Laporan

Nama : Muhammad Hanif (5114100013)
Ahmad Ismail (5114100032)

Dokumentasi :

1. Membuat shell dalam bahasa C

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

char cmd[100];
char *bagian[20];
int posisi=0;
char namakomputer[100];
char namadirektori[100];

int changedir()
{
        if (bagian[1]==NULL) chdir(getenv("HOME"));
                else{
                        if(chdir(bagian[1])!=0) fprintf(stderr,"cd: No such file or directory\n");
                    }
                return 0;
}

int jalan(char** bagian,int posisi)
{
        pid_t pid;
        int cek=0;
        if(strcmp(bagian[0],"cd")==0)
        {
                int hasil;
                hasil = changedir();
                return hasil;
        }
        else
        {
                if(strcmp(bagian[posisi-1],"&")==0) cek=1;
                pid = fork();
                if(pid==0)
                {
                        if(cek==1) bagian[posisi-1]='\0';
                        execvp(bagian[0], bagian);
                        return 0;
                }
                else if(pid<0)
                {
                        printf("ERROR\n");
                        return -1;
                }
                else
void signalhandler()
{
        fprintf(stderr,"\n%s@%s:~%s$ ",getenv("LOGNAME"),namakomputer,namadirektori);
}

int main()
{

        while(1){
                gethostname(namakomputer,100);
                getcwd(namadirektori,100);
                printf("%s@%s:~%s$ ",getenv("LOGNAME"),namakomputer,namadirektori);
                signal(SIGTSTP,signalhandler);
                signal(SIGINT,signalhandler);
                if(fgets(cmd,100,stdin)==NULL){
                        printf("\n");
                        break;
                }
                if(cmd[strlen(cmd)-1]=='\n') cmd[strlen(cmd)-1]='\0';
                bagian[posisi]=strtok(cmd," ");
                for(posisi=1;posisi<20;posisi++){
                        bagian[posisi] = strtok(NULL," ");
                        if(bagian[posisi]==NULL) break;
                }
                if(strcmp(cmd, "exit")==0) break;
                else if(jalan(bagian,posisi)==-1) break;

