LAPORAN PRAKTIKUM 2
SISTEM OPERASI E

ZAHRAH CITRA HAFIZHA 5114100012
FIKRY KHAIRYTANIM 5114100192

MEMBUAT SHELL

Kali ini kita akan membuat shell sendiri dari tugas yang diberikan. Kita ditugaskan membuat shell denga syarat-syarat:

1. Bisa berpindah directory aktif
2. Bisa menajalankan perintah-perintah yang ada di /bin dan usr/bin
3. Tidak berhenti saat ditekan ctrl+c dan ctrl+z
4. Otomatis keluar ketika menekan ctrl+d
5. Bisa menjalankan ebberapa perintah secara background jika perintah diakhiri dengan ampersand (&)
6. Dilarang menggunakan fungsi system

Untuk membuat shell pertama-tama kita mendefinisika library yang dibutuhkan;

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include <signal.h>
#include <errno.h>
#define MAXLENGTH 1024
#define DELIMS " \t\r\n"
```

Untuk membuat syarat shell nomor 3 kita membutuhkan fungsi signal handler, ctrl+c dengan SIGINT dan ctrl+z dengan SIGTSTP. Untuk menandai bahwa kodingan tidak dapat berhenti dengan SIGINT dan SIGTSTP maka akan keluar juga beberapa print didalam fungsi.

```
void signalhandler(int signum)
{
        printf("ketik exit untuk keluar program");
        printf("Masukkan perintah ");
}

int main()
{
        signal(SIGINT, signalhandler);//signaling
ctrl+c
        signal(SIGTSTP, signalhandler);
//signaling ctrl+z
```

Lalu kita mendefinisikan variabel yang dibutuhkan

```
char *cmd;
        char line[MAXLENGTH]; //get command line
        char *argv[100]; //user command
        char *path="/bin/"; //set path at bin
        char fullpath[20]; //full file path
        int argc; //argument count
```

Untuk membuat syarat shell no 1, kita bisa menggunakan fungsi DELIMS. Jika input yang diberikan adalah “cd” lakkukan fungsi STRTOK(0,DELIMS), jika tidak outputkan perintah bahwa perintah cd masih kurang.

```
while(1)
        {

        printf("Masukkan perintah ");

        if (!fgets(line, MAXLENGTH, stdin)) break;

        if ((cmd = strtok(line, DELIMS)))
        {
                errno = 0;

                if (strcmp(cmd, "cd") == 0)
                {
                char *arg = strtok(0, DELIMS);

                if (!arg) fprintf(stderr, "argumen cd masih kurang\n");
                else chdir(arg);
                }

        }

        if (strcmp(cmd, "exit") == 0)
        break;
```

Untuk bisa menjalankan syarat shell no 2 yaitu perintah /bin dan usr/bin/. Dikarenakan syarat terakhir kita tidak boleh menggunakan fungsi system yang kata pak bas berbahaya, maka kita bisa menggunakan fungsi execvp yang secara logika lebih aman.

Jangan lupa membuat syarat agar perintah dapat dijalankan secara background dengan penulisan ampersand.

```
char *token; //split command into separate things
        token=strtok(line," ");
        int i=0;
        int temp=0;
        while(token!=NULL)
        {
                argv[i]=token;
                token=strtok(NULL," ");
                i++;
        }
        argv[i]=NULL; //set last value to NULL for execvp
        argc=i; //get argument count
        strcpy(fullpath,path); //copy /bin/ to file path
        strcat(fullpath,argv[0]); //add program to path

        for(i=0;i<strlen(fullpath);i++) //delete newline
        {
                if(fullpath[i]=='\n')
                {
                        fullpath[i]='\0';
                }
        }
        if(strcmp(argv[argc-1],"&")==0) temp=1;
        pthread_t pid=fork(); //fork child
        if(pid==0) //child
        {
                if(strcmp(argv[argc-1],"&")==0) argv[argc-1]='\0';
                execvp(fullpath,argv);
        }
        wait();
        else    //parent
        {
                wait();
        }
        }
}
```

Dan untuk point no 4 yaitu ctrl+d secara otomatis dapat keluar dari shell.

LANGKAH-LANGKAH PEMBUATAN PROGRAM MENCARI JUMLAH BILANGAN PRIMA SEBANYAK LIMIT YANG DIBERIKAN

```c
#include <stdio.h>
#include <pthread.h>

int  temp;
void *prime1(void *args)
{
        int i,cek,flag=0;
        cek=*((int *)args);
        for(i=1; i<=cek; i++)
        {
                        if(cek%i==0)
                        flag++;
        }
        if(flag==2)
        temp++;
}

void main ()
{
        int data,i;
        printf("INPUT LIMIT: ");
        scanf("%d", &data);
        pthread_t t1[data];
        int input[data];
        for(i=2;i<data;i++)
        {
                input[i]=i;
                pthread_create(&t1[i], NULL, prime1,&input[i]);
        }
        for(i=2; i<data; i++)
        pthread_join(t1[i],NULL);

