Pemanfaatan Strategi DFS untuk Manipulasi File System pada Sistem Operasi Berbasis Linux

Mufi Yanuar Triputranto / 13510106
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
mufi.yanuar@students.itb.ac.id

***Abstract*—Makalah ini membahas pemanfaatan DFS pada sebuah sistem operasi, terutama pada manipulasi berkas. Sebagai contoh kasus diimplementasikan sebuah program *copy* dan program *search* untuk menunjukkan pemanfaatan DFS. Hasil menunjukkan bahwa DFS khususnya dan strategi algoritma yang tepat pada umumnya menjadi salah satu bagian penting dalam sebuah sistem operasi.**

Index Terms*—*file system*, DFS, Linux, *copy*, *search

I. PENDAHULUAN

File system adalah suatu abstraksi yang diberikan sistem operasi untuk memudahkan pengguna dalam mengorganisasikan dan menyimpan informasi. Hasil abstraksi informasi tersebut berbentuk berkas (*file*) dan *folder*. Berkas merupakan informasi yang tersimpan dalam media penyimpanan sekunder – semisal hard disk, flash disk, dan semacamnya. Sedangkan *folder* ialah berkas khusus yang menyimpan penunjuk (*pointer*) menuju berkas-berkas lain. *Folder* memudahkan pengguna dalam mengorganisasikan berkas-berkas yang ada di dalam komputer. Setiap sistem operasi yang mendukung *file system* tertentu menyediakan operasi-operasi dasar yang berkaitan dengan *file system* tersebut, misalnya operasi untuk membuat berkas, membuka berkas, membaca berkas, dan lain sebagainya. Operasi-operasi tersebut bersifat umum / abstrak sehingga untuk *file system* yang berbeda pengguna tetap dapat melakukan operasi yang sama tanpa perlu tahu desain dan mekanisme spesifik dari *file system.*

Pada sistem operasi modern yang kita kenal – semisal Linux atau Windows – berkas-berkas diorganisasikan secara hirarkis. Struktur hirarkis seperti ini bisa dipandang seperti struktur data pohon. Dengan menggunakan operasi-operasi dasar *file system* yang telah disediakan sistem operasi kita dapat melakukan operasi-operasi yang serupa seperti pada struktur data pohon. Beberapa operasi yang biasa digunakan pada file system ialah operasi penyalinan berkas dan pencarian berkas.

Pada operasi penyalinan akan mudah jika yang akan disalin ialah sebuah berkas biasa. Namun jika yang akan disalin ialah sebuah *folder*, maka harus dilakukan penelusuran satu persatu terhadap tiap-tiap berkas yang ada dalam *folder* tersebut, dan tentunya tidak menutup kemungkinan *folder* tersebut berisi *folder* lain sehingga harus dilakukan penelusuran lebih lanjut. Untuk operasi pencarian berkas pun pada dasarnya tidak berbeda jauh dengan operasi penyalinan – harus dilakukan penelusuran mulai dari *folder* awal sampai berkas yang bersangkutan ditemukan atau tiap berkas dalam *folder* telah ditelusuri.

Pada makalah ini akan ditunjukkan cara untuk melakukan penyalinan berkas dan pencarian berkas pada sebuah *file system* dalam lingkungan sistem operasi Linux. Meskipun Linux sendiri telah menyediakan program untuk melakukan penyalinan berkas dan pencarian berkas, program yang akan ditunjukkan pada makalah ini ditulis tidak lain untuk memahami manfaat dan pentingnya strategi DFS pada sebuah sistem operasi. Source code untuk program *copy* dan program *search* juga disertakan pada bagian lampiran.

