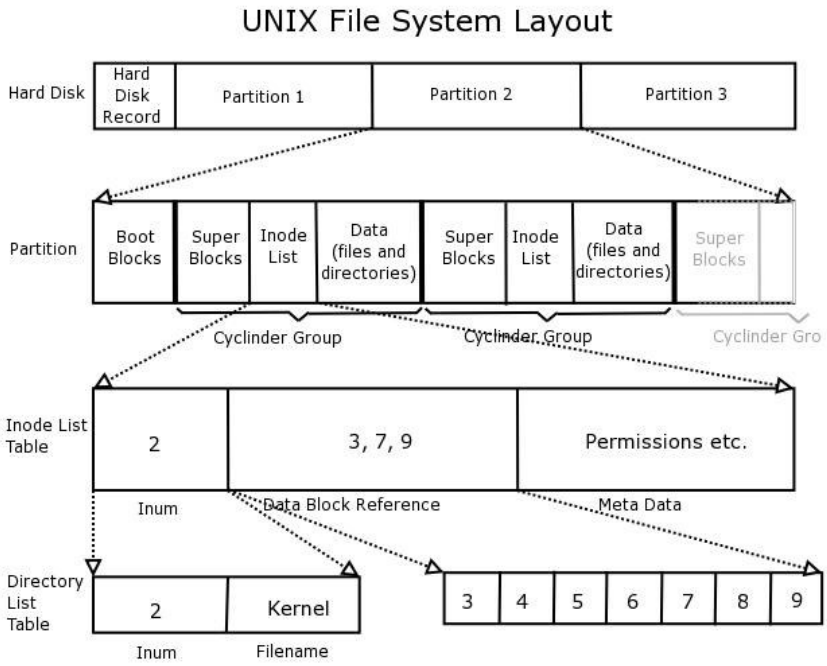
Penilaian

Penilaian dilakukan terhadap kelompok melalui pengerjaan tugas dan laporan. Penilaian juga dilakukan terhadap inividu dari performansi demo tugas. Keterlambatan dalam pengumpulan tugas akan dikenakan sanksi. Mencontek, plagiat, dan sejenisnya akan dikenai sanksi pemotongan nilai hingga tidak diterimanya tugas bersangkutan.

Hint

Filesystem

Filesystem adalah sebuah metode untuk melakukan penyimpanan dan pengaturan *file* pada sebuah media (*device*) agar mudah *file* tersebut mudah dicari dan dipakai. Sebuah *block device* dipetakan ke dalam struktur yang akan dipakai oleh *filesystem* tersebut. Sebuah UNIX *filesystem* biasanya memiliki struktur seperti di bawah.



Filesystem di atas cukup kompleks karena harus memfasilitasi berbagai keperluan seperti partisi *harddisk, hard link*, dll. *Filesystem* sederhana biasanya terdiri atas

1. s*uperblock*
   berisi metadata mengenai *filesystem* tersebut : tipe *filesystem*, ukuran, status, dll.
2. *inode list*
   berisi *inode*. *Inode* adalah reprensentasi sebuah obyek (*file*) dalam *filesystem*. *Inode* berbentuk sebuah struktur data yang berisi
   - tipe *file*: direktori, *executables, link, block special,* dll.
   - *permission*
   - *owner*
   - *filesize*
   - *block reference*, penunjuk data di mana data *file* tersebut tersimpan.
3. *data block*

Fork and Exec

Di dalam UNIX, dikenal *system call* bernama *fork* dan *exec*. *System call fork* berfungsi untuk menciptakan sebuah *process* yang menduplikasikan *process* yang memanggil *system call* tersebut. Sementara *exec* berfungsi untuk menjalankan sebuah *executable* pada *process* yang sedang berjalan.

Contoh penggunaan *fork*();

```
1. /* fork1.c */
2. #include <stdio.h>
3. int main(int argc, char * argv[])
4. {
5.     fork();
6.     puts("Hello World!");
7. }
```

Pada program di atas, *fork* akan menciptakan process yang identik dengan process utama. Keduanya akan mengeksekusi baris 6 setelah *fork* dipanggil. Oleh karena itu akan muncul dua buah "Hello World" pada terminal.

System call fork() memberikan nilai kembalian process id child pada parent dan nilai 0 pada child process. Jika forking gagal dilakukan maka nilai kembaliannya adalah bilangan negatif. Dengan cara ini *child process* dan *parent process* dapat diatur sehingga dapat melakukan aksi yang berbeda. Sebagai contoh.

```
8. /* fork2.c */
9. #include <stdio .h>
10.#include <sys /types.h>
11.
12.int main(int argc, char * argv[])
13.{
14.    pid_t childpid = fork();
15.    if (childpid == 0) /* Child proces */
16.        printf("I am the child. My PID is %d. My Parent ID is %d\n", getpid(),
   getppid());
17.    else if (childpid > 0) /* Parent process */
18.        printf("I am the parent. My PID is %d. My Child ID is %d\n", getpid(),
   childpid);
19.    else /* Failed */
20.        puts("Failed to fork");
21.}
```

Contoh hasil dari program di atas adalah

```
petra@petra-desktop:~/IF4038$ ./fork2
I am the child. My PID is 5463. My Parent ID is 5462
I am the parent. My PID is 5462. My Child ID is 5463
```

Contoh penggunaan exec

```
1. #include <unistd.h>
```

```
2. main()
3. {
4.     execl("/bin/ls", "/bin/ls", "-r", "-t", "-l", (char *) 0);
5. }
```

Program di atas akan mengeksekusi program "/bin/ls" dengan argumen-argumen "-r", "-t", "-l". Exec akan menggantikan *process image* yang sedang berjalan dengan *process image* dari *process* yang akan dijalankan.

Referensi

Berikut adalah situs-situs menarik yang mungkin dapat memberi inspirasi kepada peserta kuliah untuk mencari referensi lebih lanjut.

- http://learnlinux.tsf.org.za/courses/build/internals/ch08s04.html
- http://maleskoding.wordpress.com/2008/09/01/zombie-process/
- http://www.opengroup.org/onlinepubs/000095399/functions/exec.html
- http://www.cs.cf.ac.uk/Dave/C/