LAPORAN PRAKTIKUM 2

SISOP E07

Oleh :

1. Afifah Asmar Sari (5114100154)
2. Ilyas Bintang P. (5114100157)

1. **Membuat Shell**

Syarat :

- Berpindah direktori aktif
- Menjalankan perintah di /bin dan /usr/bin
- Tidak berhenti ketika ditekan ctrl-C dan ctrl-Z
- Otomatis keluar ketika menekan ctrl-D
- Menjalankan perintah secara background
- Tidak menggunakan system

Langkah-langkah :

- Pertama kita deklarasikan library yang akan dibutuhkan

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<stdlib.h>
#include<signal.h>
#include<sys/wait.h>
```

Stdio dan unistd merupakan library untuk mengakses input dan output. Library string digunakan untuk fungsi strtok serta strcmp. Library stdlib diperlukan saat dilakukan forking proses. Library signal digunakan untuk fungsi signal handler dan fungsi sys/wait digunakan saat proses induk menunggu proses anak.

- Kita deklarasikan sebuah variable string guna menyimpan command yang diinputkan oleh user. Selain itu, kita akan menjalankan program tersebut secara berulang selama kita tidak menekan ctrl-D karena itu kita memerlukan looping. Lalu, seperti halnya dengan shell lain yakni menampilkan header.

  Deklarasi :

```
char command[100];
char *subcommand[20];
int position=0;
```

  Header :

```
gethostname(computername,100);
getcwd(directory,100);

printf("%s@%s:~%s$ ",getenv("LOGNAME"),computername,directory);
```

Input :

```
if(fgets(command,100,stdin)==NULL)
{
        printf("\n");
        break;
}
```

- Selanjutnya kita buat fungsi sebagai signal handler untuk ctrl-C dan ctrl-Z
  Signal handler :

```
signal(SIGINT,signal_handler);
signal(SIGTSTP,signal_handler);
```

Fungsi :

```
void signal_handler()
{
       fprintf(stderr,"\n%s@%s:~%s$ ",getenv("LOGNAME"),computername,directory);
}
```

- Membagi command menjadi beberapa sub command guna memperhatikan argument yang ada. Karena command di deklarasikan sebagai string maka kita menggunakan strtok()

```
subcommand[position]=strtok(command," ");
for(position=1;position<20;position++)
{
        subcommand[position] = strtok(NULL," ");
        if(subcommand[position]==NULL) break;
}
```

- Selanjutnya kita memastikan apakah command tersebut merupakan command berpindah direktori atau bukan. Yakni dengan melihat indeks pertamanya. Pastikan indeks setelahnya tidak null dan merupakan file serta folder yang memang ada pada komputer. Untuk menjalankan perintah cd, kita gunakan chdir(argumen). Fungsi tersebut memiliki return value 0 jika sukses, jika tidak maka dapat ditarik kesimpulan bahwa tidak ada file atau folder yang diinginkan pada komputer.

```
if(strcmp(subcommand[0],"cd")==0)
{
        if(subcommand[1]==NULL) fprintf(stderr,"cd: No argument\n");
        else
        {
                if(chdir(subcommand[1])!=0) fprintf(stderr,"cd: No such file or directory\n");
        }
        return 1;
}
```

- Jika bukan merupakan perintah berpindah direktori, maka kita cek apakah perintah tersebut berjalan pada foreground atau background. Selanjutnya buat proses anak.

```
int bg=0;
if(strcmp(subcommand[position-1],"&")==0) bg=1;
pid_t pid = fork();
```

- Selanjutnya, jika command merupakan foreground maka pada proses anak akan dijalankan fungsi execvp. Jika command berjalan pada background, maka pada proses anak, & akan dirubah menjadi NULL agar dapat dijalankan di fungsi execvp.