        printf("JUMLAH BILANGAN PRIMA SEBELUM %d = %d\n",data, temp);
}
```

Dalam pembuatan program ini menggunakan library pthread.h untuk fungsi threadnya. Yang pertama kali dibuat adalah fungsi main dengan mendeklarasikan thread yang digunakan dan mendeklarasikan limit yang akan diinput.

Dalam fungsi void main:

Setelah mendeklarasikan fungsi thread dan limit untuk nanti di passing ke fungsi prime1, dibuat deklarasi create dan join. Di dalam create akan di jelaskan apa yang telah dibuat, nama fungsi yang akan di passing, dan type data saat melakkukan passing. Kemudian di join adalah sebagai wait untuk melanjutkan ke perintah berikutnya harus menunggu thread apa yang ada didalam join.

Dalam fungsi prime1:

Didalam fungsi prime1 ada value &data yang dikirim sebagai perhitungan limit, didalam fungsi ini dideklarasikan “i” dan “j” sebagai looping, *cek untuk menyimpan dari &data yang telah di passing, flag sebagai penanda berapa kali nilai tersebut dan dibagi, dan yang terakhir adalah temp untuk menentukan jumlah berapa banyak bilangan primanya.

Didalam fungsi ini mencari fungsi prima dengan mod “i” dengan “j”. pembagian “i” dimulai dari 2 sampai limit yang diminta, mod yang dilakkukan oleh “j” sampai dengan sebesar “i”, mod dilakkukan mulai dari 1 sampai sebesar “i”. Bila mod sama dengan nol maka akan di increment flag nya, jika flag sama dengan dua yaitu bilangan 1 dan nilai “i” maka akan di increment temp nya sebagai penanda jumlah dari bilangan prima.

Setelah selesai dilakkukan sampai akhir maka didalam fungsi main akan dipanggil temp terakhir yang tercetak didalam fungsi prime1.

LANGKAH-LANGKAH PEMBUATAN READ DATA KEMUDIAN DI DUPLIKAT KE FILE YANG MASIH KOSONG

*readcopy.c (~/Documents/Praktikum_github/SISOP_E15) - gedit

Open Save Undo

*readcopy.c

```
#include <stdio.h>
#include <pthread.h>

void *thread1(void *args){
        FILE *in, *out;
        char karakter;
        in=fopen ("contoh2.txt", "r");
        out=fopen ("contoh3.txt", "w");
        while ((karakter=fgetc (in))!=EOF)
        {
                (fputc(karakter, out));
        }
        fclose (in);
        fclose (out);
}
void *thread2(void *args){
        FILE *in, *out;
        char karakter;
        in=fopen ("contoh2.txt", "r");
        out=fopen ("contoh3.txt", "w");
        while ((karakter=fgetc (in))!=EOF)
        {
                (fputc (karakter, out));
        }
        fclose (in);
        fclose (out);

}
int main(){
        pthread_t t1, t2;
        pthread_create (&t1, NULL, thread1, NULL);
        pthread_create (&t2, NULL, thread2, NULL);
        pthread_join (t1, NULL);
        pthread_join (t2, NULL);
}
```

C Tab Width: 8 Ln 28, Col 2 INS

*readcopy.c (~/Documents/Praktikum_github/SISOP_E15) - gedit

Open Save Undo

*readcopy.c

```
#include <stdio.h>
#include <pthread.h>
void *thread1(void *args){
        FILE *in, *out;
        char karakter;
        in=fopen ("contoh.txt", "r");
        out=fopen ("contoh2.txt", "w");
        while ((karakter=fgetc (in))!=EOF)
        {
                (fputc(karakter, out));
        }
        fclose (in);
        fclose (out);

}
void *thread2(void *args){
        FILE *in, *out;
        char karakter;
        in=fopen ("contoh2.txt", "r");
        out=fopen ("contoh3.txt", "w");
        while ((karakter=fgetc (in))!=EOF)
        {
                (fputc (karakter, out));
        }
        fclose (in);
        fclose (out);
}
int main(){
        pthread_t t1, t2;
        pthread_create (&t1, NULL, thread1, NULL);
        pthread_join (t1, NULL);
        pthread_create (&t2, NULL, thread2, NULL);
        pthread_join (t2, NULL);
}
```

C Tab Width: 8 Ln 27, Col 2 INS

Karena saya baru diberitahu oleh teman saya bahwa boleh menceritakan dengan bahasa sendiri dan resmi maka saya akan mulai menjelaskan tugas berikutnya mengenai pembuatan readcopy file.

Nah yang akan saya jelaskan kali ini adalah bahwa yang dilakkukan oleh tugas ini yaitu dapat membaca file yang ada disuatu folder dan bisa di duplikat di suatu file yang diinginkan. Disini saya membuat dua tipe setelah pak bas menjelaskan pada pertemuan di kelas yaitu sequential yang pertama saya buat dan kemudian yang parallel yang pak bas request agar dalam berjalan bersamaan sambil read dan menduplikat.

Fungsi main:

didalam main sangat simpel yang dibutuhkan hanyalah mendeklarasikan thread, mencreate thread, dan membuat join thread. Jadi semua proses berjalan pada fungsi thread1 dan thread2.

Fungsi thread1 dan thread2:

Isi kedua thread ini sama persis jadi hanya masalah penggunaan dari dari create dan joinnya yang mengatur kerja kedua fungsi ini.

Jadi didalam fungsi ini ada pendeklarasian file untuk in dan out, yang in untuk membaca dan out untuk menulis ke file yang diinginkan. Yang dilakkukan oleh while dikodingan ini adalah membaca semua isi dari “in” sampai EOF (end of file), kemudian tiap tiap karakter pada “in” langsung dimasukkan ke dalam “out”. Kemudian setelah while sampai pada EOF, diclose keduanya yaitu “in” dan “out”.

JADI

yang membedakan antara kedua screenshot itu hanya pada di create dan joinnya. Dan bukan hanya sih tapi itu sebagai kunci dari cara kerja threadnya. Screenshot yang menuliskan thread thread join join itu berarti menjalankan keduanya terlebih dahulu baru di wait thread1 thread2nya. Untuk yang create join create join itu terlihat runtut maksudnya yaitu bekerja menunggu thread1 kelar kemudian masuk ke thread2.