II. PEMBAHASAN

A. Ide Dasar

Pada sebuah *file system* hirarkis kita dapat memperlakukan *folder* sebagai sebuah simpul akar atau simpul-simpul dalam dari sebuah pohon, sedangkan sebuah berkas merupakan simpul terminal/daun dari sebuah pohon. Untuk melakukan proses penyalinan berkas artinya kita harus menelusuri pohon di mulai dari *folder* awal sampai dengan suatu berkas. Bila suatu *folder* berisi *folder* lain maka *folder* tersebut harus disalin pula. Untuk melakukannya kita dapat menerapkan strategi DFS.

Langkah-langkah untuk melakukan penyalinan file dengan DFS ialah :

1. Buka sebuah berkas.
2. Jika berkas tersebut adalah berkas biasa, maka salin berkas tersebut ke sebuah berkas baru.
3. Jika berkas tersebut adalah sebuah *folder*, maka baca isi dari *folder* tersebut lalu ulangi mulai langkah 1 sampai *folder* selesai dibaca.

Sedangkan untuk pencarian berkas pada dasarnya sama seperti pada penyalinan file. Dengan masukkan sebuah nama berkas, harus dilakukan penelusuran ke setiap simpul sampai sebuah berkas dengan nama yang serupa

ditemukan atau semua simpul telah ditelusuri. Langkah-langkah untuk mencari file dengan DFS adalah sebagai berikut :

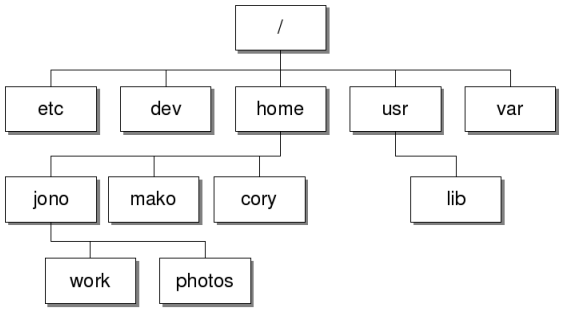
1. Jika berkas adalah berkas biasa, maka cek apakah nama berkas sama dengan nama yang diminta.
2. Jika berkas adalah *folder*, cek apakah nama *folder* sama dengan nama yang diminta. Setelah itu baca isi *folder* dan ulangi langkah 1 sampai *folder* selesai dibaca.

B. File System Linux

Linux yang memenuhi standar POSIX telah menyediakan operasi-operasi dasar terkait dengan manipulasi berkas dan *folder*. Beberapa operasi yang relevan ialah :

1. open, membuka sebuah berkas sesuai dengan nama yang ada pada parameter input dan mengembalikan sebuah deskriptor (pointer) menuju berkas tersebut. Jika berkas tidak bisa dibuka atau tidak ada maka fungsi akan mengembalikan nilai -1.
2. read, membaca berkas yang telah dibuka dan menyimpan hasil pembacaan ke sebuah buffer.
3. write, menuliskan data yang ada pada buffer ke dalam sebuah berkas yang dibuka.
4. close, menutup berkas.
5. opendir, membaca sebuah *folder* sesuai dengan nama yang ada parameter input dan mengembalikan deskriptor menuju *folder* tersebut. Jika *folder* tidak bisa dibuka atau tidak ada maka fungsi akan mengembalikan nilai -1.
6. readdir, membaca entri di dalam sebuah *folder*. Fungsi akan mengembalikan nilai berupa tipe bentukan yang menunjukkan informasi mengenai entri dalam *folder* tersebut. Entri bisa berupa berkas biasa atau *folder* lain.
7. mkdir, membuat *folder* baru dengan nama yang diberikan pada parameter input.
8. closedir, menutup *folder*.

Untuk struktur *file system*, Linux menggunakan struktur hirarkis dengan *folder* khusus dengan nama / atau disebut juga *root* sebagai *folder* utamanya. Berikut adalah struktur *file system* Linux secara umum :



Folder home adalah *folder* yang menyimpan data-data pribadi pengguna. *Folder* lainnya adalah *folder* yang dibuat oleh Linux sendiri untuk menyimpan data-data penting bagi sistem operasi seperti device driver, data username dan password pengguna, *system file*, dan lain-lain.

