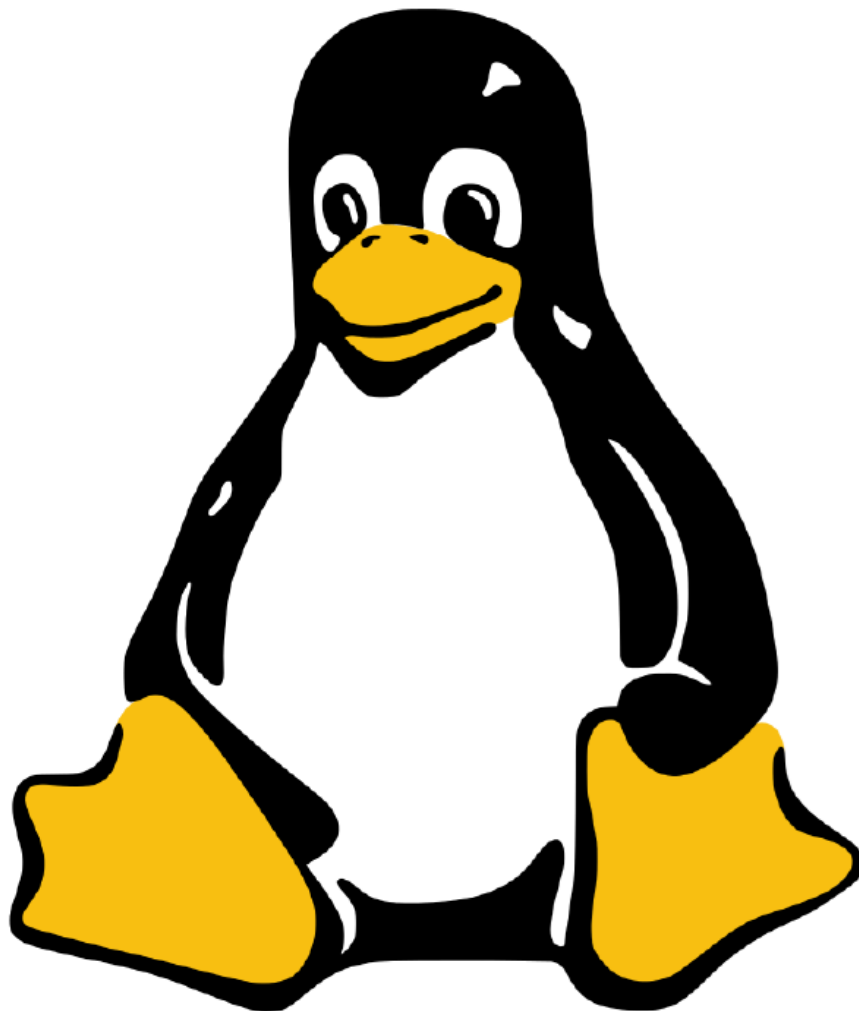


MANUAL MODUL 2

SISTEM OPERASI E

“Github, String, Fork, Thread”



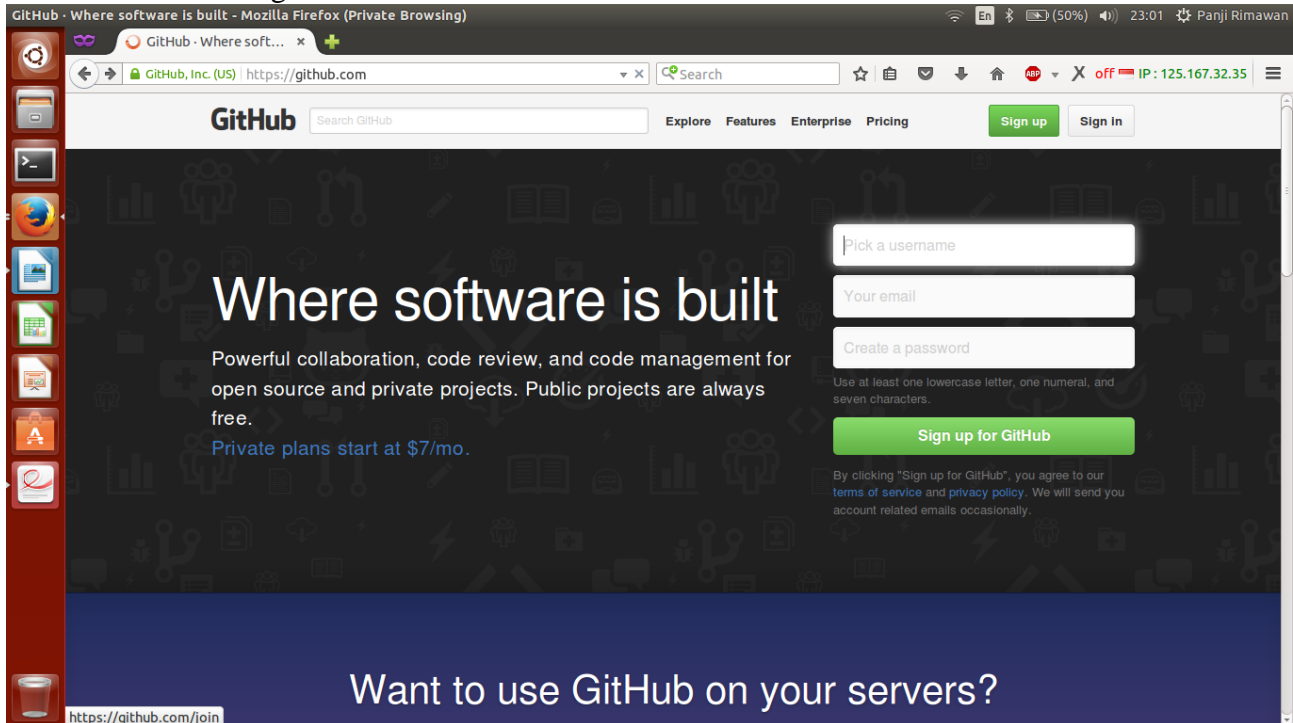
Oleh:
Kelompok E-5
Panji Rimawan (5114100075)
Steven Kurniawan (5114100100)

