IV. HASIL DAN PEMBAHASAN

Web server menggunakan *Library Swill* merupakan *embedded web server* yang dirancang oleh *programmer* sesuai dengan keinginan *user* maupun keinginan *programmer. Web server* ini dapat diakses oleh *client* menggunakan kabel cross, dan client dapat melakukan *request* pada *web server*. *Web server* sebelumnya sudah dibuat oleh **Noprianto**, yang referensinya terdapat pada majalah InfoLinux, 2008. Dan pada penelitian ini dilakukan penambahan *fungsionalitas* pada *web server.* Metode yang digunakan pada penelitian ini yaitu Metode *Waterfall.*

Metode *Waterfall*

Pembuatan dan pengembangan perangkat lunak dengan menerapkan metode *Waterfall*, metode ini terdiri dari 5 tahapan yaitu: *requirements, design, implementation/coding, verification, maintenance.*

4.1 *Requirements*

Tahap ini merupakan, pengumpulan kebutuhan-kebutuhan perangkat lunak *(software)* yang digunakan pada pembuatan dan pengembangan *software* ini, diantaranya bahasa pemograman, yaitu Bahasa C, *terminal, distro Ubuntu.9.4.*

Berdasarkan fungsi *web server* pada umumnya, yaitu dapat *merequest* file html, menjalankan CGI, dapat *mendownload* file, dapat mengunggah file, dapat

menjalankan file.php, dapat menyimpan database mysql, *web server* yang dibuat serta dikembangkan dengan *library swill* ini terdapat fungsi tambahan selain dari fungsi *web server* pada umumnya baik *GNU Linux* maupun pada *Windows*, yaitu dapat mengontrol program dari *web*, dapat menampilkan hasil program yang dijalankan pada terminal seperti ls -al, ataupun ifconfig.

Web server menggunakan *library swill* yang dibahas pada laporan skripsi terdapat fungsionalitas yaitu:

1. Menerima *request* file.html dan file lain
2. Bekerja dengan *Documen Root*
3. Handler berupa fungsi C
4. Menjalankan CGI
5. *Authentifikasi*
6. Membuat program yang dapat dikontrol dari *web*
7. Menjalankan Aplikasi tabel

4.2 *Design*

Tahap ini merupakan tahap pembuatan suatu gambaran atau *design* suatu sistem yang perlu dilakukan pengembangan, penambahan *fungsionalitas* suatu *web server*, yang kemudian diimplementasikan dalam bentuk *coding* pada tahap selanjutnya.

1. Menerima *request* file.html dan file lain

   Dapat menampilkan file.html atau *file* lain pada *web browser* sesuai dengan keinginan *user*, selain file html yang dapat *direquest* oleh *user* adalah file.jpg, file.png, file.gif, file.txt.

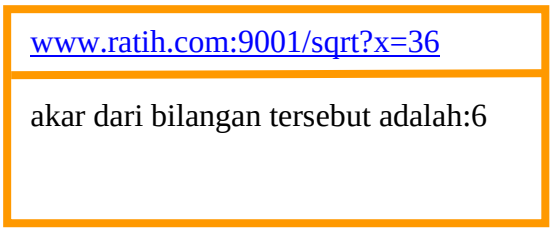
2. Bekerja dengan *Documen Root*

   Dapat menampilkan *file* atau perintah seperti ls yang terdapat pada *Document Root,* file tersebut diletakkan pada salah satu *Document Root* yang sebelumnya sudah dideklarasikan pada *source code web server*nya.

3. Handler berupa fungsi C

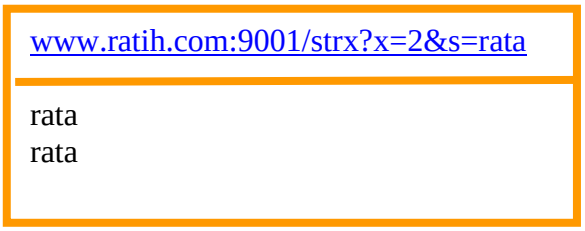
   Menjalankan fungsi C tertentu, misalkan pada *web server* ini dapat menghitung akar suatu bilangan, dan dapat menampilkan tulisan berulang yang hasilnya dapat ditampilkan pada *web browser.*

   Gambar 2 adalah desaign dari fungsionalitas menghitung akar dari suatu bilangan. Yang hasilnya dapat ditampilkan pada *web browser*.



Gambar 2. Desain menghitung akar (sqrt)

Gambar 3 adalah desain dari fungsionalitas menampilkan kalimat berulang, yang hasilnya ditampilkan pada *web browser*, seperti pada gambar 3 berikut:



Gambar 3. Desain melakukan perulangan.

4. Menjalankan CGI

   Dapat menampilkan perintah perintah yang biasanya dijalankan atau ditampilkan di *console*, dapat ditampilkan pada *web browser*. Pada *web browser* ini menjalankan perintah ls –al, ifconfig.

5. *Authentifikasi*

   Sebelum mengakses *web server*, *user* harus melakukan *authentifikasi* terlebih dahulu, yaitu dengan memasukkan *user* dan *password* yang telah disimpan pada *server*. Hal ini dilakukan untuk mengamankan *web server* , ditampilkan pada Gambar 4.

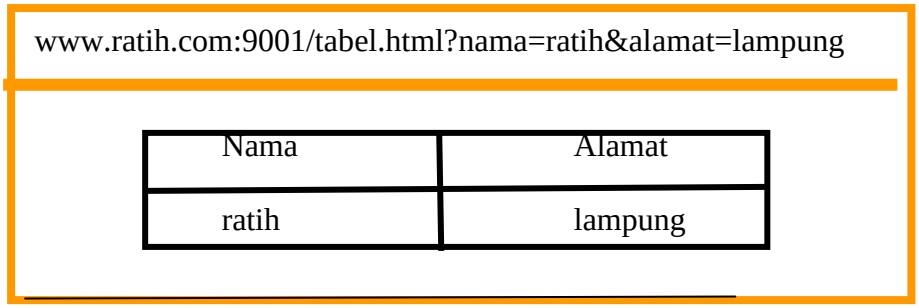
Gambar 4. Desain *Authentifikasi*

6. Membuat program yang dapat dikontrol dari *web*

   Pada fungsionalitas ini, *web server* dapat mengontrol jalannya suatu program yaitu mengendalikan program pengulangan bilangan ganjil menjadi bilangan genap dengan menuliskan m=0 pada *URL* dari *browser*.

