BAB III

PERANCANGAN DAN PEMBUATAN SISTEM

3.1. Gambaran umum sistem.

Program aplikasi serbaguna berbentuk web untuk administrasi server dan jaringan yang menggunakan sistem operasi unix/linux adalah sebuah program aplikasi yang berfungsi untuk membantu para sistem administrator sistem dan jaringan dalam melaksanakan tugas mereka melakukan proses administrasi sistem dan jaringan. Program ini dapat dijalankan dengan menggunakan sebuah browser. Client dan server berkomunikasi dengan menggunakan sebuah 'bahasa' (protocol) yang disebut HTTP.

Web browser adalah contoh dari sebuah *web client*, sedangkan sebuah remote machine yang di dalamnya berisi dokumen yang akan diminta dinamakan *web server*.

Server bukan sebuah hardware melainkan sebuah program yang berjalan dalam sebuah komputer, web server mendengarkan (listen) pada sebuah port dalam suatu jaringan (network), dan menunggu sampai clients meminta melalukan transaksi melalui HTTP protocol. Setelah merespons request dari client, browser akan menampilkan data-data yang relevan yang telah dikirim oleh server.

Pada saat server melalukan proses Initialisasi setiap kali server melakukan proses *booting* program aplikasi akan langsung dijalankan.

```
# Ask whether to run at boot time {setup.sh}
```

```
        initsupp=`grep "^os_support=" $wadir/init/module.info | sed -e
's/os_support=//g' | grep $os_type`
        atboot=0
        if [ "$initsupp" != "" ]; then
                printf "Start Linux-admin at boot time (y/n): "
                read atbootyn
                if [ "$atbootyn" = "y" -o "$atbootyn" = "Y" ]; then
                        atboot=1
                fi
        else
        echo " Linux-admin does not support being started at boot time on your
system."
        fi
        makeboot=$atboot
```

3.2. Perancangan Data.

Untuk melakukan initialisasi terhadap socket, perlu dibuat sebuah input data sebagai masukan dari user.

Data	keterangan
Config_dir	Untuk meletakkan configurasi directory
Perldef	Memberikan keterangan tentang letak directory perl
Real_os_type	Nama sistem operasi
Real_os_version	Versi sistem operasi
Port	Nomer port yang digunakan
Login	Input data login
Passw	Input data password
Host	Nama host

Tabel 3.1. Tabel data Initialisasi

Untuk melakukan penyuntingan terhadap user administrator maka dibutuhkan data-data yang berisi user administrator, password dan hak akses. Data tersebut merupakan konfigurasi dari pengguna program admin yang pertama kali.

Perlu untuk dimasukkan pada saat initialisasi karena untuk menggunakan program ini paling sedikit harus ada satu user.

Data	Keterangan
Mods	Daftar modul/Task
Ulist	Daftar user dari pwfile
PWFILE	File berisi user password

Tabel 3.2. Tabel data user administrator

Pada module *Schedule Cron Job* dibutuhkan data-data yang akan digunakan untuk menampilkan seluruh cronjob yang terdapat dalam sistem.

Data	Keterangan
TAB	Daftar seluruh cron job yang dimiliki oleh user
Jlist	Daftar seluruh cron job beserta user name dan perintahnya
cron_allow_file	Daftar user diterima (allowed)
cron_deny_file	Daftar user ditolak (denied)

Tabel 3.3. data Cron job

Modul export membutuhkan data yang berisi seluruh export file atau directory yang ada di dalam sistem untuk digunakan sebagai input untuk proses *editing* export.

Data	Keterangan
Exp	Berisi daftar export

Tabel 3.4. data export

Pada tabel 3.5 merupakan data-data yang akan berfungsi untuk menyimpan informasi tentang protocol dan service yang terdapat di dalam sistem.