        }
        return 0;
}
```

- Deklarasi gethostname untuk mencari nama komputer. Lalu, getcwd berfungsi untuk mencari nama direktori. LOGNAME merupakan Format print seperti terminal yaitu berisi tempat di mana kita berada.
- SIGNAL berfungsi untuk mencegah interupsi/berhentinya program dari Ctrl+C dan Ctrl+Z.
- Lalu, fgets cmd stdin berfungsi untuk keluar dari shell apabila menekan tombol Ctrl+D.
- Lalu, loop selanjutnya berfungsi untuk memecah string input agar dapat dibaca sebagai command. Apabila user mengetikkan “exit”, maka akan keluar dari shell. Apabila tidak, maka command akan dijalankan di fungsi jalan
- Di dalam fungsi jalan, akan dicek strcmp apakah instruksi berupa “cd”. Apabila iya, maka akan masuk ke fungsi changedir. Di fungsi changedir, dicek lagi apakah terdapat argumen tambahan atau tidak dibelakang “cd”. Apabila tidak ada, maka direktori akan kembali ke home. Apabila ada, dia akan berpindah ke direktori yang diinginkan. Namun, apabila direktori yang diinginkan tidak ada/tidak tersedia, maka akan muncul pesan “error” dan lanjut return 0;
- Selanjutnya, dia mengecek apakah command diakhiri dengan ampersand (“&”) atau tidak. Apabila iya, maka cek akan bernilai 1. Lalu, menjalankan fork dan mengecek pid. Apabila pid sama dengan 0, akan dicek apakah ceknya sama dengan 1 atau tidak. Apabila sama dengan 1, maka dia akan berjalan di background process (bagian[posisi-1]='\0'). Apabila tidak, maka akan menjalankan execvp. Execvp mengeksekusi program yang ada di /bin dan /usr/bin dan akan return 0.
- Apabila pid<0, akan muncul pesan “ERROR” dan return -1.
- Selain pid=0 dan pid<0, maka dia mengecek lagi apakah cek==1. Apabila cek==1, maka dia tidak akan melaksanakan wait(berjalan di background process). Apabila cek!=1, maka dia akan melaksanakan wait.
- Fungsi dari return tersebut adalah untuk mengecek apakah proses itu error atau tidak. Jika error dia akan break, jika tidak akan melaksanakan while lagi.

2. Membuat program pencari bilangan prima

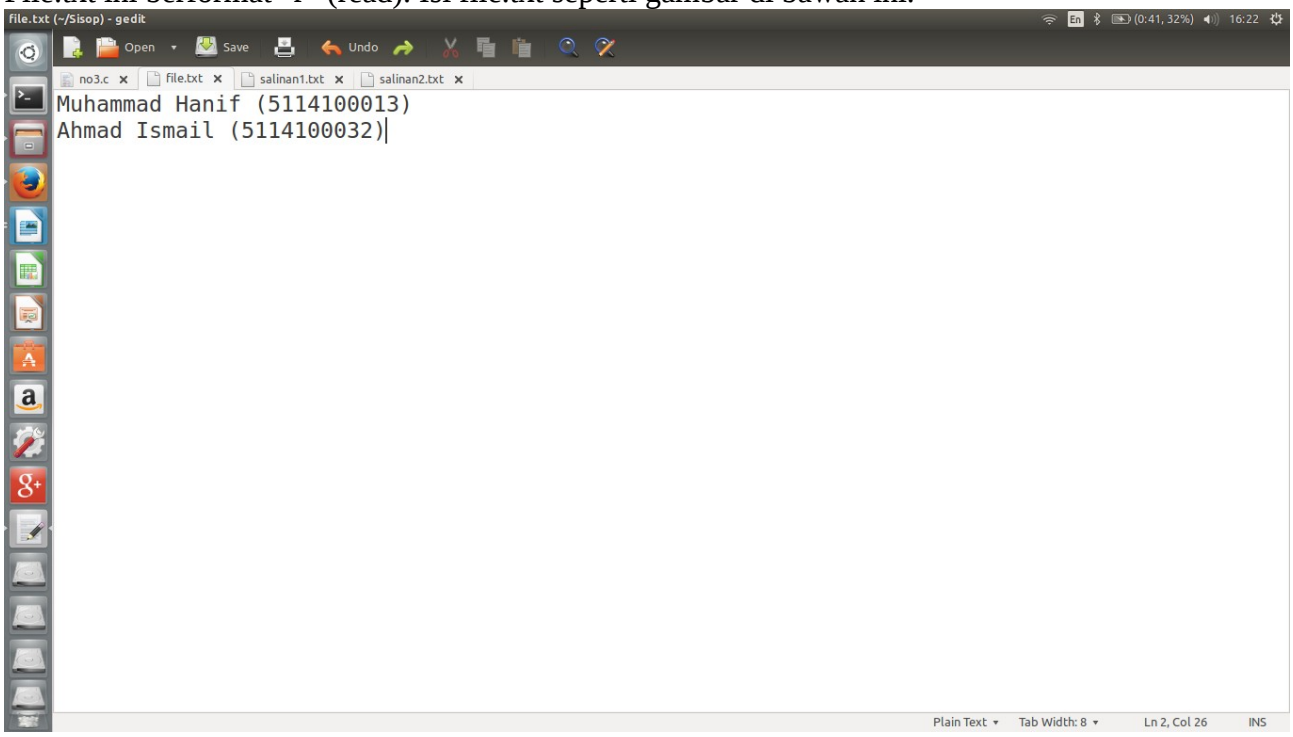
```c
#include<stdio.h>
#include<pthread.h>

int sum;
int total;

void *prime_thread(void *args)
{
        int i, n,count=0;
        n=*((int*)args);
        for(i=1; i<=n; i++)
        {
                if(n%i==0)
                {
                        count++;
                }
        }
        if(count==2)
        {
                sum = sum + 1;
                printf ("%d prima\n",n);
                total=total+n;
        }
        else printf("%d bukan prima\n",n);
}