1. GIT

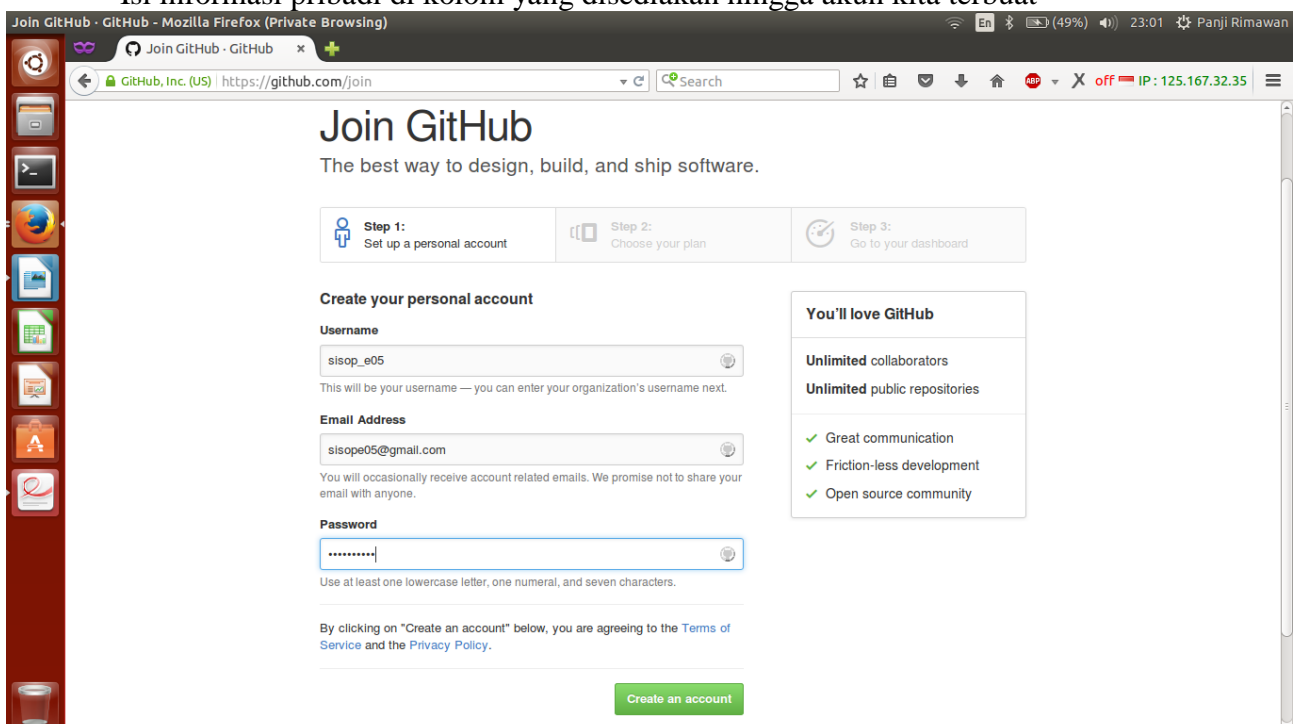
a. Pengertian GitHub?