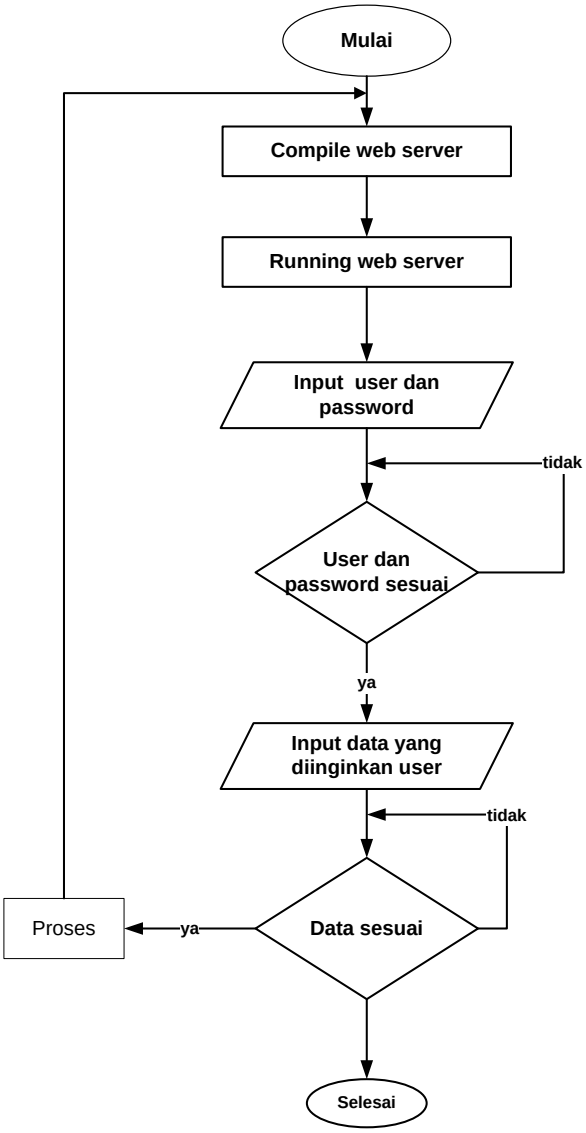
7. Menjalankan Aplikasi tabel

   Dapat menampilkan aplikasi tabel dengan input melalui *URL browser*, dan hasilnya dapat ditampilkan pada suatu *web browser*. Pada *web server* ini menginputkan suatu nama dan alamat pada tabel, desain ditampilkan pada Gambar 5 berikut:



Gambar 5. Desain aplikasi tabel

Gambar 6 merupakan Alur proses dari *web server*:



Gambar 6. Alur proses *web server*

4.3 *Coding / Implementation*

Tahap berikutnya adalah *implementation/coding.* Pada tahap ini, *design* tersebut diimplementasikan ke dalam bentuk *coding* agar dapat dimengerti oleh mesin komputer.

Pada penelitian ini dilakukan pembuatan serta pengembangan *embedded web server* pada sistem operasi GNU *Linux*, distribusi *Ubuntu* 9.4, dengan menggunakan *Library Swill* yang terdapat pada Bahasa C.

Pembuatan serta pengembangan *embedded web server* ini menggunakan fungsi-fungsi yang terdapat ada *Library Swill*, diantaranya:

a) swill_init ()
b) swill_serve()
c) swill_file()
d) swill_directory()
e) swil_poll
f) swill_handle()
g) swill_getargs()
h) swill_log()
i) swill_user()

Berikut ini adalah penjelasan serta penerapan langsung pada pemograman bahasa C, dan melakukan pembuatan serta pengembangan *embedded web server* menggunakan fungsi-fungsi *Library Swill* di atas.

4.3.1 swill_init()

Fungsi ini digunakan untuk menginisialisasi atau menjalankan *port* yang telah ditetapkan pada suatu *web server* yang dibuat serta dikembangkan.

Bentuk umum dari fungsi swill_init():

```
swill_init();
```

Salah satu dari swill_init() yang terdapat dalam pembuatan serta pengembangan *web server* ditampilkan pada source code di bawah ini:

```
#define PORT 9001
int main(void)
{

if (!swill_init (PORT))

{

fprintf (stderr, "Error.\n");
return 1;
}
```

Parameter dari fungsi swill_init () yaitu argumen yang terdapat diantara tanda (), Potongan program di atas menunjukkan bahwa *port* yang digunakan yaitu *port* 9001, dan *port* tersebut yang digunakan untuk menjalankan *web server*. Dengan adanya program tersebut, maka *web server* dapat dijalankan tetapi belum dapat melayani *request* dari *client.*

Berikut ini perintah/command untuk adalah menjalankan program dengan terminal, pada sistem operasi GNU Linux :

ratih@ratih-laptop:~$ cd swill/

ratih@ratih-laptop:~/swill$ gcc httpd1.c /usr/local/lib/libswill.a -o port

ratih@ratih-laptop:~/swill$ ./port

Server terhubung pada port:9001.

Apabila dijalankan pada *web browser*, maka terlihat tampilan pada Gambar 7 sebagai berikut:

Gambar 7. *Web server* dengan fungsi swill_init()

4.3.2 swill_serve()

Fungsi swill_serve() ini digunakan untuk menerima *request* dari *client.* Berikut ini adalah penggunaan fungsi swill_serve() pada program C, yang digunakan untuk membuat *web server* adalah

```
fprintf (stdout, "Server terhubung pada port %d.\n",PORT);
while (1);
{
swill_serve()
}
```

Dengan menjalankan fungsi ini, maka *web server* sudah dapat melayani *clien*t, sesuai dengan permintaan *client.* Apabila permintaan dari *client* tersebut sesuai dengan *fungsionalitas web server*, maka *web server* dapat melayaninya. Kemudian *swill_serve* ini dapat berjalan, sampai ada *client* yang memberikan *request*

4.3.3 swill_file()

Fungsi dari swill_file() adalah untuk menambahkan sebuah file ke *server*, untuk tipe *file* yang dapat digunakan pada fungsi swill_file() ini sudah dibahas pada bab

sebelumnya, yaitu bab 2. Jadi selama file tersebut memenuhi syarat, maka dapat ditambahkan jumlah *file* yang disimpan pada suatu *web server*. Dengan menambahkan fungsi swill_file() dalam program sesuai dengan jumlah *file* yang akan disimpan pada *web server.*

Berikut ini adalah dalam menggunakan swill_file() pada suatu program Bahasa C yang digunakan untuk membuat ataupun mengembangkan suatu *web server*:

```
swill_file (“ratih.html”,0);
swill_file (“ls”,0);
swill_file (“tania.gif”,”th_493.gif”);
swill_file (“nia.html”,0);
```

Parameter dari fungsi swill_init () yaitu argumen yang terdapat di antara tanda (), Pada program di atas, yaitu menyimpan *file* pada *web server* dengan menggunakan fumgsi swill_file(), pada swill_file ini, *file* yang dapat disimpan yaitu:

- file.txt
- file.html
- file.jpg
- file.png
- file.gif

dan bentuk umum dari fungsi swill_file() ini adalah :

```
swill_file(“namafile“);
```

Program di atas swill_file(“ratih.html”,0); menunjukkan bahwa *file* yang disimpan pada *web server* adalah ratih.html dan koma 0 menunjukkan bahwa *file* tersebut

teletak pada direktori yang sama dengan *file source code* dari *web server* tersebut. Pada `swill_file ("tania.gif","th_493.gif");` menjelaskan bahwa *file* yang disimpan yaitu th_493.gif, dengan nama tania.gif, yang ditulis pada *URL* pada saat meminta *file* tersebut dan yang terdapat pada menu *web server*.