Data	Keterangan
Config{service_file}	Konfigurasi services dari service file
Config{Inetd_conf_file}	Berisi configurasi internet service
Config{protocol_file}	Berisi daftar protocols dalam sistem
SERVICES	Daftar service
INET	Konfigurasi Inetd

Tabel 3.5. Data service & protocol

Pada modul printer administration diperlukan tempat untuk menyimpan data-data yang akan digunakan untuk proses edit, delete, maupun create.

Data	**Keterangan**
Prn	Menampung nama printer
Jobs	Daftar printer Jobs
CAP	File princap
Plist	Daftar semua printer

Tabel 3.6. Data printer

Untuk menunjukkan status file-file maupun direktori tertentu dapat menggunakan data-data seperti di bawah ini untuk menyimpan informasi dari sistem tentang status file maupun direktori yang ada dalam sistem.

Data	**Keterangan**
MTAB	Daftar mount (/etc/mtab)
FSTAB	Daftar filesystem saat booting (/etc/fstab)

Tabel 3.7. Data Mount

Berikut adalah data-data yang diambil dari sistem yang akan digunakan untuk administrasi dan konfigurasi network .

Data	**Keterangan**
HOTS	Daftar hosts dan alamat
IFC	Daftar Interface
RESOLV	Konfigurasi dns
STATIC	Konfigurasi rute statis

Tabel 3.8. Data Network

Untuk melakukan proses administrasi terhadap unix/linux user dibutuhkan informasi tentang user, group dan password. Untuk menyimpan informasi tersebut dibutuhkan data seperti daftar linux user maupun daftar group user.

Data	Keterangan
Passwd	Daftar linux user
Group	Daftar linux group
Pam	Daftar Password acak
Base uid	UID untuk user
Base gid	GID untuk group

Tabel 3.9. Data Linux user

3.3. Perancangan Proses.

Sebelum melakukan proses pengiriman dan penerimaan data (HTTP *Interactions*), terlebih dahulu dilakukan proses membuka socket. Socket dapat diibaratkan sebagai *chanel* pada sebuah radio yang mempunyai gelombang-gelombang tertentu dalam proses pengiriman dan penerimaan data. Apabila sebuah socket dibuka dengan "gelombang" tertentu, maka hanya komputer dengan "gelombang" yang sama yang dapat menerima data yang dikirim. Setelah socket dibuka, maka dialokasikan alamat protokol lokal ke dalamnya. Alamat protokol terdiri dari kombinasi 32-bit sampai dengan 128-bit alamat internet (IP Address) ditambah dengan 16-bit nomor port pada protokol TCP dan UDP. Proses pengalokasian alamat protokol disebut proses **BIND**. Socket yang telah diaktifkan dan dialokasikan kemudian bersiap untuk merima data yang dikirim oleh komputer lain (***listen***). Pada server, proses ini digunakan untuk menunggu koneksi dari client. Setiap koneksi ditampung dalam satu antrian dan secara otomatis server akan mengatur koneksi-koneksi tersebut serta memilih koneksi mana yang benar-benar komplit dan siap untuk ditanggani (***accept***). Client yang sudah terkoneksi dapat melakukan proses pengiriman dan penerimaan data antar client-client yang lain.

Server akan menunggu koneksi terhadap network pada port, ketika client terhubung dengan port, server akan menerima koneksi tersebut dan kemudian akan berhubungan dengan client dengan menggunakan protocol yang mereka sepakati (tcp, http, nntp, smtp, dll).