Untuk melakukan manipulasi file seperti menyalin atau menghapus, Linux biasanya hanya memperbolehkan manipulasi tersebut berada di dalam *folder* home. Namun dimungkinkan juga pada *folder* root dengan akses sebagai administrator.

III. IMPLEMENTASI

Implementasi dilakukan pada lingkungan sistem operasi Linux Fedora 17 dan bahasa pemrograman C. Bahasa C dipilih karena pada dasarnya Linux menggunakan bahasa C dalam implementasinya, sehingga lebih intuitif jika menggunakan bahasa C dalam memrogam sistem dalam lingkungan Linux.

Untuk pendefinisan fungsi-fungsi dasar terkait manipulasi berkas digunakan fungsi bawaan yang telah disediakan pustaka (*library*) dasar bahasa C sesuai standar C11 dan juga pustaka bawaan Linux untuk pemrograman sistem.

A. Implemetasi Program Copy

Program *copy* dibagi menjadi dua komponen fungsi utama. Satu fungsi adalah fungsi yang dijadikan basis untuk melakukan penyalinan berkas biasa dan satu fungsi lagi adalah fungsi rekursif untuk melakukan penyalinan *folder*, atau dengan kata lain menyalin 1:1 struktur sebuah bagian pohon dengan cara penelusuran DFS. Pemanggilan fungsi tersebut melalui fungsi main seperti program C pada umumnya.

B. Implementasi Program Search

Program pencarian berkas yang diimplementasikan yaitu mencari berkas berdasarkan nama berkas. Bila ditemukan, maka program akan mencetak path relatif berkas dari *folder* awal. Pencarian akan dilakukan sampai semua *folder* dan berkas di dalam *folder* awal telah selesai dilakukan.

IV. HASIL UJI COBA

A. Copy

Hasil uji coba menunjukkan bahwa penyalinan dapat dikerjakan dalam waktu yang sangat cepat bila struktur *folder* tidak terlalu dalam. Bila *folder* yang disalin memiliki kedalaman yang terlalu besar, program *copy* akan mengalami crash. Hal ini memang sesuai perkiraan mengingat program dipanggail secara rekursif sehingga bila pemanggilan yang dilakukan terlalu banyak maka memori yang digunakan program akan segera habis. Namun terlepas dari itu program *copy* berhasil melakukan penyalinan 1:1 tanpa ada cacat. Untuk kompleksitas algoritma program *copy* ini akan sama dengan jumlah *folder* yang disalin ditambah waktu penyalinan tiap berkas yang ada, sehingga kompleksitasnya *O(n+m.k)*, dengan n adalah jumlah *folder* yang ada, m adalah total berkas yang ada, dan k adalah konstanta untuk waktu penyalinan sebuah berkas.

B. Search

Hasil uji coba menunjukkan bahwa pencarian sebuah berkas dengan nama tertentu berhasil dilakukan selama *folder* tempat pencarian dilakukan tidak memiliki kedalaman yang terlalu besar. Untuk kompleksitas algoritma program *search* akan sama dengan banyaknya *folder* dan berkas yang ditelusuri sehingga kompleksitasnya adalah *O(n)*.

V. KESIMPULAN

DFS adalah strategi algoritma yang penting dalam suatu sistem operasi, terutama yang berkaitan dengan *file system*. Tidak hanya DFS, pendekatan lain pun dapat diimplementasikan dalam sistem operasi untuk melakukan operasi tertentu, seperti Divide & Conquer untuk membandingkan kesamaan berkas. Dalam makalah ini yang ditunjukkan bagaimana proses penyalinan dan proses pencarian berkas bekerja dalam sebuah sistem operasi.