Potongan program di atas menunjukkan bahwa *file* - *file* yang dapat di*request* oleh *client* adalah: ratih.html, nia.html, ls, dan tania.gif dimana *file* - *file* tersebut harus diletakkan satu direktori dengan *file source code* dari *web server* tersebut.

Berikut ini cara menjalankan program pada terminal.

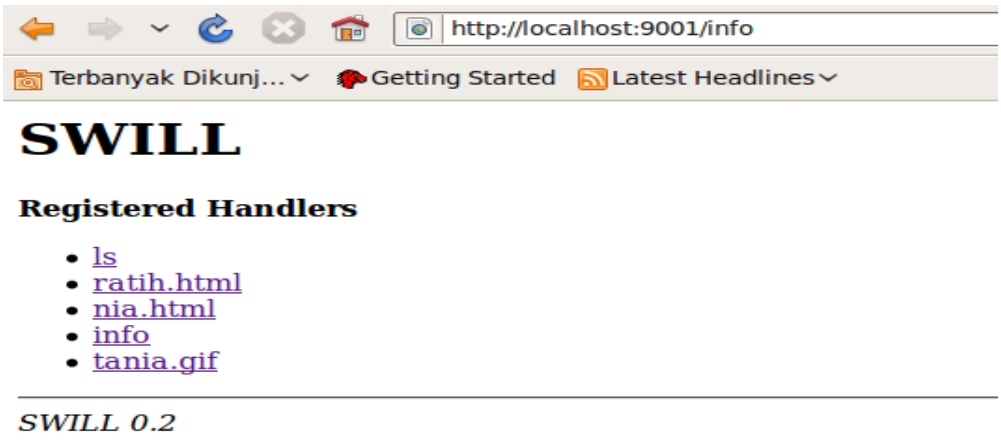
ratih@ratih-laptop:~$ cd swill/

ratih@ratih-laptop:~/swill$ gcc httpdfile.c /usr/local/lib/libswill.a -o file

ratih@ratih-laptop:~/swill$ ./file

Server terhubung pada port 9001.

Berikut ini Gambar 8 adalah tampilan *web server* apabila dijalankan pada *web browser:*



Gambar 8. *Web server* dengan fungsi swill_file()

Apabila *client* meminta *file* ls, maka *web server* mejalankan perintah *download* file ls, hal ini membuktikan bahwa *web server* yang dibuat menggunakan *Library swill* dapat men*download* suatu aplikasi atau suatu informasi yang terdapat pada *console* atau *terminal*. Tampilannya pada Gambar. 9 sebagai berikut:

Gambar 9. Proses *mendownload file* ls

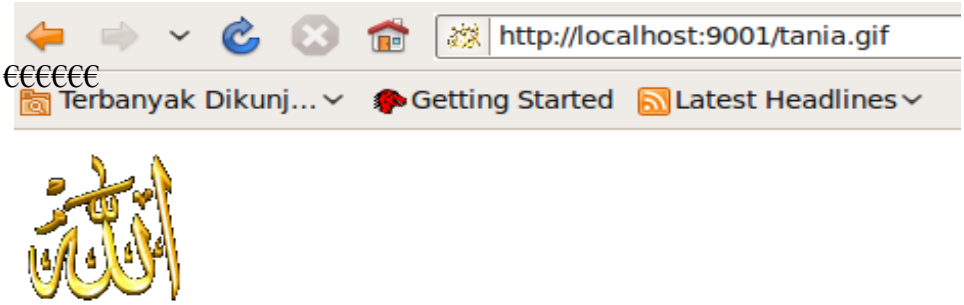
Apabila *client* me*request file* ratih.html, maka dalam *web browser* menampilkan Gambar 10 sebagai berikut:

Gambar 10. Proses *merequest* ratih.html

Apabila *client* me*request file* nia.html, maka keluar tampilan Gambar 11 sebagai berikut:

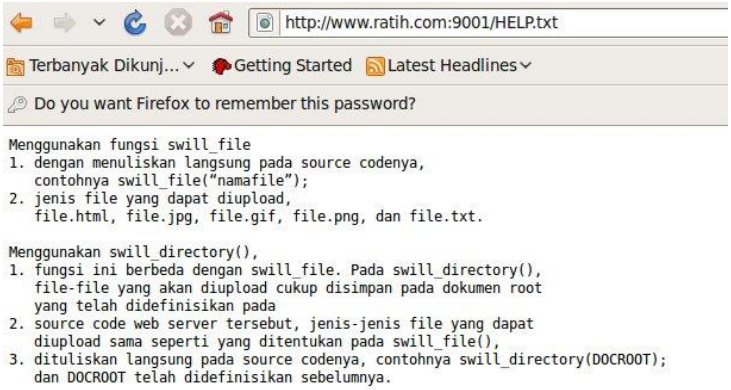
Gambar 11. *Web server* menjalankan *file* nia.html

Apabila *client merequest file* tania.gif, maka *web browser* akan menampilkan Gambar 12 sebagai berikut:



Gambar 12. *Web server* menjalankan *file* tania.gif

Apabila *client* me*request file* ratih.html, maka dalam *web browser* menampilkan Gambar 13 sebagai berikut:



Gambar 13. *Web server* menjalankan *file* HELP.txt

4.3.4 swill_directory()

Fungsi dari swill_directory() ini adalah menyimpan *dokumen root*, fungsi ini hampir sama dengan fungsi dari swill_file(). Tetapi perbedaannya adalah apabila menggunakan fungsi swill_file() maka *file-file* tersebut harus diterjemahkan dengan menggunakan fungsi tersebut secara satu per satu dengan kata lain jumlah swill_file() yang terdapat dalam program harus sesuai dengan jumlah *file* yang disimpan pada *web server*.

Dengan menggunakan swill_directory() jumlah *file* yang disimpan pada suatu *web server*, dapat diterjemahkan pada suatu program dengan menggunakan fungsi swill_directoty(). Direktori yang dibuat diletakkan pada salah satu direktori yang terdapat pada *root*.

Bentuk umum dari swill_directory :

```
swill_directory(nama direktori);
```

Parameter dari fungsi swill_init () yaitu argumen yang terdapat di antara tanda (), Pada penelitian ini adalah, direktori dibuat dengan nama *web* dan CGI, diletakkan pada /opt.