Server akan menggunakan *socket() system call* untuk menciptakan socket, dan *bind() call* akan menugaskan socket pada port tertentu pada host. Server kemudian akan menggunakan rutin *listen()* dan *accept()* untuk menciptakan komunikasi pada port. Pada sisi yang lain client juga menggunakan *socket() system call* to create socket dan kemudian akan menggunakan *connect() call* untuk inisialisasi koneksi yang berasosiasi dengan socket pada *remote* host tertentu dan port. Server menggunakan accept() call untuk menerima koneksi yang datang dan inisialisasi komunikasi dengan client. Client dan server akan menggunakan *sysread()* dan *syswrite()* untuk berkomunikasi pada HTTP sampai transaksi selesai. Apabila transaksi telah selesai maka baik client maupun server akan menggunakan *close()* atau *shutdown()* untuk mengakhiri koneksi. Proses-proses yang terjadi diatas akan dijelaskan dalam sebagai berikut:

3.3.1. Proses transaksi http.

Sebelum memasuki proses penerimaan dan pengiriman data yang merupakan inti dari sistem aplikasi ini, diperlukan beberapa deklarasi awal yang berlaku secara global atau menyeluruh disemua proses-proses yang ada didalam sistem. Setelah itu dilakukan inisialisasi terhadap proses dan prosedur yang telah dibuat tersebut sebagai tanda dimulainya proses pengiriman dan penerimaan data.

a. Inisialisasi socket

Salah satu deklarasi yang perlu didefinisikan adalah terlebih dahulu adalah socket. *Function* dari pendeklarasian tersebut adalah sebagai berikut :

```
$perl -e 'use Socket; socket(FOO, PF_INET, SOCK_STREAM,
getprotobyname("tcp")); setsockopt(FOO, SOL_SOCKET, SO_REUSEADDR,
pack("l", 1)); bind(FOO, sockaddr_in($ARGV[0], INADDR_ANY)) || exit(1);
exit(0);' $port
    if [ $? != "0" ]; then
            echo "ERROR: TCP port $port is already in use by another program"
            echo ""
            exit 13
```

Baik client dan server keduanya sama-sama menggunakan fungsi *socket()* untuk menciptakan I/O buffer dalam sistem operasi. *Socket()* memerlukan beberapa argumen yang berupa *file handle* yang berasosiasi dengan socket, network protocol, dan bagaimana tipe socket, *stream-oriented* atau *record-oriented*. Untuk transaksi HTTP, tipe socket yang digunakan adalah tipe *stream-oriented*. Dalam contoh berikut di bawah ini, file handle *SH* berasosiasi dengan socket yang baru diciptakan. *PF_INET* adalah indikasi dari internet protocol, sedangkan *getprotobyname('tcp')* merupakan indikasi dari *Transmission Control Protocol* (TCP), sedangkan *SOCK_STREAM* adalah indikasi dari tipe socket stream-oriented. Bila socket calls gagal maka program akan ***die()*** menggunakan pesan kesalahan (*error message*) yang terdapat dalam $!. Di bawah adalah berbagai macam variasi yang biasa digunakan untuk melalukan insialisasi socket:

```
socket(SH, PF_INET, SOCK_STREAM, getprotobyname("tcp")) || $!;

$proto = getprotobyname('tcp');
socket(MAIN, PF_INET, SOCK_STREAM, $proto) ||
    die "Failed to open main socket : $!";
```

```
socket($h, PF_INET, SOCK_STREAM, getprotobyname("tcp")) ||
        &error("Failed to create socket : $!");
```