int main()
{
        int i, input;
        printf("Masukkan Nilai : ");
        scanf("%d",&input);
        pthread_t a[input];
        int b[input];
        for(i=2; i<input; i++)
        {
                b[i]=i;
                pthread_create(&a[i],NULL,prime_thread,&b[i]);
        }
        for(i=2; i<input; i++)
        {
                pthread_join(a[i],NULL);
        }
        printf("Total bilangan prima Kurang dari %d ada %d\n",input, sum);
        printf("Total semua nilainya adalah %d\n",total);
        return 0;
}
```

- Deklarasi input sebagai nilai masukkan yang akan dimasukkan oleh user.
- Thread yang akan dibuat adalah input yang dimasukkan user dan memulai loop dari 2 < input.
- Setelah itu menjalankan thread_create, nilai looping tersebut yang disimpan di dalam variabel b akan dipassing ke variabel n. Lalu, masuk ke fungsi prime_thread.
- Di fungsi tersebut, terdapat variabel n yang tadi telah dipassing oleh variabel b. Lalu, mulai looping dari dari i=1 < = n.
- Nilai prima adalah apabila nilai tersebut dapat dibagi 1 dan nilai tersebut, maka terdapat 2x operasi pembagian dengan hasil modulus == 0. Lalu, counter akan bertambah. Apabila counter == 2, sum= sum+1 yang berfungsi untuk menghitung jumlah bilangan prima, lalu mencetak bilangan prima dan total=total+n berfungsi untuk menjumlahkan semua bilangan prima.
- Apabila bukan bilangan prima, maka akan mencetak bukan prima.
