

**-MANUAL-
PANDUAN
RANGKUMAN CATATAN SESI
LAB
SISTEM OPERASI E
ZAHRAH CITRA HAFIZHA
5114100012
FIKRY KHAIRYTANIM 5114100192**

GITHUB

Github adalah control sytem, dimana akan mengontrol jalannya suatu program dan mencatat perkembangan dan perubahan-perubahan suatu program, dari awal dibuat hingga program selesai dibuat.

Github ini bisa digunakan oleh beberapa contributors. Dalam kesalahan yang terjadi dalam pembuatan program pun bisa dilihat contributor mana yang melakukan kesalahan. Kesalahan dalam pembuatan program pun bisa di revert. Setiap project program akan tersimpan di repository.

Tahap membuat git:

1. Buat repository. Di bagian gitignore, pilih C supaya compilernya tidak perlu di save namun hanya code nya saja
2. Install git dengan perintah **sudo su apt-git install git**
3. Lalu masuk ke master, copy **https.url** di ujung kanan bawah
4. Untuk meng-clone repo dengan contributors, setting proxy terlebih dahulu
5. Lalu clone url yang tadi di copy dengan memasukkan perintah **git clone url** di terminal
6. Lalu tambahkan colaborator dengan teman anda.

Untuk mencoba membuat file program:

1. buat file.txt (missal contributor.txt)
2. ketik **git config** di terminal, berguna untuk setting email
3. ketik **git add kontributor.txt**, untuk memasukkan file ke dalam github
4. cek status dengan **git status**
5. lalu **git commit** untuk konfirmasi, buatlah comment yang jelas agar tidak terjadi kekeliruan
6. lalu push dengan **git push**
7. bagi user ke-2 (yang meng-clone repo yg user 1) kalo tidak bisa git push, lakukan git pull, berguna supaya menyamakan hal yang ada di repo. Lalu lakukan git push

Cara untuk mengedit coding:

1. git pull
2. liat apa yg di cek
3. git add kontributor.txt
4. git commit "comment sesuatu"
5. git push

Perintah-perintah yang akan selalu dipakai di git:

1. git add namafile : menambahkan file untuk di track
2. git comit -a : meyetujui perubahan
3. git push : upload ke repo
4. git pull : download yang terbaru dr repo

Note: pastikan setiap push berikan keterangan yang deskriptif, tiap mau memperbarui program harus pull dulu, dan pastikan pull di file yang terbaru agar tidak terjadi error.

STRING PROCESSING

Perintah yang perlu diketahui:

man strstr : locate a substring. return value nya pointer.

man strtok : extract token from string

berguna untuk parsing sesuatu

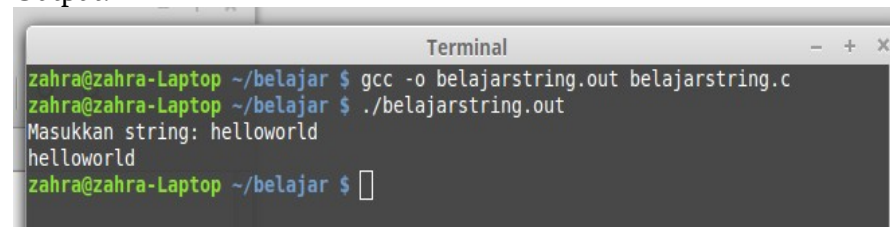
Contoh program string processing:

Contoh 1:

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

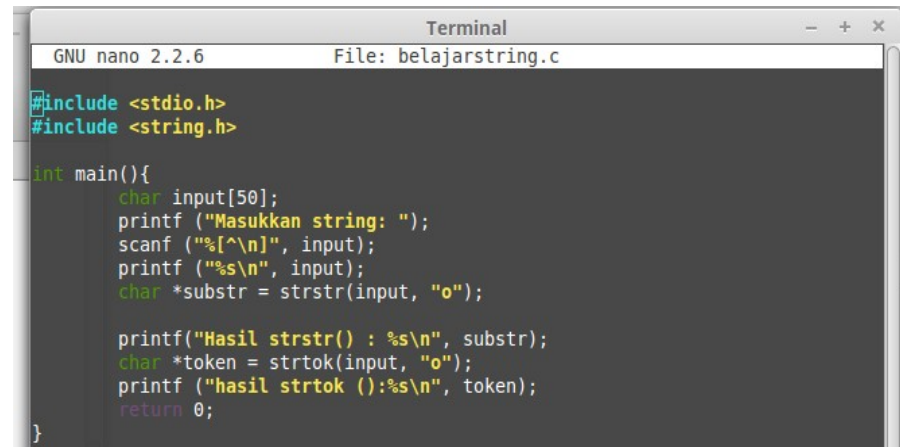
int main(){
    char input[50];
    printf ("Masukkan string: ");
    scanf ("%s", input);
    printf ("%s", input);
    return 0;
}
```

Output:



```
Terminal
zahra@zahra-Laptop ~/belajar $ gcc -o belajarstring.out belajarstring.c
zahra@zahra-Laptop ~/belajar $ ./belajarstring.out
Masukkan string: helloworld
helloworld
zahra@zahra-Laptop ~/belajar $
```

Contoh 2:



```
GNU nano 2.2.6 File: belajarstring.c
#include <stdio.h>
#include <string.h>

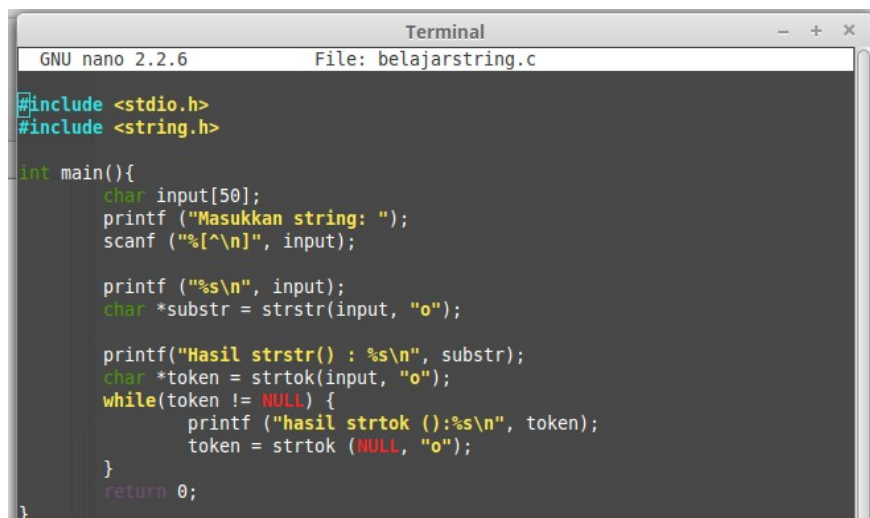
int main(){
    char input[50];
    printf ("Masukkan string: ");
    scanf ("%s", input);
    printf ("%s", input);
    char *substr = strstr(input, "o");

    printf("Hasil strstr() : %s", substr);
    char *token = strtok(input, "o");
    printf ("hasil strtok ():%s", token);
    return 0;
}
```

Output:

```
zahra@zahra-Laptop ~/belajar $ gcc -o belajarstring.out belajarstring.c
zahra@zahra-Laptop ~/belajar $ ./belajarstring.out
Masukkan string: hello world
hello world
Hasil strstr() : o world
hasil strtok():hell
zahra@zahra-Laptop ~/belajar $
```

Contoh 3:



```
GNU nano 2.2.6 File: belajarstring.c
#include <stdio.h>
#include <string.h>

int main(){
    char input[50];
    printf ("Masukkan string: ");
    scanf ("%s", input);

    printf ("%s\n", input);
    char *substr = strstr(input, "o");

