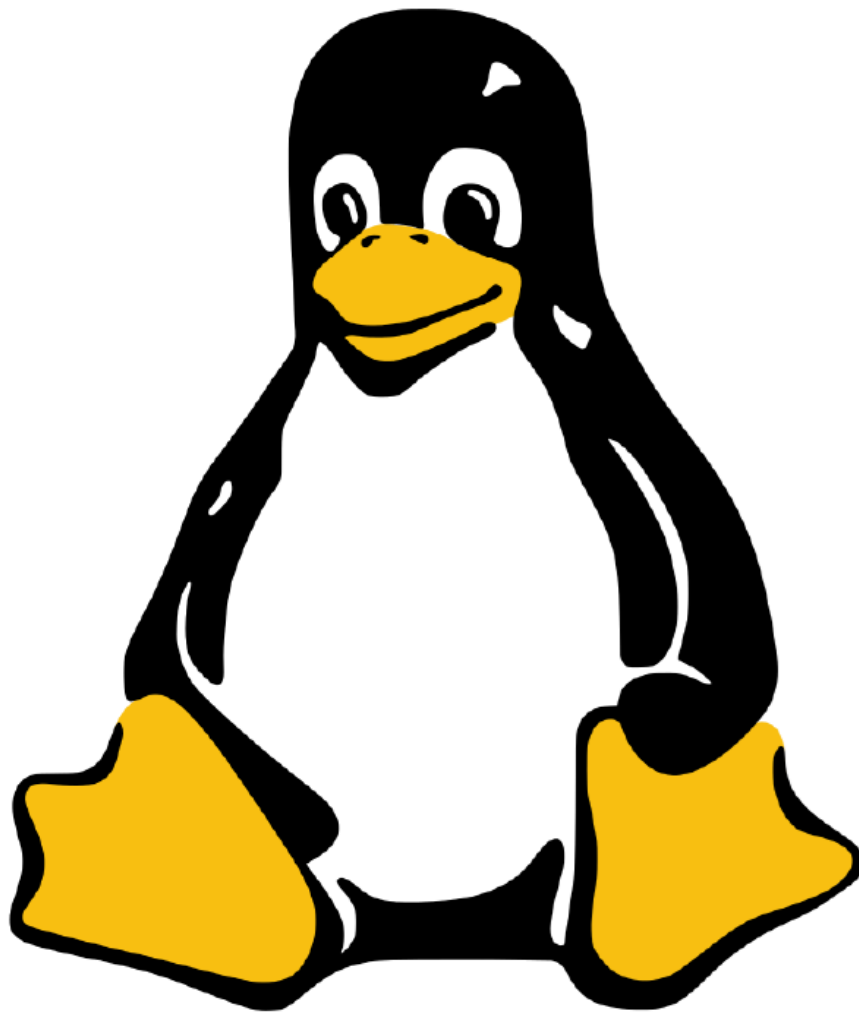


DOKUMENTASI PROGRAM

MODUL 2

SISTEM OPERASI E



Oleh:
Kelompok E-5
Panji Rimawan (5114100075)
Steven Kurniawan (5114100100)

1. Program Shell

Program ini dapat berjalan seperti shell pada umumnya. Seperti : bisa berpindah direktori aktif; bisa menjalankan perintah-perintah yang ada di **/bin** dan **/usr/bin**; tidak berhenti ketika ditekan **CTRL+C** dan **CTRL+Z**; otomatis logout/keluar ketika menekan **CTRL+D**; bisa menjalankan beberapa perintah secara background jika perintah diakhiri dengan **ampersand (&)**.

- a. Hal pertama yang dilakukan adalah memasukkan library dan mendeklarasi macro variable dari **MAX_COMMAND** (sebagai panjang maksimal dari baris perintah) dan **MAX_PARAMETER** (sebagai panjang maksimal dari parameter yang bisa diinput).