Untuk kasus penyalinan berkas secara intuitif kita bisa tahu memang DFS-lah strategi yang paling mudah untuk diimplementasikan, terutama apabila DFS yang digunakan menggunakan pendekatan rekursif. Sayangnya pendekatarn rekursif terbilang mahal untuk sebuah mesin riil karena membutuhkan memori yang besar apabila *folder* yang akan disalin memiliki kedalaman yang besar. Oleh karena itu DFS ini bisa diganti menggunakan pendekatan iteratif yang lebih lambat namun hemat memori. Selain itu tidak menutup kemungkinan ada strategi lain yang lebih baik daripada DFS untuk proses penyalinan berkas.

Sedangkan untuk proses pencarian berkas meskipun terbilang sederhana, namun pada kenyataannya strategi DFS tidak digunakan pada kebanyakan sistem operasi pada saat ini. Kekurangan dari DFS dalam pencarian berkas ialah – selain karena memakan banyak memori – tidak mempertimbangkan bahwa pada umumnya struktur suatu *file system* tidak banyak berubah. Untuk pencarian berkas saat ini banyak yang menggunakan *indexing* daripada DFS, yaitu pencatatan berkas-berkas yang ada dalam komputer secara berkala.

Secara keseluruhan, strategi algoritma yang tepat sangat penting dalam perancangan sistem operasi yang baik.

VI. LAMPIRAN

A. Source Code Program Copy

```
#include "copy.h"

int copy_file(string source, string target) {
    FILE *src;
    FILE *tgt;
    int buf[BUFFER_SIZE];
    int len;

    if ((src = fopen(source,"rb")) == NULL) {
        return 0;
    }
    if ((tgt = fopen(target,"wb")) == NULL) {
        fclose(src);
        return 0;
    }

    while(!feof(src)) {
        len = fread(buf,1,BUFFER_SIZE,src);
        if(ferror(src)) {
            printf("error¥n");
            fclose(src);
            fclose(tgt);
            return -1;
        } else {
            fwrite(buf,1,len,tgt);
        }
    }
    fclose(src);
    fclose(tgt);
    return 1;
}

void copy(string source, string target) {
    struct stat fileinfo;
    stat(source,&fileinfo);

    if (S_ISREG(fileinfo.st_mode)) {
        copy_file(source,target);
    } else {
        DIR *d;
        struct dirent *dir;
        char path[2048];
        char tempsource[2048];

        mkdir(target,0777);
        d = opendir(source);
        if(d) {
            while((dir = readdir(d))!=NULL) {
                if(strcmp(dir-
>d_name,"..")&&strcmp(dir->d_name,".")) {
                    strcpy(tempsource,source);
                    strcat(tempsource,"/");
                    strcat(tempsource,dir-
>d_name);

                    strcpy(path,target);
                    strcat(path,"/");
                    strcat(path,dir->d_name);

                    copy(tempsource,path);
                }
            }
            closedir(d);
        }
    }
```

B.Source Code Program Search

```
#include "search.h"

void search(string name, string path) {
    struct stat fileinfo;
    stat(path,&fileinfo);

    if (S_ISREG(fileinfo.st_mode)) {
        if (strstr(path,name)!=NULL) {
            printf("%s¥n",path);
        }
    } else {
        if (strstr(path,name)!=NULL) {
            printf("%s¥n",path);
        }
        DIR *d;
        struct dirent *dir;
        char file_path[2048];
        d = opendir(path);
        if(d) {
            while((dir = readdir(d))!=NULL) {
                if (strcmp(dir-
>d_name,"..")&&strcmp(dir->d_name,".")) {
                    strcpy(file_path,path);
                    strcat(file_path,"/");
                    strcat(file_path,dir-
>d_name);

                    search(name,file_path);
                }
            }
            closedir(d);
        }
    }
```

VII. REFERENSI

Levitin, A. (2012). *Introduction to The Design and Analysis of Algorithm 3rd Edition.* Pearson.

Tanenbaum, A. S. (2007). *Modern Operating Systems.*

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2012

Mufi Yanuar Triputranto