Kemudian socket diinisialisasi (dibuka) sesuai dengan jenis protocol yang digunakan. Berikut ini adalah inisialisasi socket dengan menggunakan protocol TCP secara lengkap.

```
# Open main socket  {miniserv.pl}
$proto = getprotobyname('tcp');
socket(MAIN, PF_INET, SOCK_STREAM, $proto) ||
die "Failed to open main socket : $!";
setsockopt(MAIN, SOL_SOCKET, SO_REUSEADDR, pack("l", 1));
$baddr = $config{"bind"} ? inet_aton($config{"bind"}) : INADDR_ANY;
bind(MAIN, sockaddr_in($config{port}, $baddr)) ||
        die "Failed to bind port $config{port} : $!";
listen(MAIN, SOMAXCONN);
# Read the HTTP request and headers
($reqline = &read_line()) =~ s/\r\n//g;
if (!($reqline =~ /^(GET|POST)\s+(.*)\s+HTTPV1\..$/)) {
        &http_error(400, "Bad Request");
        }
$method = $1; $request_uri = $page = $2;
%header = ();
while(1) {
        ($headline = &read_line()) =~ s/\r\n//g;
        if ($headline eq "") { last; }
        ($headline =~ /^(\S+):\s+(.*)$/) || &http_error(400, "Bad Header");
        $header{lc($1)} = $2;
        }
if (defined($header{'host'})) {
        if ($header{'host'} =~ /^([^:]+):([0-9]+)$/) { $host = $1; }
        else { $host = $header{'host'}; }
        }
if ($page =~ /^([^\?]+)\?(.*)$/) {
        # There is some query string information
        $page = $1;
        $querystring = $2;
        if ($querystring !~ /=/) {
                $queryargs = $querystring;
                $queryargs =~ s/\+/ /g;
                $queryargs =~ s/%(..)/pack("c",hex($1))/ge;
```

```
            $querystring = "";
            }
    }
```

b. Membuat koneksi network

Dengan mengunakan fungsi *connect()* dapat dibuat sebuah hubungan dengan server dalam sebuah network (*network connections*) berikut host dan port yang dikehendaki, dan menyelaraskan (*associate*) dengan I/O buffer yang telah diciptakan oleh *socket()*.

```
$sin = sockaddr_in (80, inet_aton, ('localhost.localdomain.com'));
connect(SH,$sin) || die  $!;
```

Rutin *sockaddr_in()* menerima sebuah *port number* sebagai parameter pertama dan *IP address* sebaggai parameter kedua. Inet_aton() menerjemahkan (*translates*) sebuah *hostname string* atau sebuah *dotted decimal string* menjadi sebuah IP address 32-bit. *sockaddr_in* mengembalikan struktur data yang datang melalui fungsi *connect()*, dari sanalah *connect()* mampu melakukan proses koneksi pada sebuah jaringan komputer pada server dan port secara spesifik. Berikut adalah penerapan fungsi *connect()* pada aplikasi:

```
($addr = inet_aton($_[0])) ||
    &error("Failed to lookup IP address for $_[0]");
connect($h, sockaddr_in($_[1], $addr)) ||
    &error("Failed to connect to $_[0]:$_[1] : $!");
```

Apabila koneksi terhadap *network* tersebut sukses, maka nilai yang kembali adalah *true*, namun apabila gagal maka nilai *false* akan diberikan, untuk memberikan pesan kesalahan (*error message*) dapat digunakan $!. Gunakan *die()*

setelah *connect()* untuk menghentikan program dan melaporkan kesalahan. Fungsi *connect()* ini hanya dapat digunakan oleh client saja.

```
# open_socket(host, port, handle) <weblib.pl>
sub open_socket
{
local($addr, $h); $h = $_[2];
socket($h, PF_INET, SOCK_STREAM, getprotobyname("tcp")) ||
      &error("Failed to create socket : $!");
($addr = inet_aton($_[0])) ||
      &error("Failed to lookup IP address for $_[0]");
connect($h, sockaddr_in($_[1], $addr)) ||
      &error("Failed to connect to $_[0]:$_[1] : $!");
select($h); $| =1; select(STDOUT);
}
```

c. Menulis data pada koneksi network

Untuk menulis *file handle* (data) yang berasosiasi dengan socket (network connection), dapat digunakan rutin *syswrite()*. Parameter pertama merupakan file handle untuk menulis data, sedang data yang akan ditulis dijadikan parameter kedua. Sedangkan parameter ketiga adalah panjang data yang akan ditulis. *Syswrite()* digunakan oleh client dan server.

```
$buffer = ”Linux Web Administration”;
          syswrite (FH, $buffer, length($buffer));
# sysprint(handle, [string]+)
sub sysprint
{
local($str, $fh);
$str = join(", @_[1..$#_]);
$fh = $_[0];
syswrite $fh, $str, length($str);
}
```