GitHub adalah layanan penginangan web bersama untuk proyek pengembangan perangkat lunak yang menggunakan sistem pengontrol versi Git. Situs ini menyediakan fungsionalitas jejaring sosial seperti umpan web, pengikut, wiki (menggunakan perangkat lunak Wiki gollum) dan grafik jejaring sosial untuk menampilkan bagaimana para pengembang menggarap versi repositori mereka. (source: wikipedia.org)

a. Membuat akun di github.com



- Buka link github.com
- Pilih register untuk membuat akun
- Isi informasi pribadi di kolom yang disediakan hingga akun kita terbuat



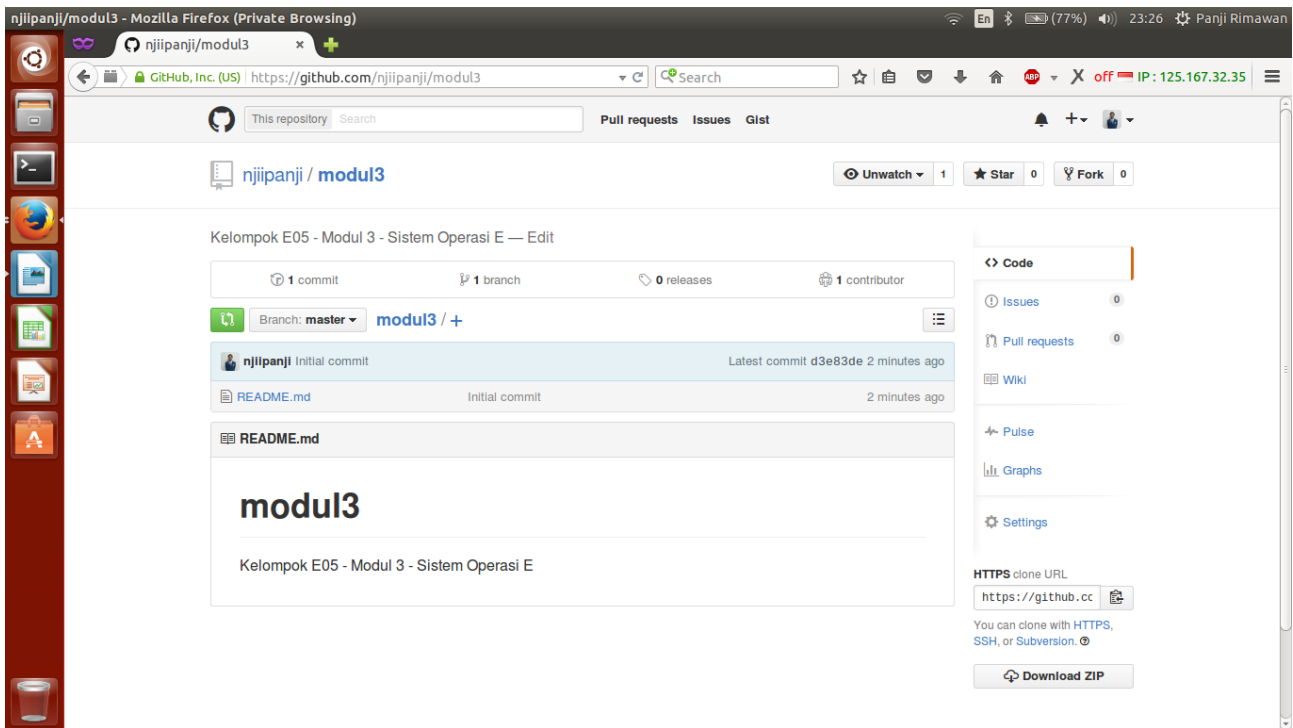
b. Membuat repositories baru di akun github

- Klik tombol “New Repository”, lalu akan muncul halaman baru yang berisikan permintaan untuk memasukkan nama repositories yang akan dibuat

The top screenshot shows the GitHub homepage in a Mozilla Firefox browser. The page features a 'GitHub Bootcamp' section with four steps: 1. Set up Git, 2. Create repositories, 3. Fork repositories, and 4. Work together. Below this is a 'Welcome to GitHub!' message and a 'Your repositories' section with a '+ New repository' button. The bottom screenshot shows the 'Create a new repository' page. It includes a form with fields for 'Owner' (njilpanji), 'Repository name' (SISOP), and 'Description (optional)'. There are also checkboxes for 'Public' (selected), 'Private', and 'Initialize this repository with a README'. At the bottom, there are buttons for 'Add .gitignore' and 'Add a license', and a green 'Create repository' button.