Berikut ini adalah program dengan menggunakan *swill_directory()*

```
#define DOCROOT "/opt/web/"
#define CGIROOT "/opt/cgi/"
#define RATIH "/home/ratih/"

void do_list ()
{

char cmd[255];
sprintf (cmd,"ls-al %s", RATIH);
system (cmd);
}

void do_listhtml ()
{

char cmd [255];
sprintf (cmd,"/opt/cgi/list.cgi%s",DOCROOT);
system (cmd);
}


swill_directory (DOCROOT);
```

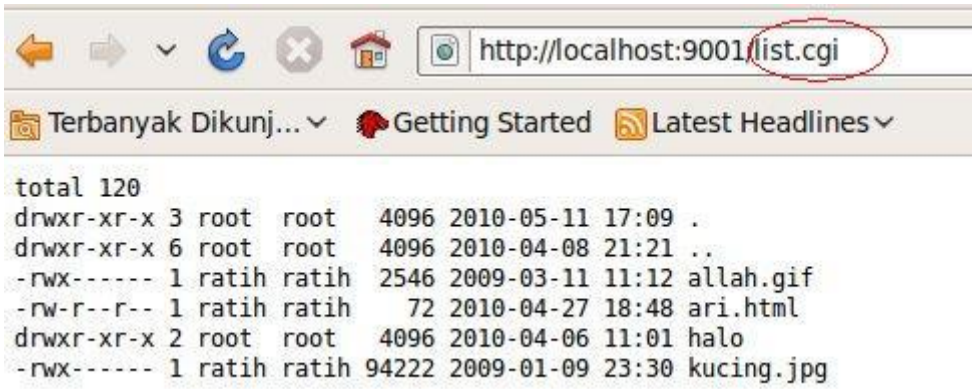
Pada potongan program di atas, akan menjalankan fungsi yang dapat menampilkan ls -al dari */opt/web/*, *dan /home/ratih/*

Gambar14 merupakan cara menjalankan program tersebut menggunakan *terminal* atau *console*, seperti pada gambar 14 berikut:

Gambar 14. Menjalankan swill_directory() pada terminal

Apabila *client merequest* ls dari /opt/web, tampil seperti Gambar 15 berikut:

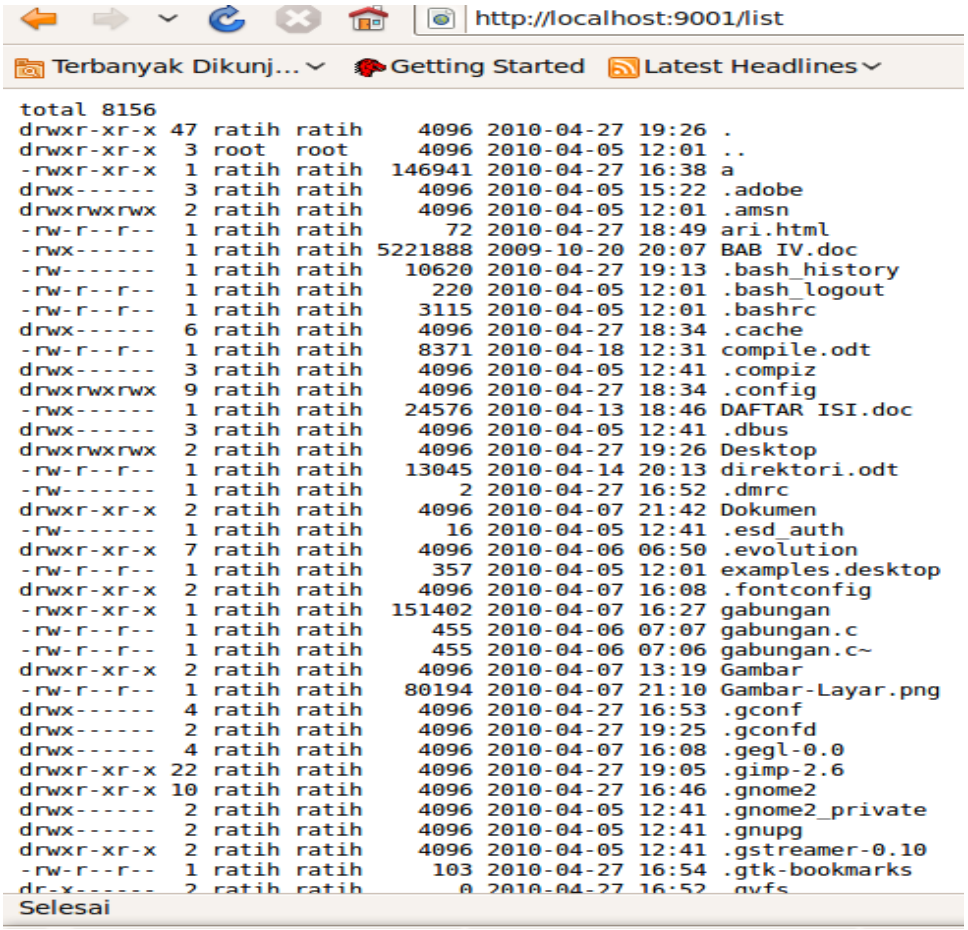
:



Gambar 15. *Web Server* menjalankan *list.cgi* dengan swill_directory()

Pada Gambar.15, menjelaskan bahwa *web server* dapat menjalankan perintah ls –al yang terdapat pada /opt/web/ (DOCROOT), yang mana perintah ls –al tersebut meripakan perintah dari sistem operasi *GNU Linux* yang dapat menampilkan seluruh *file* dan direktori (*hidden* dan aktif) yang terdapat pada direktori /opt/web/.

Apabila *client* me*request* ls dari /home/ratih/, maka ditampilkan Gambar 16 sebagai berikut:



Gambar 16. *Web server* menjalankan *list* dengan swill_directory()

Pada Gambar 16, menjelaskan bahwa *web server* dapat menjalankan perintah ls –al yang terdapat pada /home/ratih (DOCROOT), dimana perintah ls –al tersebut merupakan perintah dari sistem operasi *GNU Linux* yang dapat menampilkan seluruh *file* dan direktori (*hidden* dan aktif) yang terdapat pada direktori /home/ratih

Hal ini juga membuktikan bahwa *web server* yang dibuat menggunakan *swill* dapat menjalankan *comand-comand* yang terdapat pada *terminal* atau *console* dan dapat ditampilkan pada *web browser*, seperti contoh yang diperlihatkan sebelumnya.

Pada saat *client merequest* ari.html yang terdapat pada *direktori root* /opt/web/, maka *web browser* menampilkan Gambar 17 sebagai berikut:

:

Gambar 17. Menampilkan *file* ari.html

Pada saat *client merequest* gambar allah.gif yang terdapat pada *direktori root* /opt/web/, maka web browser menampilkan Gambar 18 sebagai berikut:

Gambar 18. Menampilkan gambar allah.gif

Pada saat *client* me*request* gambar kucing.jpg yang terdapat pada direktori root /opt/web/, maka web browser menampilkan Gambar 19 sebagai berikut:

Gambar 19. Menampilkan gambar kucing.jpg

4.3.5 swill_poll()

Fungsi ini digunakan untuk hampir sama dengan swill_serve(), yaitu menerima *request* dari *client*, perbedaannya dengan fungsi swill_serve() adalah apabila fungsi ini menerima *request* dari *client* tetapi menunggu ada *request* dari *client* terlebih dahulu baru fungsi ini dapat dijalankan, sedangkan fungsi swill_poll() dapat melayani *request* dari *client* tanpa harus ada permintaan dari *client*, jadi fungsi ini sudah berjalan.