d. Membaca data dari koneksi network

Untuk membaca data dari sebuah koneksi network dapat digunakan rutin *sysread()* fungsi tersebut dapat berjalan baik pada server maupun client *sysread()* membaca data dari *file handle* yang berasosiasi dengan socket tersebut, pada parameter pertama file handle diberikan untuk menerangkan koneksi yang akan dibaca. Sedangkan parameter kedua menunjukkan sebuah variabel scalar untuk menyimpan data yang telah dibaca. Parameter ketiga menunjukkan bytes maksimum yang akan dibaca dari sebuah koneksi.

```
Sysread(FH, $buffer, 200);  # read at most 200 bytes from FH
$buffer = <FH>;
```

e. Mengakhiri Koneksi

Setelah transaksi selesai (*complete*), rutin *close()* akan mengakhiri koneksi jaringan (*closing network connections*). Fungsi ini dapat digunakan baik oleh client maupun server.

```
close(FH);
```

f. Mendengarkan port

Fungsi *bind()* digunakan hanya oleh *server*. Rutin *bind()* menyelaraskan (*associate*) antara socket buffer dengan port pada perangkat keras (computer). Apabila port yang dimaksud telah digunakan oleh program lain maka *bind()* akan memberikan nilai *false* (*zero*). *Sockaddr_in()* dapat digunakan untuk identifikasi atau mengenali port untuk *bind()*. Definisi port dapat berupa variabel dapat juga ditulis secara langsung. Proses ini dikenal dengan *Binding the Port.*

```
bind(FOO, sockaddr_in($ARGV[0], INADDR_ANY)) || exit(1); exit(0);' $port
```

```
bind(MAIN, sockaddr_in($config{port}, $baddr)) ||
    die "Failed to bind port $config{port} : $!";
```

Socket yang telah diinisialisasi atau diaktifkan, menandai dibukanya jalur akses ke alamat lokal protokol. Oleh karena itu diperlukan proses untuk memasuki jalur yang telah disediakan oleh socket tersebut.

Fungsi bind akan sukses atau berhasil dijalankan jika nilai yang dihasilkan berupa 0 (null) dan bernilai –1 (socket_error) jika tidak berjalan semestinya. Proses bind akan menspesifikasikan alamat lokal dan port yangg sudah didefinisikan sebelumnya. Alamat lokal yang tidak didefinisikan secara jelas (inaddr_any), akan menonaktifkan sistem untuk memilih alamat lokal sampai adanya koneksi dari socket yang lain. Dengan demikian default alamat lokal dengan INADDR_ANY sangat effisien dipakai diawal pertama kali socket diaktifkan.

g. Menunggu koneksi

Fungsi *listen()* hanya digunakan oleh *server* untuk menunggu datangnya koneksi dari *client*. Fungsi *listen()* memberitahukan pada sistem operasi bahwa server siap untuk menerima koneksi jaringan yang akan masuk pada port yang telah ditentukan. Parameter pertama merupakan file handle pada socket untuk melakukan proses listen.

```
listen(MAIN, SOMAXCONN);
```

h. Menerima Koneksi

Function *Accept()* digunakan hanya pada *server*, fungsi ini bertujuan untuk

menerima datangnya koneksi dari *client*. Dengan kata lain *Accept()* berfungsi untuk menunggu datangnya permintaan (*request*) pada server. Untuk parameter *accept()* menggunakan dua *file handle*, satu berassosiasi dengan socket sedang yang lain berassosiasi dengan koneksi *network* tertentu (*specifik network connections*).

```
# got new connection
    $acptaddr = accept(SOCK, MAIN);
    if (!$acptaddr) { next; }
```

Proses yang terjadi pada program aplikasi ini akan dikelompokkan berdasar fungsi-fungsi yang ada dalam program aplikasi.