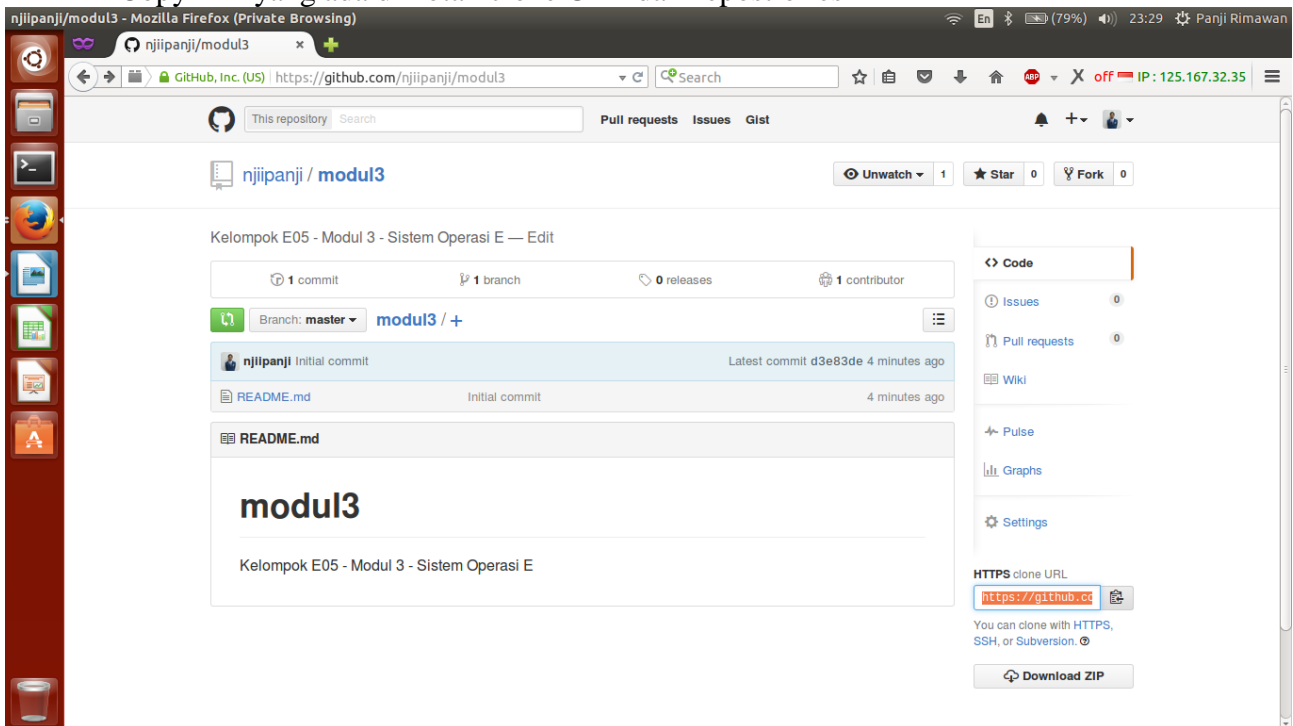
- Setelah selesai memberi nama, langkah terakhir yaitu klik tombol “Create Repository”. Maka akan muncul halaman seperti di bawah ini.





c. Menambahkan collaborator

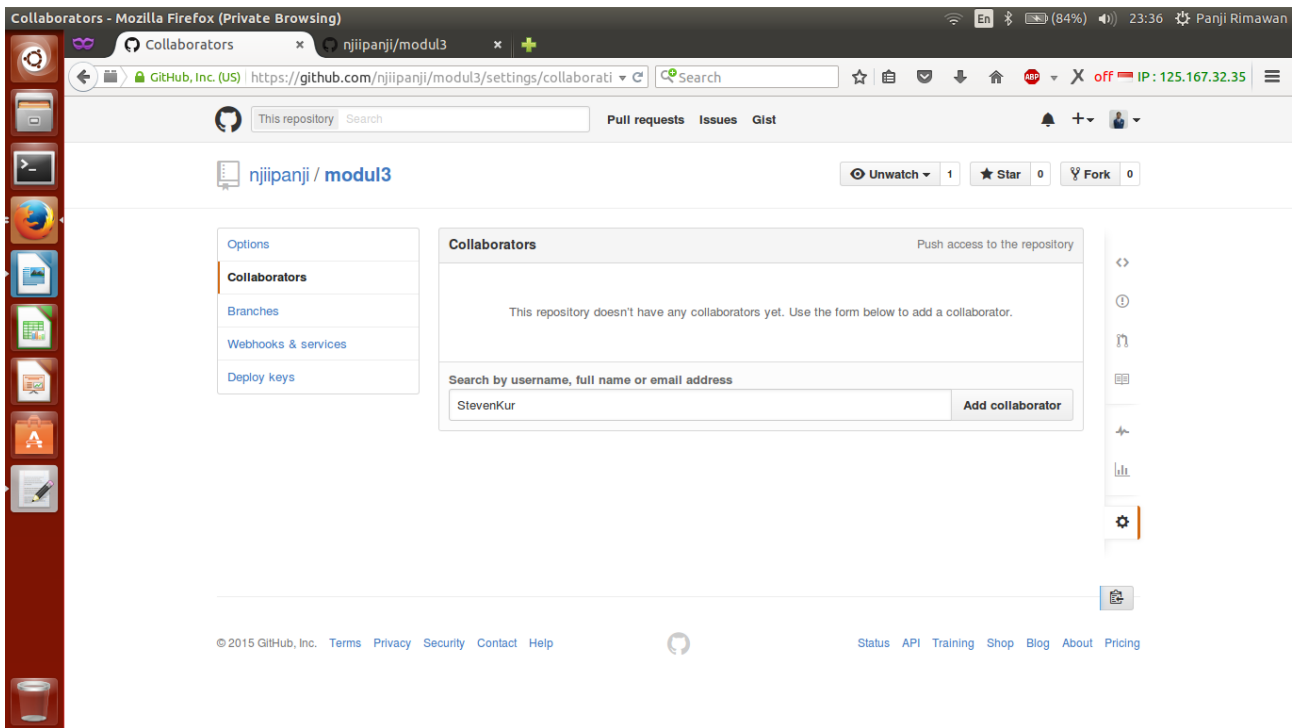
- Copy link yang ada di kotak clone URL dari repositories