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#define MAX_COMMAND 150
#define MAX_PARAMETER 10
```

- b. Deklarasi fungsi yang akan digunakan

```
void signalHandler (int);
void parsingCommand(char*, char**);
int executeCommand(char**);
int flag=0;
```

Fungsi signalHandler → Digunakan untuk menangkap sinyal yang diinterupsi dari user.

Fungsi parsingCommand → Digunakan untuk mem-parsing / memilah-milah baris perintah yang diinputkan oleh user.

Fungsi executeCommand → Digunakan untuk mengeksekusi perintah inputan user yang sudah di parsing di fungsi parsingCommand.

Variable flag → Digunakan untuk menandai apakah perintah diakhiri dengan ampersand '&' atau tidak

- c. Implementasi fungsi

- Fungsi signalHandler

```
void signalHandler (int sigNum)
{
    if (sigNum == SIGTSTP) printf (" myShell received SIGTSTP\n");
    else if (sigNum == SIGINT) printf (" myShell received
SIGINT\n");
}
```

- Fungsi parsingCommand

```
void parsingCommand(char *command, char **parameter)
{
    int i;
    for (i=0; i<MAX_PARAMETER; i++) {
        parameter[i]=strsep(&command, " ");
        if (parameter[i]==NULL) break;
    }
}
```



- Fungsi executeCommand

```
int executeCommand(char **parameter)
{
    pid_t pid = fork();

    if (pid==-1) {
        // menampilkan pesan error jika gagal duplikat/forking
        char *error=strerror(errno);
        printf ("fork: %s\n", error);
        return 1;
    }
    else if (pid==0) {
        // eksekusi perintah
        execvp(parameter[0], parameter);

        // error message, jika perintah yang diinputkan user tidak
        // ada di folder binary
        char *error=strerror(errno);
        printf ("shell: %s: %s\n", parameter[0], error);
        return 0;
    }
    else {
        if (flag==-1) {
            // jika tidak terdapat '&' di akhir perintah
            int childStats;
            waitpid(pid, &childStats, 0);
            return 1;
        }
    }
}
```



d. Main function dari program ini

```
int main (void)
{
    char command[MAX_COMMAND+1];
    char *parameter[MAX_PARAMETER+1];

    if (signal(SIGTSTP, signalHandler) == SIG_ERR)
        printf("\n Proccess can't catch SIGTSTP\n");
    if (signal(SIGINT, signalHandler) == SIG_ERR)
        printf("\n Proccess can't catch SIGSTOP\n");

    while(1) {
        // cetak username
        char *username=getenv("USER");
        printf ("%s@myshell> ", username);

        // baca perintah dari input user, CTRL+D->exit program
        if (fgets(command, sizeof(command), stdin) == NULL) {
            printf("\n");
            break;
        }

        // cek ampersand
        if ((int)command[strlen(command)-2]== (int)('&')) {
            flag=1;
            command[strlen(command)-2]='\0';
        }
        else {
            flag=-1;
        }

        // menghapus karakter '\n', diubah menjadi karakter '\0'
        if (command[strlen(command)-1] == '\n') {
            command[strlen(command)-1] = '\0';
        }

        // parsing command
        parsingCommand(command, parameter);

        // builtin command : cd, exit
        if (strcmp(parameter[0], "exit") == 0) break;
        else if (strcmp(parameter[0], "cd") == 0) {
            chdir(parameter[1]);
        }
        // eksekusi perintah
        else if (executeCommand(parameter) == 0) break;
    }

    return 0;
}
```



Contoh output ketika program dijalankan:

```
panji@PANJI-PC: ~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$ ./soal1-new
panji@myshell> pwd
/home/panji/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2
panji@myshell> ls
catatan_sesi2  coba git  file1.txt  file1.txt~  file2.txt  file3.txt  soal1-new  soal1-new.
panji@myshell> mkdir abc
panji@myshell> ls
abc  catatan_sesi2  coba git  file1.txt  file1.txt~  file2.txt  file3.txt  soal1-new  soal1
panji@myshell> ^C myShell received SIGINT
^Z myShell received SIGTSTP
cd abc
panji@myshell> pwd
/home/panji/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2/abc
panji@myshell> cd ..
panji@myshell> ls&
panji@myshell> abc  catatan_sesi2  coba git  file1.txt  file1.txt~  file2.txt  file3.txt
Soal.png
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$
```

2. Program untuk Mencari Jumlah Bilangan Prima kurang dari N dan Membuat Aplikasi Multithread untuk Menyalin Isi File

- **Mencari Jumlah Bilangan Prima kurang dari N**

Untuk mencari jumlah bilangan prima kurang dari N, pertama-tama kita meminta user untuk menginputkan besar dari N, kita juga punya variabel counter yang awalnya bernilai 0 untuk menghitung banyak bilangan prima. Kemudian kita membuat thread untuk mencari bilangan prima tersebut. Di dalam thread tersebut, kita mempunyai variabel cek yang bernilai 0. Suatu bilangan dikatakan bilangan prima apabila hanya habis dibagi dengan 1 dan bilangan itu sendiri. Apabila bilangan tersebut habis dibagi dengan bilangan selain itu, maka nilai cek akan berubah menjadi 1. Apabila setelah dibagi nilai cek tetap 1, maka counter akan ditambah 1. Nilai pada counter inilah yang menghitung banyak bilangan prima kurang dari N.

- **Membuat Aplikasi Multithread untuk Menyalin Isi File**

Untuk membuat ini, pertama kita harus membuat 2 buah thread, dimana thread pertama bertugas untuk menyalin isi file dari file 1 ke file 2, sedangkan thread kedua bertugas untuk menyalin isi file dari file 2 ke file 3. Dan kedua thread tersebut harus bisa bekerja secara bersamaan. Pada fungsi main, pertama-tama kita memberi tanda=0 terlebih dahulu, kemudian memanggil thread pertama. Pada thread yang pertama ini tanda kita rubah menjadi 1. Kemudian kita membuka file 1 sebagai input file dan file 2 sebagai output file. Kemudian isi dari file 1 kita taruh di variabel char temp dengan fgetc(inp), kemudian kita salin ke file 2 dengan fputc(temp,out). Ini akan berlangsung terus sampai temp berada di EOF. Setelah selesai, file 1 dan file 2 ditutup kemudian tanda dirubah menjadi 2. Pada thread yang kedua, kita menunggu apabila tanda masih 0. Apabila tanda sudah tidak 0, maka artinya thread 1 sudah berjalan sehingga kita dapat menjalankan thread kedua. Isi dari file 2 kita taruh di variabel char temp dengan fgetc(inp), kemudian kita salin ke file 3 dengan fputc(temp,out). Kemudian kita menunggu sampai thread 1 selesai dijalankan. Apabila sudah selesai maka kita menutup file 2 dan file 3.



a. Implementasi fungsi :

- Deklarasi file header yang digunakan di program ini.

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

int counter=0;
```

- Implementasi fungsi

Fungsi thread **Prima**, berguna untuk mencari jumlah bilangan prima kurang dari N.

```
void *Prima (void *args)
{
    int *n = (int *) args;
    int p,q;
    for(p=2; p<*n; p++)
    {
        int cek=0;
        for (q=2; q<p; q++)
        {
            if(p%q == 0)
            {
                cek=1;
            }
        }
        if(cek==0)
        {
            counter=counter+1;
        }
    }
    if(*n<=2)
    {
        counter=0;
    }
}
```



Fungsi **Thread1**, berguna untuk menyalin isi *file1.txt* ke *file2.txt*

```
void *Thread1 (void *args)
{
    int *tanda = (int *) args;
    FILE *inp, *out;
    *tanda=1;

    inp = fopen("file1.txt","r");
    out = fopen("file2.txt","w");
    char temp;

    while(1)
    {
        temp=fgetc(inp);
        if(temp==EOF) break;
        else fputc(temp,out);
    }

    fclose(inp);
    fclose(out);
    *tanda=2;
}
```

Fungsi **Thread2**, berguna untuk menyalin isi *file2.txt* ke *file3.txt*

```
void *Thread2 (void *args)
{
    int *tanda = (int *) args;
    FILE *inp, *out;

    while(1)
    {
        if(*tanda==0) continue;
        else
        {
            inp=fopen("file2.txt","r");
            out=fopen("file3.txt","w");
            break;
        }
    }
    char temp;

    while(1)
    {
        temp=fgetc(inp);
        if(temp==EOF)
        {
            if(*tanda==1) continue;
            else break;
        }
        else fputc(temp,out);
    }

    fclose(inp);
    fclose(out);
}
```