- Lalu, fungsi thread_join adalah untuk menunggu thread lain sebelum program selesai.
- Hasil yang akan ditampilkan adalah nilai-nilai yang tidak terurut, karena menggunakan thread dan selalu random.
- Di akhir, akan mencetak total bilangan prima kurang dari input sebanyak jumlah bilangan prima dan mencetak hasil penjumlahan semua bilangan prima.

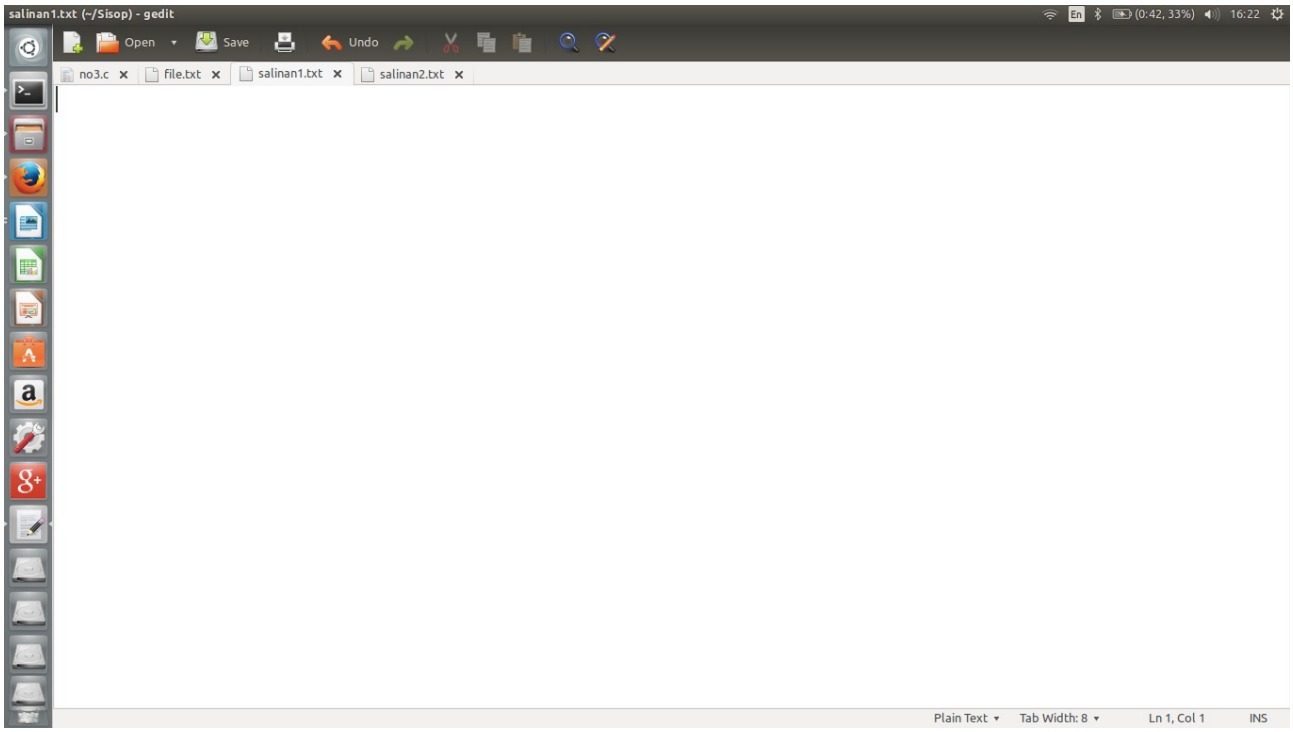
3. Membuat backup file/menyalin isi file

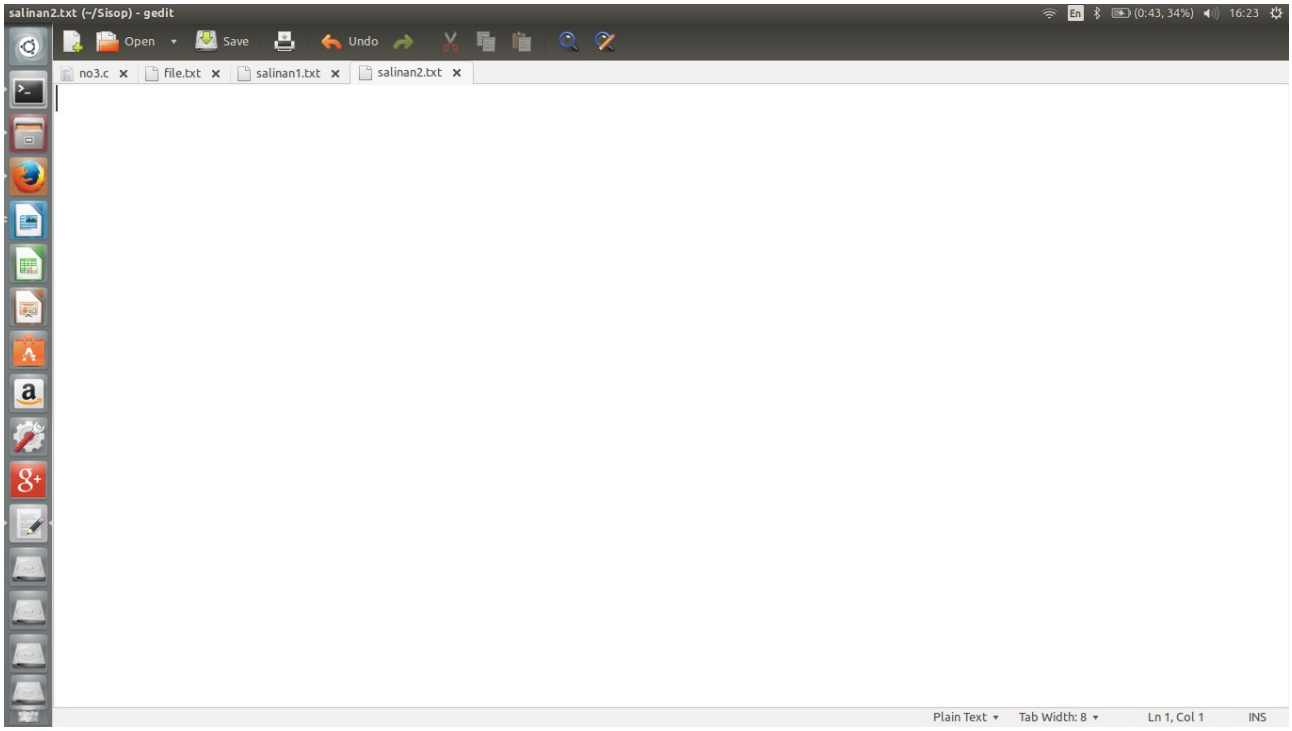
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *task1 ()
{
        FILE *baca, *salin;
        baca = fopen("file.txt", "r");
        salin = fopen("salinan1.txt", "w");

        char kar;
        while((kar=fgetc(baca))!=EOF)
        {
                //fprintf(salin, "thread 1 : ");
                fputc(kar,salin);
                //fprintf(salin, "\n");
        }
        fclose(baca);
        fclose(salin);
}

void *task2 ()
{
        FILE *in, *out;
        in = fopen("salinan1.txt","r");
        out = fopen("salinan2.txt","w");

        char isi;;
        while((isi=fgetc(in))!=EOF)
        {
                //fprintf(out, "thread 2 : ");
                fputc(isi,out);
                //fprintf(out, "\n");
        }
        fclose(in);
        fclose(out);
}

void main()
{
        pthread_t t1, t2;
        pthread_create(&t1, NULL, task1, NULL);
        pthread_join(t1, NULL);
        pthread_create(&t2, NULL, task2, NULL);
        pthread_join(t2, NULL);
}
```