i. Proses penerimaan data

Pada saat data yang diminta dikirim segera dijalankan proses penerimaan data. Proses ini merupakan tahap dari sistem untuk menerima koneksi dari socket lain. Setelah alamat dan port sudah teralokasikan melalui proses bind, dapat diartikan bahwa socket telah diaktifkan untuk menerima koneksi dari socket lain serta membuka jalur bagi data-data yang akan memasuki sistem (receive).

```
# read_line()
# Reads one line from SOCK
sub read_line
{
local($idx, $more, $rv);
if ($use_ssl) {
    while(($idx = index($read_buffer, "\n")) < 0) {
        # need to read more..
        if (!($more = Net::SSLeay::read($ssl_con))) {
            # end of the data
            $rv = $read_buffer;
            undef($read_buffer);
            return $rv;
```

```
                    }
                $read_buffer .= $more;
                }
        $rv = substr($read_buffer, 0, $idx+1);
        $read_buffer = substr($read_buffer, $idx+1);
        return $rv;
        }
else { return <SOCK>; }
}
```

Fungsi receive form berfungsi sebagai jalur pembuka yang selalu siap menangani setiap socket yang berinteraksi.

j. Proses pengiriman data

Proses listen atau proses penerimaan data diulang secara terus menerus agar sistem selalu siap menerima semua paket data yang yang ditujukan ke alamat sistem. Proses penerimaan data sistem bersifat pasif yaitu menunggu datangnya data sehingga sistem harus selalu *stand by.*

```
# write_data(data)
# Writes a string to SOCK
sub write_data
{
if ($use_ssl) {
        Net::SSLeay::write($ssl_con, $_[0]);
        }
else {
        print SOCK $_[0];
        }
$write_data_count += length($_[0]);
}
```

3.3.2. Proses dalam Modul

Proses modul adalah proses pasif, artinya proses tersebut tidak akan dijalankan apabila tidak ada perintah dari user, meskipun user telah terlebih dahulu

mengaktifkan proses transaksi http.

a. Admin Users

Admin user adalah sebuah tools untuk membuat, edit, menghapus, user administrator, atau orang-orang yang memiliki hak untuk melakukan proses administrasi dengan menggunakan program aplikasi ini. Dalam tools ini seorang administrator dapat mengatur fasilitas-fasilitas mana saja yang diperkenankan untuk dipakai oleh setiap user administrator.

Untuk menampilkan data-data user yang berhak untuk menggunakan program admin dibuat subrutin yang berfungsi untuk mengambil data user yang berada dalam /etc/admin. Data tersebut kemudian ditempatkan dalam sebuah file handle bernama PWFILE. PWFILE akan disimpan dalam array untuk ditampilkan pada form yang disediakan.

```
sub list_users
{
local(@mods, %miniserv, $_, @rv, %acl);
&read_acl(undef, \%acl);
@mods = &list_modules();
&get_miniserv_config(\%miniserv);
open(PWFILE, $miniserv{'userfile'});
while(<PWFILE>) {
    if (/^([^:\s]+):([^:\s]+)(:(\d+))?/) {
            local(%user);
            $user{'name'} = $1;
            $user{'pass'} = $2;
            $user{'sync'} = $4 if ($3);
            $user{'modules'} = $acl{$1};
            push(@rv, \%user);
            }
    }
close(PWFILE);
```

Untuk menciptakan user baru bagi administrator maka perlu membuka file handle PWFILE. Program admin akan menerima input dari user, kemudian akan menjalan subrutin create_user yang akan menambahkan input dari user untuk disimpan dalam /etc/admin.

```
# create_user(&details)
sub create_user
{
local(%user, %miniserv, @mods);
%user = %{$_[0]};
&get_miniserv_config(\%miniserv);
open(PWFILE, ">> $miniserv{'userfile'}");
print PWFILE "$user{'name'}:$user{'pass'}:$user{'sync'}\n";
close(PWFILE);
```