- Main function dari program

```
void main()
{
    int tc;

    do
    {
        printf("1. Menghitung Bilangan Prima kurang dari N\n");
        printf("2. Aplikasi Multithread untuk menyalin isi
file\n");
        printf("0. Keluar\n");
        printf("Masukkan pilihan: ");
        scanf("%d",&tc);
        if(tc==1)
        {
            int n;

            printf("Masukkan nilai N: ");
            scanf("%d",&n);

            int *pointer=&n;

            pthread_t thread;
            pthread_create(&thread,NULL,Prima,(void *)pointer);
            pthread_join(thread,NULL);

            printf("Jumlah bilangan prima kurang dari %d
adalah %d\n",n,counter);
        }
        else if(tc==2)
        {
            int tanda=0;
            pthread_t t1, t2;
            pthread_create(&t1,NULL,Thread1,&tanda);
            pthread_create(&t2,NULL,Thread2,&tanda);
            pthread_join(t1,NULL);
            pthread_join(t2,NULL);
        }
        else if(tc==0)
        {
            break;
        }
        else printf("Input Salah!!!\n");
    } while(tc!=0);
}
```

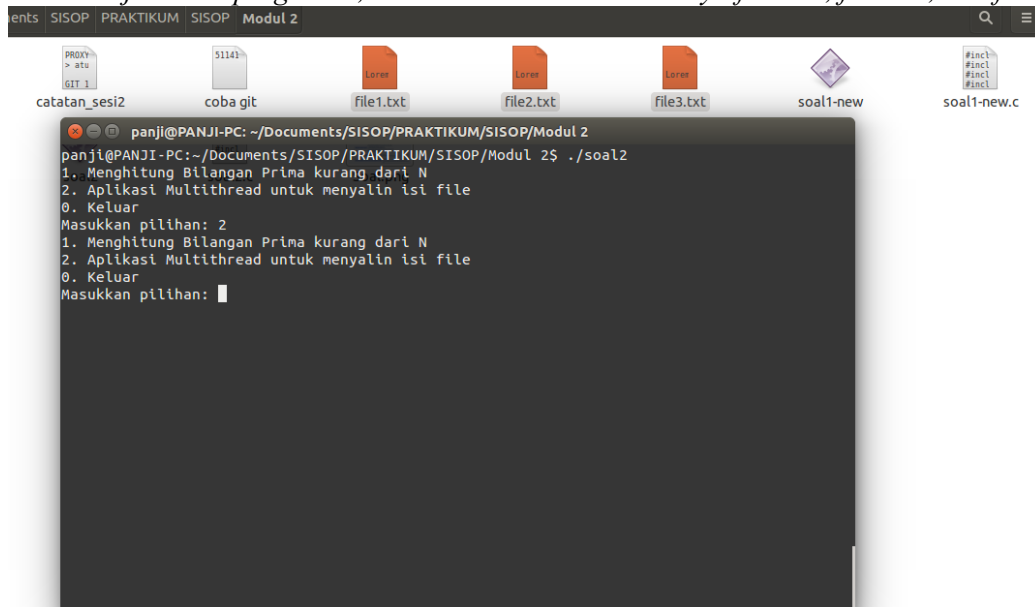


b. Output program ketika dijalankan

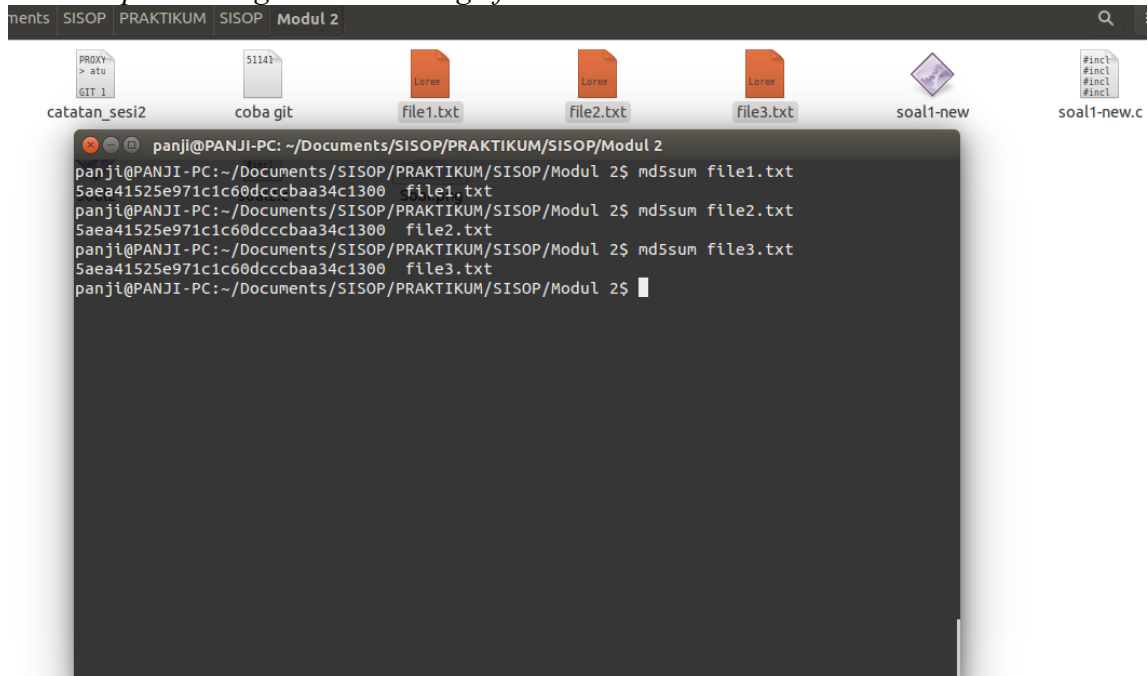
Menjalankan program 1

```
panji@PANJI-PC: ~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$ ./soal2
1. Menghitung Bilangan Prima kurang dari N
2. Aplikasi Multithread untuk menyalin isi file
0. Keluar
Masukkan pilihan: 1
Masukkan nilai N: 1000
Jumlah bilangan prima kurang dari 1000 adalah 168
1. Menghitung Bilangan Prima kurang dari N
2. Aplikasi Multithread untuk menyalin isi file