- Deklarasi dalam void main ada 2 task yang digunakan, yaitu task 1 (untuk menjalankan instruksi 1) dan task 2 (untuk menjalankan instruksi 2). Lalu, masuk ke pthread_create yang pertama yang akan menjalankan task 1.
- Di dalam task 1 terdapat fungsi bernama variabel baca untuk membuka file yang bernama file.txt. File.txt ini berformat “r” (read). Isi file.txt seperti gambar di bawah ini.

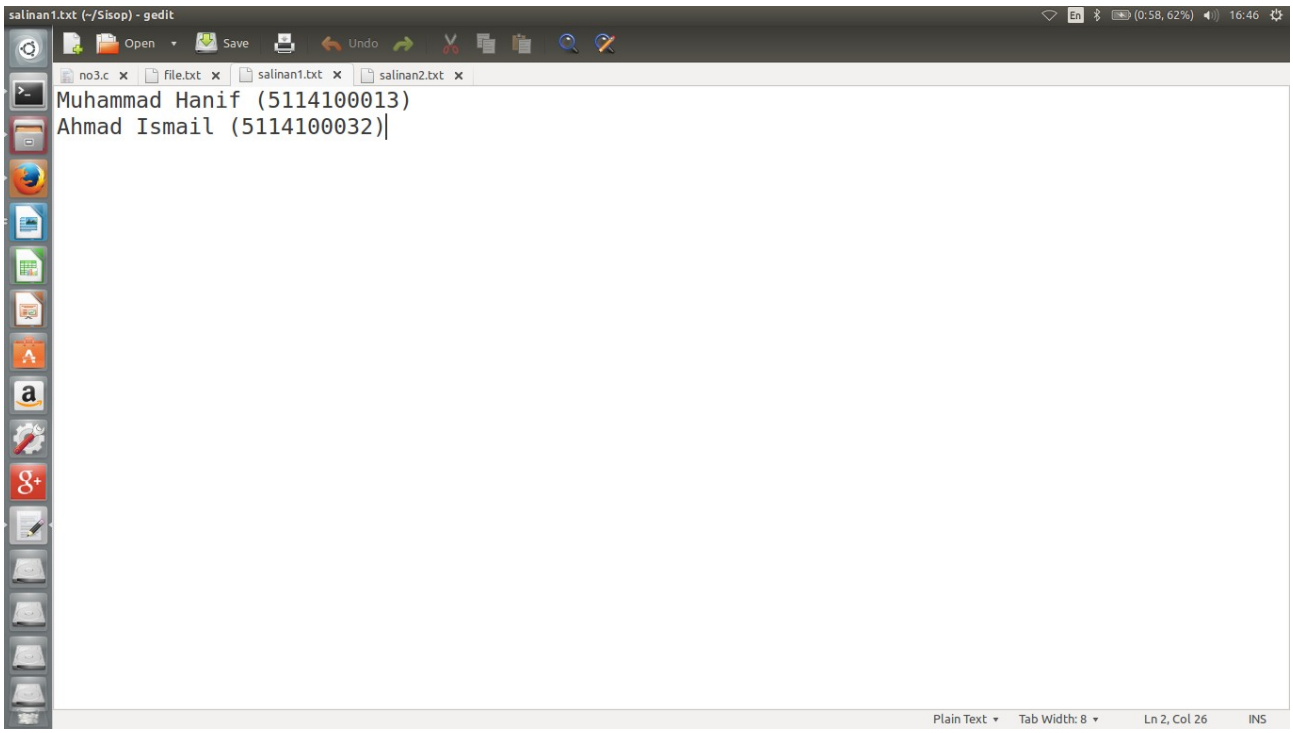


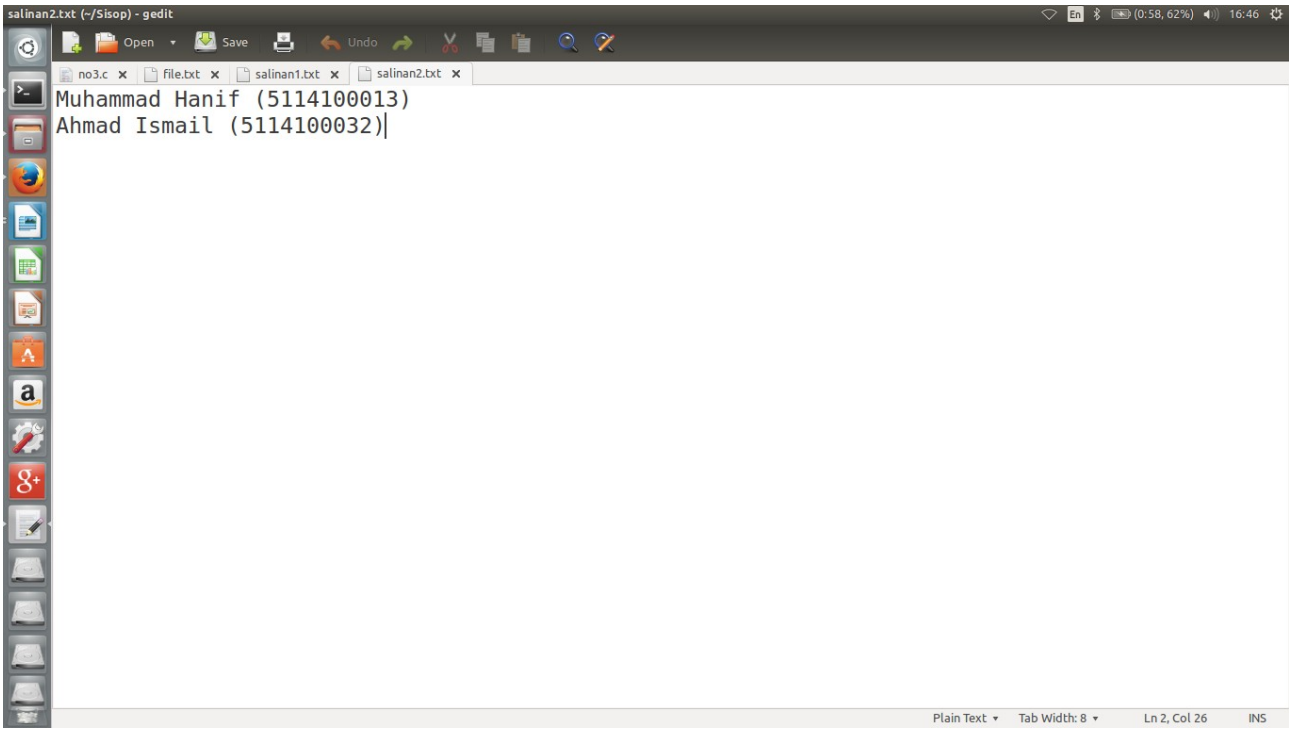
- Lalu, menjalankan fungsi bernama variabel salin salinan1.txt yang berformat “w” (write).
- Isi dari salinan 1.txt dan salinan 2.txt masih kosong, seperti gambar di bawah ini.
- Deklarasi char kar yang berfungsi untuk menyimpan karakter dari file yang ingin dibuka nanti.





- Setelah itu, masuk ke looping while. Di dalam while terdapat instruksi untuk membaca file samping akhir dari file tersebut (EOF).
- Lalu, file tersebut akan mencetak (fputc) dengan alamat dari char kar dan instruksi salin.