- Klik settings di repositories, lalu pindah ke tab Collaborator

- Masukkan username dari teman/member lain yang 1 project, kemudian klik “Add Collaborator”

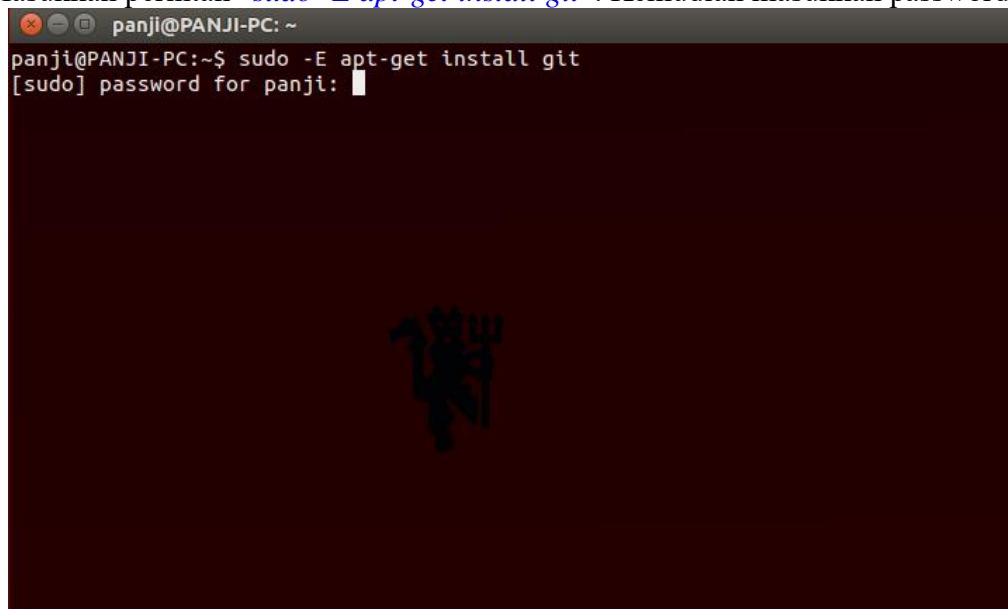




- Akan muncul username dari teman/member lain yang kita tambahkan barusan

d. Install git di linux melalui terminal

- Masukkan perintah “`sudo -E apt-get install git`”. Kemudian masukkan password.



- Cek versi git yang telah terinstall di ubuntu dengan mengetikkan perintah “`git --version`”

e. Perintah git yang dapat dieksekusi melalui terminal

- git clone *alamat_folder_repositories* -> membuat clone dari repositories ke PC user.
- git add *namafile* -> menambahkan file untuk ditrack .
- git commit -a -> menyetujui adanya perubahan dari repositories.
- git push -> upload file-file terbaru ke repositories .
- git pull -> download file-file yang terbaru dari repositories.



f. Implementasi penggunaan git di linux

1. Buat/masuk ke folder dimana kita akan menyimpan/mengunduh repositories kita ke dalam PC
2. Ketikkan “clone [alamat_folder_repositories](#)” repositories yang akan kita download ke PC. Folder clone nantinya akan otomatis menjadi folder repositories yang ada di PC. Link alamat untuk clone bisa dicopy dari repositories di github.com. Hal ini hanya dilakukan