```
if(pid==0)
{
        if(bg==1) subcommand[position-1]='\0';
        execvp(subcommand[0], subcommand);
        return 1;
}
```

- Kemudian, pada proses induk, jika ia foreground maka akan memanggil fungsi waitpid. Sedangkan apabila ia background maka proses induk tidak perlu menunggu fungsi anak.

```
if(bg==1);
else
{
        int status;
        waitpid(pid,&status,0);
}
return 1;
```

Program ini akan berjalan selama user tidak menekan tombol ctrl+D. Inputan yang dimasukkan sebelumnya akan tertimpa dengan inputan baru.

2. Mencari Jumlah Bilangan Prima Kurang dari N

Langkah-langkah :

- Pertama-tama kita tentukan library yang nantinya diperlukan dalam akses fungsi

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

Yang membedakan library pada soal nomor dua ini dengan soal sebelumnya adalah library pthread. Yakni sebuah library yang mutlak diperlukan saat kita menggunakan thread dalam program kita.

- Kemudian, kita buat fungsi yang nantinya dijalankan oleh thread. Tentunya fungsi ini merupakan operasi pencarian bilangan prima sesuai input yang diberikan user.

```
void *Hitung_Prima(void *n)
{
        int counter=0,i,j;
        int *a=(int *)n;
        for(i=2;i<*a;i++)
        {
                for(j=1;j<=i;j++) if(i%j==0) counter++;
                if(counter==2)
                {
                        printf("%d ",i);
                        prima++;
                }
                counter=0;
        }
        printf("\n");
}
```

- Barulah kita masuk pada fungsi main dimana kita membuat thread terlebih dahulu, baru membuat fungsi main menunggu fungsi perhitungan bilangan prima selesai. Dan tahap terakhir adalah menampilkan hasilnya.

```
void main()
{
        int N;
        scanf("%d",&N);
        pthread_t Bilangan_Prima;
        pthread_create(&Bilangan_Prima, NULL, Hitung_Prima, (void *)&N);
        pthread_join(Bilangan_Prima, NULL);
        printf("Jumlah bilangan prima kurang dari %d adalah %d\n",N,prima);
}
```

3. Membuat Aplikasi Multithread untuk Menyalin Isi File

Syarat :

- Ada 2 thread
- Thread 1 berfungsi membaca file dan menuliskannya pada file salinan 1
- Thread 2 berfungsi membaca file salinan 1 dan menuliskannya pada file salinan 2

Langkah-langkah :

- Library yang digunakan lebih sederhana daripada dua soal sebelumnya yakni sebagai berikut.

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

- Pertama, kita buat struct terlebih dulu. Hal ini dikarenakan yang dapat dioper pada thread hanya 1 variabl saja. Sedangkan terdapat 2 variable yang ingin dioper. Oleh karena itu kita buat struct yang berisi 2 variable tersebut.

```
typedef struct copy
{
        struct copy *in, *out;
} copy;
```

- Kemudian kita buat fungsi dari menyalin isi file

```
void *jalan1 (void **args)
{
        copy *file=(copy *)*args;
        char line[100];
        while(fgets(line, 100, (*file)->in) != NULL)
        {
                fputs(line, (*file)->out);
        }

}
```

- Selanjutnya kita set pointer step1->in untuk menunjuk file.txt dengan akses read dan mengeset pointer step1->out untuk menunjuk salinan1.txt dengan akses read&write. Lalu inisialisasi pointer struct step2, dengan step->in=step->out pointer yang menunjuk salinan1.txt dengan akses read&write. Kemudian, kita melakukan fseek yang akan menggeser pointer of file yang menujuk end of file salinan1.txt ke awal file dan dengan step->out menunjuk salinan2.txt dengan akses write lalu membuat thread dengan fungsi yang sama dengan step 1.

```
int main()
{
        pthread_t t1, t2;
        copy *step1, step2;
        step1->in = fopen("file.txt", "r");
        step2->out = fopen("file.txt", "r+");
        pthread_create(&t1, NULL, jalan1, (void *)&step1);

}
```