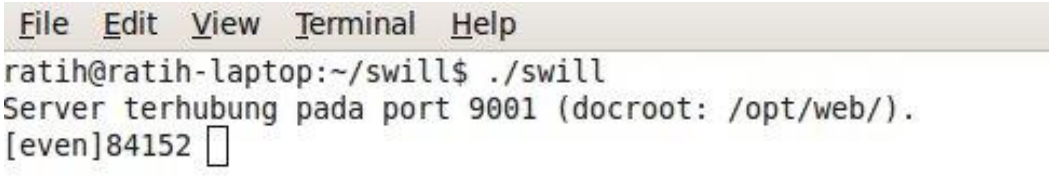
Bentuk umum dari swill_poll:

```
swill_poll();
```

Berikut ini program yang menggunakan fungsi dari swill_poll()

```
}
swill_poll();
}
```

Cara menjalankan program tersebut dengan *terminal* atau *console* adalah pada Gambar 20 sebagai berikut:



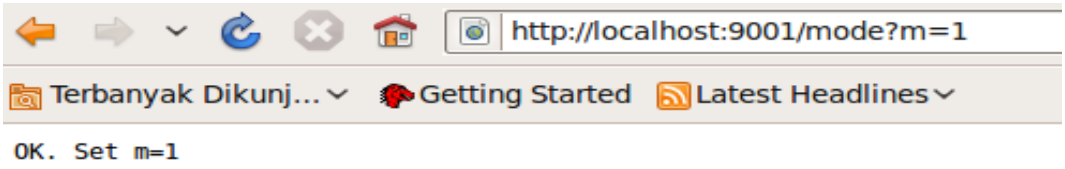
Gambar 20. *Web server* dengan fungsi swill_poll()

Dalam tampilan teks tersebut terlihat bahwa program yang belum *direquest* oleh *client*, tetapi sudah berjalan dan fungsi swill_poll() ini, juga menunjukkan bahwa *web server* yang dibuat dengan *Library swill* , dapat mengontrol program dari *web*, dalam *web server* ini contohnya yaitu dapat menampilkan pengulangan bilangan yang sebelumnya program menjalankan pengulangan bilangan ganjil menjadi pengulangan bilangan genap.

Untuk mengontrol program dari *web* dan dalam hal ini mengubah pengulangan bilangan ganjil menjadi pengulangan bilangan genap, yaitu dengan mengetikkan pada *URL web browser* m=0 untuk bilangan ganjil, dan m=1 untuk merubah kembali menjadi pengulangan bilangan ganjil.

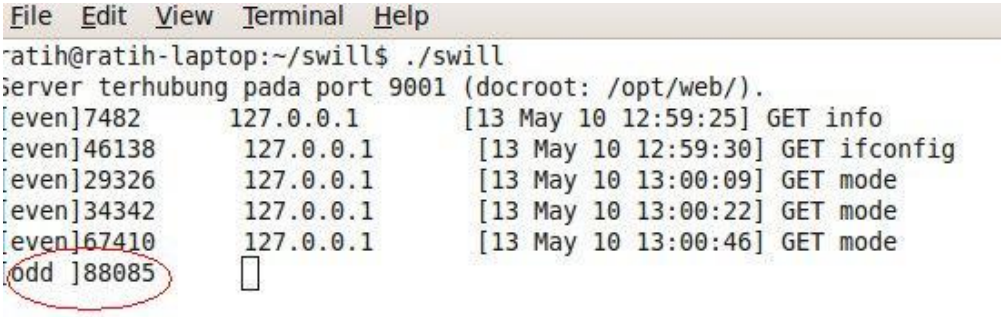
Berikut ini adalah Gambar 21 sampai Gambar 23 merupakan tamplian sebelum diketikkan m=0 (m=1)

Tampilan pada *web browser*:



Gambar 21. swill_poll() pada *Web browser*

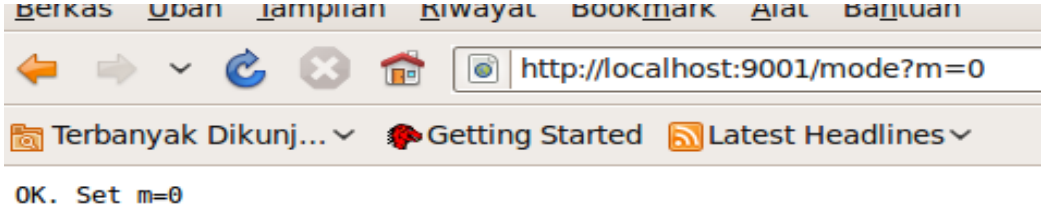
Pada Terminal:



Gambar 22. swill_poll() pada terminal

Setelah di *URL* diketikkan m=0, pengulangan menjadi bilangan genap.

Pada *Web Browser:*



Gambar 23. Mengendalikan *program* dengan *swill* pada *web browser*

Pada Gambar 24 menunjukkan bahwa pengulangan bilangan ganjil berubah menjadi bilangan genap dengan mengetikkan ?m=0 pada URL.seperti pada gambar 24 berikut:

Pada Terminal:

Gambar 24. Mengendalikan program dengan *swill* pada terminal

4.3.6 swill_handle()

Fungsi ini berfungsi untuk menjalankan fungsi C yang dapat menjalankan pekerjaan tertentu. Dalam pembuatan *web server* salah satu fungsi yang dijalankan yaitu menjalankan ifconfig, yang seharusnya dijalankan pada *terminal*

Berikut ini merupakan potongan program yang terdapat pada pembuatan serta pengembangan *web server* ini:

Bentuk umum dari swill_handle:

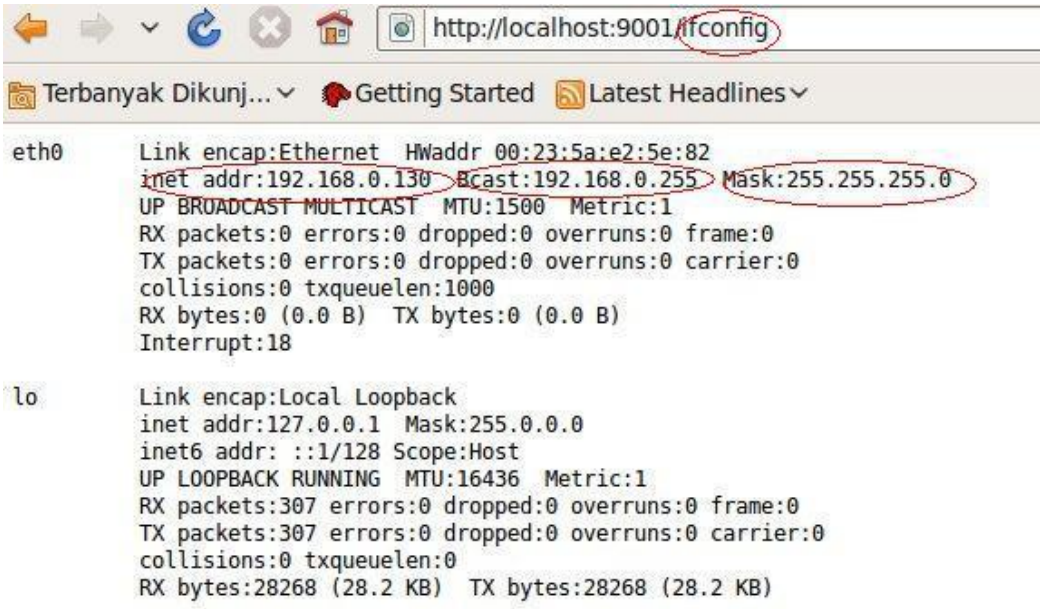
```
swill_handle("namaURL","namafungsi");
```

Potongan Program dari swill_handle():

```
void do_ifconfig ()
}
system("ifconfig");
}


swill_handle("stdout:ifconfig",do_ifconfig,0);
```

Pada potongan program di atas swill_handle ("stdout:ifconfig",do_ifconfig,0); menunjukkan bahwa stdout berfungsi untuk menampilkan *command* yang terdapat pada terminal atau *console*, ifconfig merupakan nama yang akan ditulis pada *URL web browser* sedangkan do_ifconfig yaitu nama fungsi. Koma 0, di sini berarti bahwa tidak ada fungsi lain yang dijalankan, selain fungsi ifconfig.