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2
panji@PANJI-PC:~/Documents/SISOP/PRAK2$ git clone https://github.com/njlipanj1/SISOP.git
Cloning into 'SISOP'...
remote: Counting objects: 9, done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 9
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
panji@PANJI-PC:~/Documents/SISOP/PRAK2$
```

Akan muncul folder repositories di PC kita:

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2
panji@PANJI-PC:~/Documents/SISOP/PRAK2$ ls
catatan_sesi2      tesstring
catatan_sesi2~    tesstring.c
LAPORAN MODUL-2.odt thread
modulsisop2       thread.c
prima_fork        Tutorial - Write a Shell in C • Stephen Brennan_files
prima_fork.c      Tutorial - Write a Shell in C • Stephen Brennan.html
Process, Thread.png Writing Your Own Shell_files
shell.txt         Writing Your Own Shell.html
signals.c         Writing Your Own Shell LG #111_files
SISOP             Writing Your Own Shell LG #111.html
Soal.png
panji@PANJI-PC:~/Documents/SISOP/PRAK2$
```

3. Setelah itu, masukkan perintah “git pull”, untuk mendownload file-file repositories terbaru.

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/SISOP
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git pull
Already up-to-date.
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$
```



4. Jika sudah mengambil file-file terbaru, maka kita bisa memulai edit file / add file ke dalam folder tersebut untuk ditambahkan/diubah isi repositories. Jangan sampai member dari project yang sama di repositories tersebut mengedit sebuah file yang sama dalam waktu yang bersamaan (harus berbeda).

5. Setelah selesai menambahkan file / mengubah isi suatu file, kita bisa langsung menguploadnya ke repositories dengan memasukkan perintah “git add *namafile*”

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/SISOP
panji@PANJI-PC:~/Documents/SISOP/PRAK2$ cp Soal.png /home/panji/Documents/SISOP/
PRAK2/SISOP/
panji@PANJI-PC:~/Documents/SISOP/PRAK2$ cd SISOP/
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git add Soal.png
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git commit -a
[master 200d753] Contoh add file ke repositories.
1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Soal.png
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$
```

6. Kemudian dilanjutkan dengan memasukkan perintah “git commit -a” untuk membuat sebuah comment tentang perubahan/update dari file yang akan di push.

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/SISOP
GNU nano 2.2.6 File: ...ments/SISOP/PRAK2/SISOP/.git/COMMIT_EDITMSG Modified

Contoh add file ke repositories.
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#   new file:   Soal.png
#
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text    ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

7. Sebelum di push, masukkan terlebih dahulu username.email & username.git sebelum push ke git, dengan memasukkan perintah

```
git config --global user.email "email yang terdaftar di github/collaborator"
git config --global user.name "usernamegit"
```



```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/SISOP
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git config --global user.email "panjirimawan8a@gmail.com"
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git config --global user.name "njiipanji"
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$
```

8. Setelah selesai, langkah terakhir yaitu push / upload file-file yang sudah ditambahkan di folder, dengan memasukkan perintah “git push”. Saat proses, akan muncul tampilan seperti di bawah, dan kita perlu untuk memasukkan **username github** beserta **password** nya, agar file/folder yang kita add bisa ter-push.

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/SISOP
create mode 100644 Soal.png
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