- Setelah itu, menjalankan instruksi fclose yang berfungsi untuk menutup file dari file yang telah terbuka tersebut.
- Task 1 telah selesai dari fungsi pthread_create t1. Lalu, masuk ke fungsi pthread_join t1 yang berfungsi untuk menunggu giliran thread lain sebelum program selesai.
- Task 2 sama halnya dengan task 1, yang akan bekerja untuk menyalin isi file dari file.txt ke salinan2.txt. Yang berbeda hanya variabel yang digunakan agar tidak terjadi error apabila terdapat variabel yang sama.
- Kedua task ini melaksanan instruksinya secara random, karena menggunakan thread.
- Hasil akhir dari program tersebut adalah akan tercetak isi dari file.txt ke salinan1.txt dan salinan2.txt, seperti gambar di bawah ini.


salinan1.txt (~/Sisop) - gedit
Open
Save
Undo
no3.c
file.txt
salinan1.txt
salinan2.txt
Muhammad Hanif (5114100013)
Ahmad Ismail (5114100032)
Plain Text
Tab Width: 8
Ln 2, Col 26
INS


salinan2.txt (~/Sisop) - gedit
Open
Save
Undo
no3.c
file.txt
salinan1.txt
salinan2.txt
Muhammad Hanif (5114100013)
Ahmad Ismail (5114100032)
Plain Text
Tab Width: 8
Ln 2, Col 26
INS