Apabila dijalankan pada *web browser*, maka seperti Gambar 25 berikut:



Gambar 25. Menampilkan ifconfig menggunakan fungsi swill_handle()

Pada Gambar 25 menunjukkan bahwa fungsi ini untuk menampilkan perintah ifconfig pada web browser, dimana perintah ifconfig untuk melihat *localhost, IP Address*, *Netmask*, dan *Broadcast* pada sebuah PC (terdapat pada gambar diatas yang ditandai dengan lingkaran merah), yang biasanya perintah tersebut dijalankan di terminal

.

4.3.7 swill_getargs()

Fungsi ini digunakan untuk menangkap atau mentejermahkan *file.html* dan *URL variabel* pada suatu *web browser.* Parameter dari fungsi swill_init () yaitu argumen yang terdapat diantara tanda ().

Bentuk umum dari fungsi swill_getargs():

```
swill_getargs("tipevariabel(namavariabel");
```

Pada pembuatan *web server*, swill_getargs() ini digunakan untuk menjalankan fungsi *sqrt*, *strx*, dan *mode*, dan penambahan fungsionalitas tabel yang belum ada sebelumnya.

Berikut ini adalah contoh potongan program Bahasa C yang menggunakan fungsi swill_getargs(), yang merupakan *source code* dari pengembangan *web server* tersebut yaitu penambahan aplikasi tabel

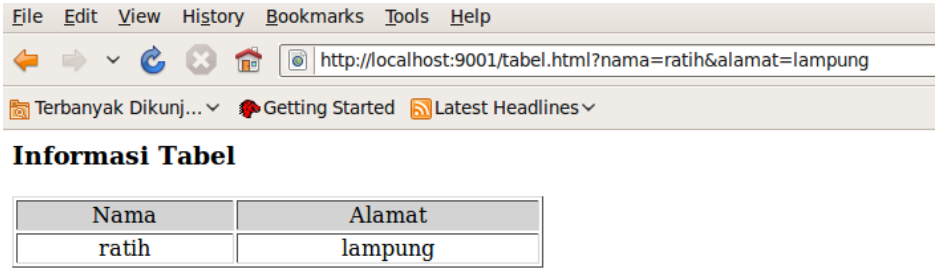
```
void do_sayhello ( FILE *f )
{
char *nama;
char *alamat;

if (!swill_getargs (“s(nama)|s(alamat)”,&nama,&alamat))
{
{

fprintf (f, “Error, input nama dan alamat pada URL,\n”);
}
else

{
fprintf (f,”<h3><b>Informasi Tabel</b></h3><TABLE BORDER=1
WIDTH=40px
>\n<TR>\n<TD align=center bgcolor=lightgrey>Nama</TD>\n<TD
align=center bgcolor=lightgrey>Alamat</ TD>\n</TR>\n<TR>\n<TD
align=center>%s</TD>\n<TD align=center>
%s</TD>\n</TR>\n</TABLE>\n”,nama, alamat);

}
}
```

ratih@ratih-laptop:~/swill$ gcc httpd7.c /usr/local/lib/libswill.a -o tabel -lm

ratih@ratih-laptop:~/swill$ ./tabel

Server terhubung pada port: 9001.

Pada contoh program pembuatan tabel di atas menggunakan *variabel s (string)* sebagai nama dan alamat, maka di *web browser* harus diketikkan *URL variable*



Gambar 26. Menampilkan Aplikasi Tabel Pada *Web Browser* Dengan Fungsi swill_getargs()

4.3.8 swill_log()

Fungsi swill_log() dapat digunakan untuk menampilkan proses-proses penting yang terdapat *web server*.

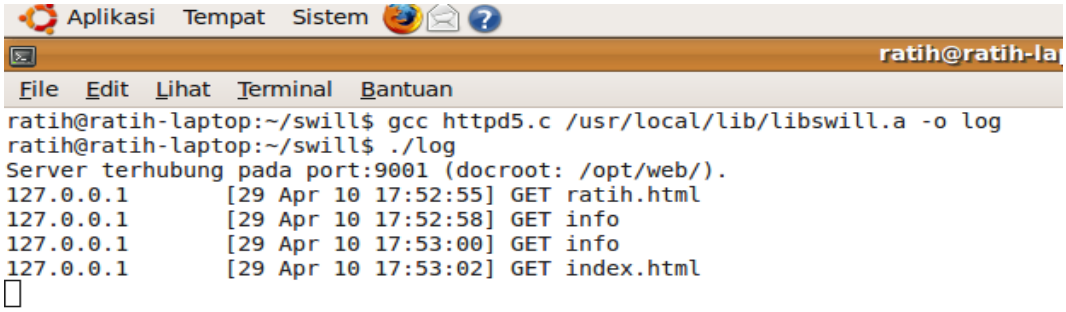
Bentuk umum dari swill_log:

```
swill_log ();
```

Potongan program menggunakan fungsi swill_log() yang terdapat pada *web server* ini adalah:

```
{
swill_log (stdout);
}
```

Proses dari swill_log() ini dapat dilihat tampilannya pada terminal. Seperti contoh Gambar 27 di bawah ini:



Gambar 27. Menampilkan proses swill_log() pada terminal

Informasi yang dapat diterima dari proses *web server* diantaranya: *IP Address* yang mengakses *web server*, waktu saat melakukan *request* dari *web server*, serta nama *file* yang *direquest.*

4.3.9 swill_user()

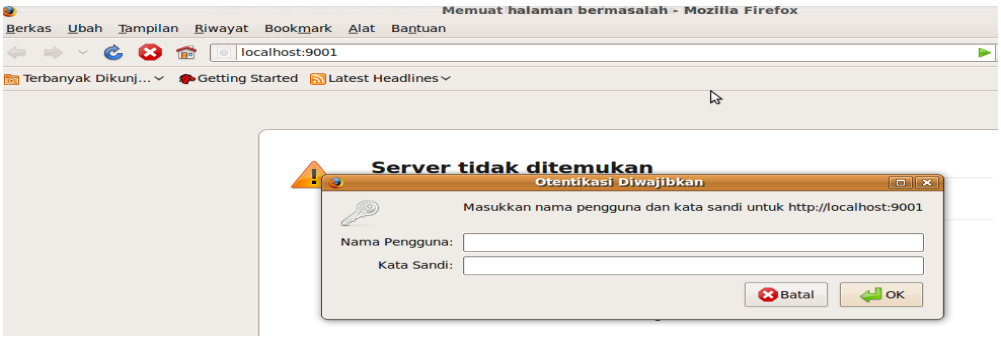
Fungsi dari *swill_user()* ini adalah untuk *authentifikasi*, apabila *user* ingin mengakses *web server* ini, maka diwajibkan untuk mengisi *username* dan *password* terlebih dahulu. Parameter dari fungsi swill_init () yaitu argumen yang terdapat diantara tanda (). Bentuk umum dari swill_user():

```
swill_user (“user”,”password”);
```