    git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

    git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

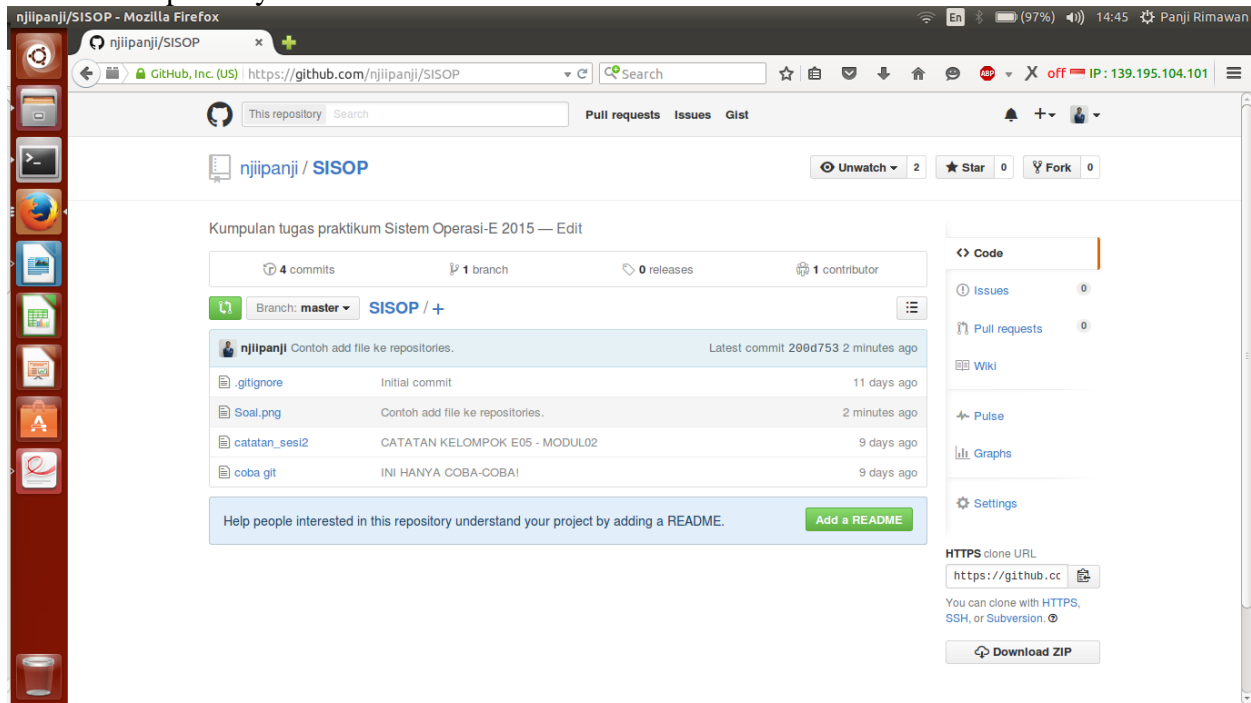
In Git 2.0, Git will default to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': njiipanji
Password for 'https://njiipanji@github.com':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 36.79 KiB | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/njiipanji/SISOP.git
 f566476..200d753 master -> master
panji@PANJI-PC:~/Documents/SISOP/PRAK2/SISOP$
```



Cek repository online



2. THREAD

a. Deklarasi fungsi pthread

Jika ingin menggunakan thread dalam program, perlu include library “pthread.h”. Setelah itu deklarasi fungsi thread di dalam program yang syntax nya adalah:

namafungsi(*variabel pthread*, NULL, *nama fungsi yang ingin dijalankan*, NULL);

contoh:

```
pthread_t tid[2];           // inisialisasi array untuk menampung thread
pthread_create(&(tid[0]), NULL, &playAndCount, NULL);
pthread_join(variable pthread, NULL) → Perintah untuk menunggu thread yang lain selesai
terlebih dahulu sebelum program mati
```

contoh implementasi program:

```
panji@PANJI-PC: ~/Documents/SISOP/PRAK2/modulisop2
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

// install vlc terlebih dahulu
// sesuaikan path file lagu sesuai folder masing2
// compile

pthread_t tid[2];           // inisialisasi array untuk menampung thread (dalam kasus ini terdapat 2 thread)
int length=20;             // inisialisasi jumlah counter

void* playAndCount(void *arg) { // fungsi yang akan dijalankan sbg thread
    unsigned long i=0;
    pthread_t id=pthread_self();
    int iter;
    if(pthread_equal(id, tid[0])) {
        system("clear");
        // printf("\n First thread processing\n");
        for(iter=length; iter>0; iter--) {
            printf("%d", iter);
            fflush(stdout);
            sleep(1);
            system("clear");
        }
    } else if(pthread_equal(id, tid[1])) {
        printf("jalan!\n");
        system("cvlc --play-and-exit --quite coba.mp3");
    }
    return NULL;
}
```



```

int main (void)
{
    int i=0;
    int err;
    while(i<2) { // looping untuk membuat thread sebanyak 2x
        err = pthread_create(&tid[i], NULL, &playAndCount, NULL); // membuat thread
        if (err!=0)
            printf("\nCan't create thread: [%s]", strerror(err));
        else
            // printf("\n Thread created successfully\n");
            i++;
    }
    pthread_join(tid[0], NULL); // thread join untuk menunggu thread pertama selesai
    pthread_join(tid[1], NULL); // thread join untuk menunggu thread kedua selesai

    return 0;
}

```

52,1 Bot

Output:

```

panji@PANJI-PC: ~/Documents/SISOP/PRAK2/modulsisop2
20VLC media player 2.1.6 Rincewind (revision 2.1.6-0-gea01d28)
vlc: unknown option or missing mandatory argument '--quite'
Try 'vlc --help' for more information.

```

14

b. Cara compile program yang menggunakan thread

Dengan menambahkan parameter “-lpthread” di akhir perintah

`gcc -o fileexecutable filesourcecode -lpthread`

ctt: POSIX Thread : Harus return value (void*), pointer cuma 1 (*void)

3. STRING

Fungsi strstr: fungsi ini akan mencari lokasi substring dari suatu string. Cara penulisannya adalah strstr(string,index). Contoh:



```
stevenkur@Kurniawan: ~/Praktikum
GNU nano 2.2.6 File: strstr.c

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

int main()
{
    char input[50];

    printf("Masukkan String : ");
    scanf("%[^\n]",input);
    printf("%s\n",input);

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

    return 0;
}

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

stevenkur@Kurniawan: ~/Praktikum
stevenkur@Kurniawan:~/Praktikum$ ./strstr
Masukkan String : Hello World
Hello World
strstr = o World
stevenkur@Kurniawan:~/Praktikum$
```

Fungsi strtok: fungsi ini membagi string menjadi beberapa bagian yang dibatasi oleh karakter yang telah ditentukan. Cara penulisannya adalah strtok(string,index). Apabila kita ingin melanjutkan bagian yang kedua dan seterusnya, maka fungsi strtok ditulis dengan strtok(NULL,index) dan dilakukan looping sampai fungsi strtok!=0. Contoh:



```
stevenkur@Kurniawan: ~/Praktikum
GNU nano 2.2.6 File: strtok.c

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

int main()
{
    char input[50];

    printf("Masukkan String : ");
    scanf("%[^\n]",input);
    printf("%s\n",input);

    char *token=strtok(input,"o");
    while(token!=NULL)
    {
        printf("strtok = %s\n",token);
        token=strtok(NULL,"o");
    }

    return 0;
}

[ Read 20 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

stevenkur@Kurniawan: ~/Praktikum
stevenkur@Kurniawan:~/Praktikum$ ./strtok
Masukkan String : Hello World
Hello World
strtok = Hell
strtok = W
strtok = rld
stevenkur@Kurniawan:~/Praktikum$
```

4. FORK

Fork & Exec: *system call* fork yang berfungsi untuk membuat proses baru. Proses yang memanggil *system call* fork ini akan dibagi jadi dua, proses induk dan proses turunan yang identik. Analoginya seperti pembelahan sel, dimana satu sel membelah jadi dua sel yang identik. Proses induk dan turunan independen satu sama lain dan berjalan bersamaan. *Return code* dari *system call* ini adalah suatu integer. Untuk proses anak *return code*-nya adalah 0 sementara untuk proses induk *return code*-nya adalah nomor identifikasi proses (PID) dari turunannya. Ada juga *system call* exec yang berguna untuk membuat proses turunan yang terbentuk memiliki instruksi yang



berbeda dengan proses induknya. Dengan kata lain, proses induk dan proses turunan tidak lagi identik tapi masing-masing punya instruksi berbeda. Contoh:

```
stevenkur@Kurniawan: ~/Praktikum
GNU nano 2.2.6 File: latihan.c

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

void task1()
{
    int a,b;
    scanf("%d %d",&a,&b);
    printf("%d\n",a+b);
}

void task2()
{
    execl("/bin/ls","ls","/home",NULL);
}

void task3()
{
    int i;
    for(i=0;i<100;i++)
    {
        printf("%d\n",i+1);
    }
}

int main()
{
    pid_t pid;
    pid=fork();
    if(pid==0)
    {
        task1();
    }
    else if(pid>0)
    {
        wait();
        pid=fork();
        if(pid==0)
        {
            task2();
        }
        else if(pid>0)
        {
            wait();
            pid=fork();
            if(pid==0)
            {
                task3();
            }
            else wait();
        }
    }
    return 0;
}
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell



```
stevenkur@Kurniawan: ~/Praktikum
stevenkur@Kurniawan:~/Praktikum$ ./latihan
Masukkan 2 bilangan : 3 4
Hasil dari 3+4=7
stevenkur
Nomer 1
Nomer 2
Nomer 3
Nomer 4
Nomer 5
Nomer 6
Nomer 7
Nomer 8
Nomer 9
Nomer 10
stevenkur@Kurniawan:~/Praktikum$
```

5. SIGNALING

Signal adalah sebuah event yang dikirimkan ke suatu proses oleh proses itu sendiri atau proses lain. Kadang juga digunakan untuk mengirimkan notifikasi kepada suatu proses bahwa ada suatu event. Penggunaannya harus menggunakan library sebagai berikut.

```
#include <sys/types.h>
#include <signal.h>
```

Untuk menangkap signal dibutuhkan signal handler kita bisa membuat signal handler. Contoh penggunaan fungsinya:

```
void signalhandler()
```

Untuk memanggilnya bisa menggunakan system call signal (signal, namafungsi)