Sama dengan ketika membuat user baru, untuk merubah user data-data tentang user perlu ditampilkan terlebih dahulu, user akan menekan tombol save apabila telah selesai melakukan perubahan. Ketika tombol save ditekan maka subrutin modify user akan dijalankan. File handle PWFILE yang menyimpan data user akan ditulis kembali menggantikan data yang lama.

```
# modify_user(name, &details)
sub modify_user
{
local(%user, %miniserv, @pwfile, @acl, @mods, $_, $m);
%user = %{$_[1]};
&get_miniserv_config(\%miniserv);
open(PWFILE, $miniserv{'userfile'});
@pwfile = <PWFILE>;
close(PWFILE);
open(PWFILE, "> $miniserv{'userfile'}");
foreach (@pwfile) {
      if (/^([^:\s]+):([^:\s]+)/ && $1 eq $_[0]) {
                print PWFILE "$user{'name'}:$user{'pass'}:$user{'sync'}\n";
                }
      else { print PWFILE $_; }
```

```
    }
close(PWFILE);
```

Data user masih harus ditampilkan apabila hendak menghapus user yang telah ada. Input dari user yang berupa penekanan tombol delete akan menjalankan subrutin delete_user. Data yang tersimpan dalam PWFILE akan disimpan kembali menggantikan data sebelumnya.

```
# delete_user(name){func del_usr-/acl/acl-lib.pl}
sub delete_user
{
local($_, @pwfile, @acl, %miniserv);
&get_miniserv_config(\%miniserv);
open(PWFILE, $miniserv{'userfile'});
@pwfile = <PWFILE>;
close(PWFILE);
open(PWFILE, "> $miniserv{'userfile'}");
foreach (@pwfile) {
    if (!/^([^:\s]+):([^:\s]+)/ || $1 ne $_[0]) { print PWFILE $_; }
    }
close(PWFILE);
```

Apabila ada kegagalan dalam proses penghapusan user maka error message dapat ditampilkan dalam $whatfailed.

```
# modify_user(name, &details){func edt_usr-/acl-lib.pl}
sub modify_user
{
local(%user, %miniserv, @pwfile, @acl, @mods, $_, $m);
%user = %{$_[1]};
&get_miniserv_config(\%miniserv);

open(PWFILE, $miniserv{'userfile'});
@pwfile = <PWFILE>;
close(PWFILE);
open(PWFILE, "> $miniserv{'userfile'}");
foreach (@pwfile) {
```

```
        if (/^([^:\s]+):([^:\s]+)/ && $1 eq $_[0]) {
                print PWFILE "$user{'name'}:$user{'pass'}:$user{'sync'}\n";
                }
        else { print PWFILE $_; }
        }
close(PWFILE);
```

b. Linux Users

Linux user adalah modul yang mengatur administrasi terhadap user-user linux. Dalam modul ini seorang administrator mampu untuk membuat user baru, menghapus user, atau melakukan proses editing terhadap *account* seorang user.

Untuk keperluan tersebut diperlukan sebuah fungsi yang dapat mencari acount dari linux user. Berikut adalah fungsi untuk mengetahui user-user yang memiliki hak untuk menggunakan linux.

```
# password_file(file)/{user-lib.pl}
sub password_file
{
if (!$_[0]) { return 0; }
elsif (open(SHTEST, $_[0])) {
        local($line);
        $line = <SHTEST>;
        close(SHTEST);
        return $line =~ /^\S+:\S*:/;
        }
else { return 0; }
}

# list_users()
sub list_users
{
# read the password file
. . . . . .
        # start by reading /etc/passwd
        $lnum = 0;
        open(PASSWD, $config{'passwd_file'});
        while(<PASSWD>) {
```

```
        s/\r|\n//g;
        if (/\S/ && !/^[#\+\-]/) {
                @pw = split(/:/, $_, -1);
                push(@rv, { 'user' => $pw[0],'pass' => $pw[1],
                        'uid' => $pw[2],      'gid' => $pw[3],
                        'real' => $pw[4],     'home' => $pw[5],
                        'shell' => $pw[6],    'line' => $lnum,
                        'num' => scalar(@rv) });
                $idx{$pw[0]} = $rv[$#rv];
                }
        $lnum++;
        }
close(PASSWD);
```