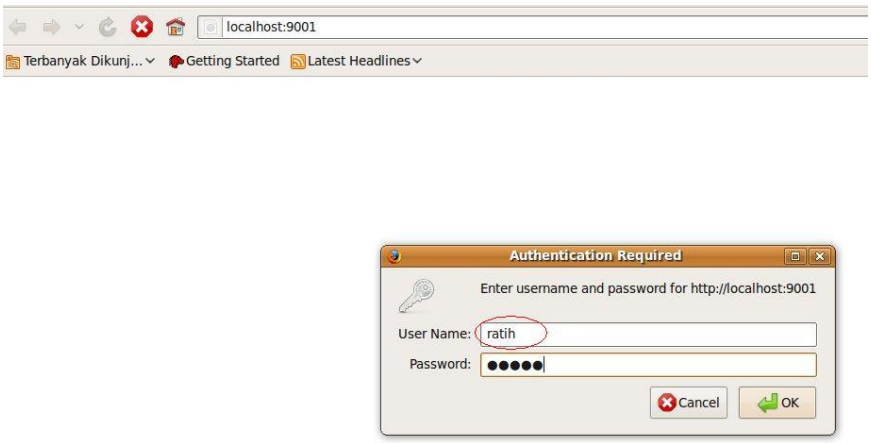
Berikut ini potongan program yang menggunakan fungsi swill_user()

```
swill_user (“ratih”,”ratih”);
swill_user (“user”,”user”);
```

Apabila dijalankan pada *web browser*, maka menampilkan seperti pada Gambar 28 berikut:



Gambar 28. *Web server* dengan fungsi swill_user()



Gambar 29. *Web server* dengan *username* dan *passwrod*

Tampilan di atas adalah pada saat proses memasukkan *username* serta *password* sebelum menggunakan *web server* yang dibuat menggunakan *library swill*.

Pada penelitian ini semua fungsi-fungsi yang telah dijelaskan di atas, dijadikan satu dalam satu program, dengan tujuan agar dapat mempermudah dalam menggunakan *web server* ini.

Cara menjalankan *web server* ini, yang telah dijadikan satu yaitu pada Gambar 30 sebagai berikut

Gambar 30. Menjalankan *web server*

Gambar 31 merupakan tampilan awal dari *web server* setelah dijalankan pada *web browser* :

Gambar 31. Tampilan awal dari *web server*

Setelah diklik here pada tampilan sebelumnya, maka pada *web browser* menampilkan menu yang tersedia dari *web server* yang telah digabungkan, tampilannya pada Gambar 32 sebagai berikut:

Gambar 32. Tampilan *web server* dengan *library swill*

4.4 *Testing / Verification*

Pada tahap ini dilakukan *testing* setelah sistem yang dikembangkan telah selesai dan sudah dapat digunakan sesuai dengan keinginan *user*. Sistem tersebut harus dilakukan *testing/verification*, untuk memastikan bahwa *web server* yang telah dibuat atau dikembangkan dapat berfungsi sebagaimana mestinya, dan sesuai dengan keinginan *user*.

Testing dapat dilakukan dengan cara mengakses *web server* pada *web browser* dengan mengetikkan alamat www.localhost:9001 atau www.ratih.com:9001 atau menghubungkan *web server* dengan *client* menggunakan kabel *cross*, kemudian pada *client disetting* terlebih dahulu *IP Addressnya* yang sesuai dengan *serv*er, kemudian mengatur file /etc/hosts, untuk memberikan *hostname, client* dan *server* terhubung , dan *client* dapat melakukan *request.*

Testing dilakukan dengan cara mengakses *web server* atau *merequest* salah satu file yang telah ada pada *server,* menampilkan aplikasi yang dapat dijalankan oleh terminal, mengontrol program dari *web* serta menjalankan aplikasi tabel merupakan aplikasi pengembangan yang ada pada *web server* ini. Apabila hal-hal di atas telah dilakukan dan berhasil, maka *web server* ini dibuat serta dikembangkan dengan baik.

Tabel 1 merupakan *Testing* yang dilakukan pada *Web Server*

Tabel 1. Testing web server

No	Fitur	Tanggal	Penguji	Hasil	Keterangan
1	mode (Mengontol Program dari *web*)	08 juni 2010	*Programmer*	Sesuai	Mengubah pengulangan bilangan ganjil menjadi pengulangan bilangan genap (m=0), dan sebaliknya
2	Meminta file.html yang ada pada *web server* dengan swill_file ()	08 juni 2010	*User*	Sesuai	Apabila meminta *file* ratih.html, akan menampilkan tulisan assalamualaikum teks berjalan.

3	Meminta file.gif yang ada pada *web server* dengan swill_file()	08 juni 2010	*User*	Sesuai	Menampilkan gambar bergerak(gif)
4	Menjalankan CGI (menjalankan fungsi ifconfig dengan menggunakan fungsi C)	08 juni 2010	*Programmer*	Sesuai	Menampilkan proses ifconfig pada *web browser*.
5	Menjalankan fungsi C (strx atau pengulangan)	08 juni 2010	*User*	Sesuai	Menampilkan tulisan alamat yang melakukan pengulangan sebanyak 5 kali
6	Menjalankan Fungsi C (sqrt atau akar bilangan)	08 juni 2010	*User*	Sesuai	Menampilkan akar dari hasil bilangan yang diinputkan.
7	Meminta file.html dengan aplikasi tabel	08 juni 2010	*Programmer*	Sesuai	Menampilkan aplikasi tabel.
7	Bekerja pada *dokumen root*	08 juni 2010	*User*	Sesuai	Menampilkan *file* ari.html(salah satu *file* yang terdapat pada *dokumen root*) dengan tulisan Ratih Ari Tania..
8	*Authentifikasi*	08 juni 2010	*User*	Sesuai	Dengan mengisi *user* dan *password* sebelum mengakses *web server*.