    printf("Hasil strstr() : %s\n", substr);
    char *token = strtok(input, "o");
    while(token != NULL) {
        printf ("hasil strtok ():%s\n", token);
        token = strtok (NULL, "o");
    }
    return 0;
}
```

Output:

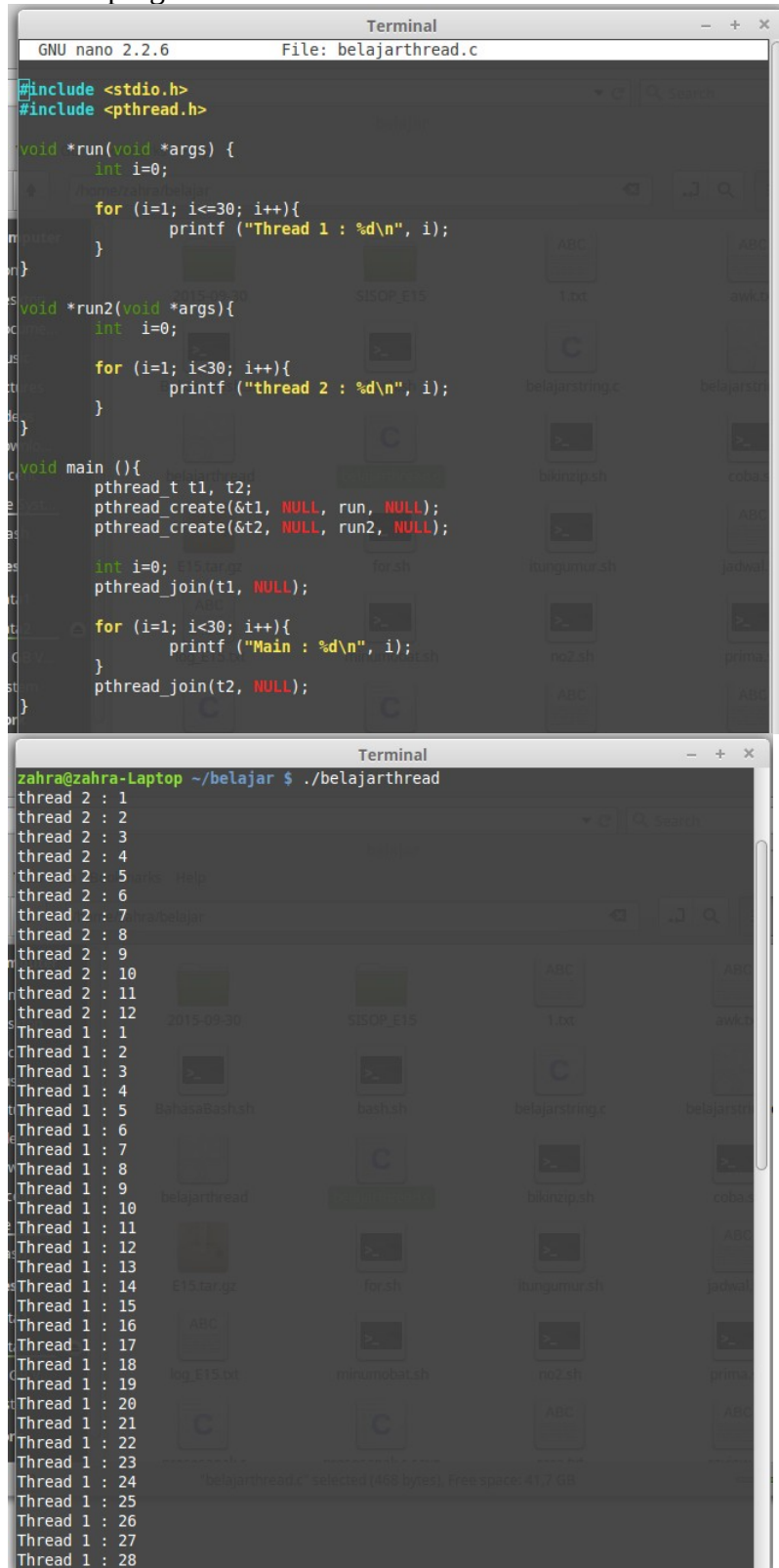
```
zahra@zahra-Laptop ~/belajar $ nano belajarstring.c
zahra@zahra-Laptop ~/belajar $ gcc -o belajarstring.out belajarstring.c
zahra@zahra-Laptop ~/belajar $ ./belajarstring.out
Masukkan string: Hallo World
Hallo World
Hasil strstr() : o World
hasil strtok():Hall
hasil strtok(): W
hasil strtok():rld
zahra@zahra-Laptop ~/belajar $
```

THREAD

Setiap thread memilih fungsi untuk dijalankan, dan fungsi tersebut akan mengikuti fungsi tersebut yang telah dipilih. Dalam membuat thread, kita membutuhkan library **POSIX** thread, dan kita butuh fungsi **pthread_create**

Di thread, kita membuahkan id thread, atribut, dan nama fungsi yang akan dijalankan. Thread tidak bisa mengatur fungsi mana yang akan jalan terlebih dahulu. Didalam thread, kita bisa menggunakan **pthread_join** = join with terminated thread

Contoh program thread:



The image displays two terminal windows. The top window shows the source code of a C program named `belajarthread.c` being edited in the `nano` text editor. The code defines two threads, `run` and `run2`, each performing a loop of 30 iterations. The `main` function creates these threads, joins them, and then prints the main thread's progress. The bottom window shows the execution of the program, with output indicating that thread 2 runs first, followed by thread 1, and finally the main thread.

```
GNU nano 2.2.6 File: belajarthread.c
#include <stdio.h>
#include <pthread.h>

void *run(void *args) {
    int i=0;
    for (i=1; i<=30; i++){
        printf ("Thread 1 : %d\n", i);
    }
}

void *run2(void *args){
    int i=0;
    for (i=1; i<30; i++){
        printf ("thread 2 : %d\n", i);
    }
}

void main (){
    pthread_t t1, t2;
    pthread_create(&t1, NULL, run, NULL);
    pthread_create(&t2, NULL, run2, NULL);

    int i=0;
    pthread_join(t1, NULL);

    for (i=1; i<30; i++){
        printf ("Main : %d\n", i);
    }
    pthread_join(t2, NULL);
}
```

```
zahra@zahra-Laptop ~/belajar $ ./belajarthread
thread 2 : 1
thread 2 : 2
thread 2 : 3
thread 2 : 4
thread 2 : 5
thread 2 : 6
thread 2 : 7
thread 2 : 8
thread 2 : 9
thread 2 : 10
thread 2 : 11
thread 2 : 12
Thread 1 : 1
Thread 1 : 2
Thread 1 : 3
Thread 1 : 4
Thread 1 : 5
Thread 1 : 6
Thread 1 : 7
Thread 1 : 8
Thread 1 : 9
Thread 1 : 10
Thread 1 : 11
Thread 1 : 12
Thread 1 : 13
Thread 1 : 14
Thread 1 : 15
Thread 1 : 16
Thread 1 : 17
Thread 1 : 18
Thread 1 : 19
Thread 1 : 20
Thread 1 : 21
Thread 1 : 22
Thread 1 : 23
Thread 1 : 24
Thread 1 : 25
Thread 1 : 26
Thread 1 : 27
Thread 1 : 28
```