0. Keluar
Masukkan pilihan: 1
Masukkan nilai N: 2000
Jumlah bilangan prima kurang dari 2000 adalah 303
1. Menghitung Bilangan Prima kurang dari N
2. Aplikasi Multithread untuk menyalin isi file
0. Keluar
Masukkan pilihan: 1
Masukkan nilai N: 3000
Jumlah bilangan prima kurang dari 3000 adalah 430
1. Menghitung Bilangan Prima kurang dari N
2. Aplikasi Multithread untuk menyalin isi file
0. Keluar
Masukkan pilihan: █
```

Menjalankan program 2, serta melihat terbentuknya file1.txt, file2.txt, dan file3.txt



Ketika ketiga file dicek menggunakan perintah `md5sum`, melihatkan bahwa perbandingan isi dari ketiga file tersebut sama.



The screenshot shows a Linux desktop environment. The top panel displays the breadcrumb path: `ments` `SISOP` `PRAKTIKUM` `SISOP` `Modul 2`. The desktop contains several icons: `catatan_sesi2` (a document icon), `coba git` (a document icon), `file1.txt` (a text file icon), `file2.txt` (a text file icon), `file3.txt` (a text file icon), `soal1-new` (a document icon), and `soal1-new.c` (a document icon). A terminal window is open, showing the following commands and output:

```
panji@PANJI-PC: ~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$ md5sum file1.txt
5aea41525e971c1c60dcccbaa34c1300  file1.txt
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$ md5sum file2.txt
5aea41525e971c1c60dcccbaa34c1300  file2.txt
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$ md5sum file3.txt
5aea41525e971c1c60dcccbaa34c1300  file3.txt
panji@PANJI-PC:~/Documents/SISOP/PRAKTIKUM/SISOP/Modul 2$
```