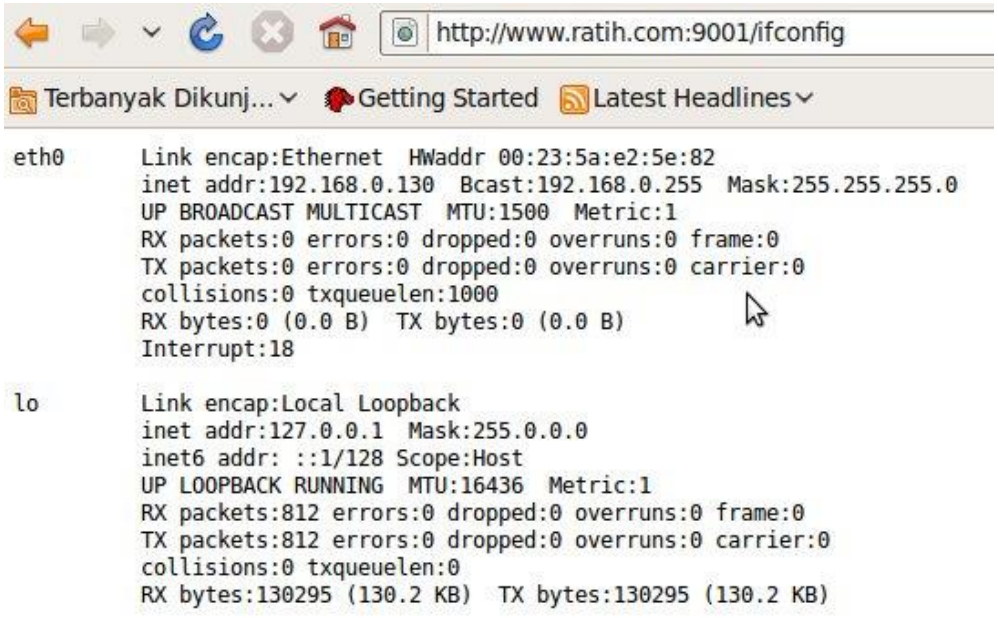
Dari tabel di atas dapat diuraikan sebagai berikut:

Setelah dilakukan *testing* pada *web server* ini ternyata bahwa *web server* berfungsi sesuai dengan apa yang direncanakan sebelumnya, hal ini terlihat dengan hasil *testing* dari *web server* tersebut.

Pada *testing* yang telah dilakukan oleh *user* maupun *programmer* hasilnya sesuai dengan apa yang direncanakan sebelumnya, hasilnya tidak mengalami perubahan pada saat *user* yang menjalankan *web server* yang sebelumnya telah dijalankan oleh *programmer* itu sendiri.

Web server ini juga tidak mempermasalahkan kapan *web server* tersebut diakses atau digunakan, artinya *web server* ini dapat digunakan kapanpun tanpa melihat waktu (tanggal atau jam) pada saat mengaksesnya.

Keterangan ganbar pada tabel di atas pada no 4 dan no 7 terdapat pada Gambar 33 dan Gambar 34 berikut:

4. 

Gambar 33. Menjalankan fungsi CGI dengan ifconfig

7. 

Gambar 34. Tampilan aplikasi tabel

4.5 Maintenance

Setelah dilakukan *testing/verification* pada *web server* ini, maka tahap selanjutnya adalah *maintenance*, hal ini perlu dilakukan karena apabila ada *error-error* kecil yang belum ditemukan sebelumnya, dan apabila pada sistem ini dilakukan pengembangan fungsionaltias atau ditambahkan fitur-fitur yang belum ada sebelumnya.

4.6 Pembahasan

Pada pembuatan serta pengembangan *web server* menggunakan *library swill* dapat dibuat atau dikembangkan sesuai dengan keinginan *user*, dan *web server* ini dapat digunakan seperti halnya *web server* lainnya, hanya saja *web server* ini bersifat lebih ringan.

Dalam penelitian ini dilakukan pengembangan *fungsionalitas* atau fitur yang belum terdapat pada *web server* sebelumnya, yaitu aplikasi tabel, Hal ini membuktikan bahwa *web server* ini dapat ditambahkan aplikasi tabel yang nantinya dapat digunakan atau dikembangkan lagi sesuai dengan keinginan *user*.
Embedded web server menggunakan *Library swill* dapat menampilkan aplikasi dari *terminal* atau *console* ke *web browser* serta dapat mengontrol program pada terminal melalui *web browser*.

Kekurangan dari *web server* yang dibuat dengan *library swill*, yaitu tidak dapat melakukan *upload file* dengan menggunakan PHP. Pada *web server* ini untuk *mengupload file* yaitu langsung ditambahkan pada *source code web server* tersebut dengan menggunakan swill_file(), dan swill_directory().
Berikut ini penjelasan cara *mengupload file* pada *web server* ini
Pada *web server* yang dibuat dengan *library swill*, ada dua cara yang dapat digunakan untuk *mengupload file*, yaitu

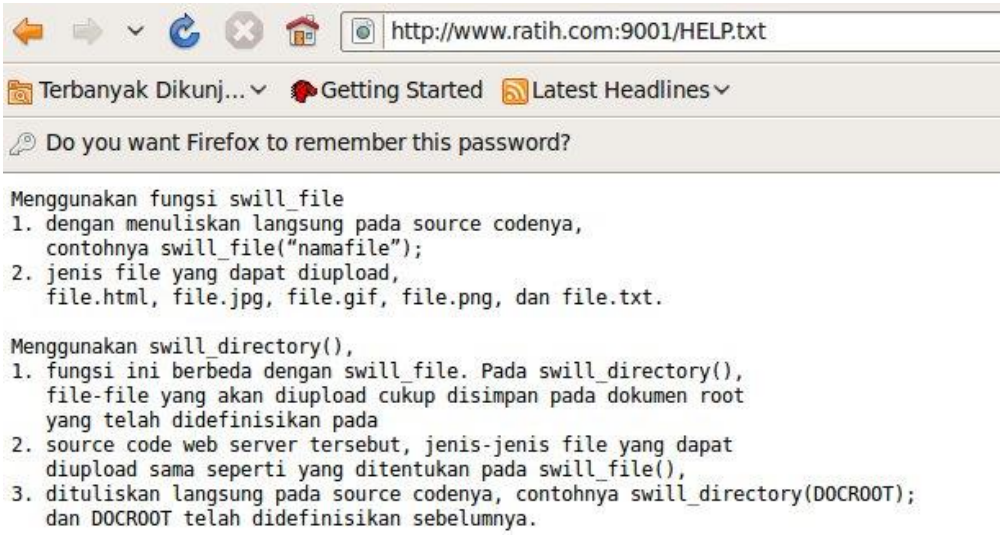
- Menggunakan fungsi swill_file
    1. Dengan menuliskan langsung pada *source code*nya, contohnya swill_file(“namafile”);

2. Jenis *file* yang dapat *diupload*, file.html, file.jpg, file.gif, file.png, dan file.txt.

- Menggunakan swill_directory(),
  1. Fungsi ini berbeda dengan swill_file. Pada swill_directory(), file-file yang akan diupload cukup disimpan pada *dokumen root* yang telah didefinisikan pada *source code web server* tersebut, jenis-jenis file yang dapat *diupload* sama seperti yang ditentukan pada swill_file(),
  2. Dituliskan langsung pada *source code*-nya, contohnya swill_directory(DOCROOT); dan DOCROOT telah didefinisikan sebelumnya.

Dengan kekurangan yang telah dijelaskan di atas, maka *web server* ini menyediakan fitur HELP yang akan membantu *user* dalam menggunakan *web server* ini.

Gambar 35 merupakan tampilan pada *web browser*, apabila menjalankan fitur HELP pada *web server*, seperti pada gambar 35 berikut:



Gambar 35. Menjalankan fitur HELP