Fungsi ini mencari file yang berada pada lokasi /etc/passwd, yaitu lokasi untuk menyimpan data-data password dari linux user. Data-data ini kemudian dibaca dan disimpan dalam *hashtable array* bernama PASSWD, dari data ini akan diperoleh informasi tentang user, pass, uid, gid, real, home, shell karena setiap file yang diambil mengandung informasi tersebut untuk masing-masing user.

Pada prinsipnya untuk membuat user baru, dapat ditambahkan pada daftar user yang disimpan dalam passwd, untuk selanjutnya disimpan kembali dalam direktory /etc/passwd.

```
sub crete_user < Creates a new user with the given details >
{
local(%u) = %{$_[0]};
if (&passfiles_type() == 1) {
      # just need to add to master.passwd
      open(PASSWD, ">> $config{'master_file'}");
      print PASSWD
"$u{'user'}:$u{'pass'}:$u{'uid'}:$u{'gid'}:$u{'class'}:$u{'change'}:$u{'expire'}:$
u{'real'}:$u{'home'}:$u{'shell'}\n";
      close(PASSWD);
      }
```

```
else {

# add to /etc/passwd
      open(PASSWD, ">> $config{'passwd_file'}");
      print PASSWD "$u{'user'}:",&passfiles_type() == 2 ? "x" : $u{'pass'},
                  ":$u{'uid'}:$u{'gid'}:$u{'real'}:$u{'home'}:$u{'shell'}\n";
      close(PASSWD);
      if (&passfiles_type() == 2) {
                  # add to shadow as well..
                  open(SHADOW, ">> $config{'shadow_file'}");
                  print SHADOW
"$u{'user'}:$u{'pass'}:$u{'change'}:$u{'min'}:$u{'max'}:$u{'warn'}:$u{'inactive'
}:$u{'expire'}:\n";
                  close(SHADOW);
                  }
      }
}
```

Hal yang sama terjadi juga terhadap fungsi modify user. Data-data dapat diambil dari passwd, kemudian merubah data tersebut dan menyimpannya kembali dalam lokasi yang sama.

```
# modify_user(&old, &details)
sub modify_user
{
local(%u) = %{$_[1]};
local(@passwd, @shadow);
if (&passfiles_type() == 1) {
      # just need to update master.passwd
      &replace_file_line($config{'master_file'}, $_[0]->{'line'},
          "$u{'user'}:$u{'pass'}:$u{'uid'}:$u{'gid'}:$u{'class'}:".
          "$u{'change'}:$u{'expire'}:$u{'real'}:$u{'home'}:$u{'shell'}\n");
      }
else {

# update /etc/passwd
      &replace_file_line($config{'passwd_file'}, $_[0]->{'line'},
                  "$u{'user'}:".(&passfiles_type() == 2 ? "x" : $u{'pass'}).
                  ":$u{'uid'}:$u{'gid'}:$u{'real'}:$u{'home'}:$u{'shell'}\n");
      if (&passfiles_type() == 2) {
                  # update shadow file as well..
```

```
                &replace_file_line($config{'shadow_file'}, $_[0]->{'sline'},
                        "$u{'user'}:$u{'pass'}:$u{'change'}:$u{'min'}:".
                        "$u{'max'}:$u{'warn'}:$u{'inactive'}:$u{'expire'}:\n");
                }
        }
}
```

Berikut adalah fungsi yang bertugas untuk menghapus user. Masih memanfaatkan daftar user passwd.

```
# delete_user(&details)
sub delete_user
{
if (&passfiles_type() == 1) {
        &replace_file_line($config{'master_file'}, $_[0]->{'line'});
        }
else {
        &replace_file_line($config{'passwd_file'}, $_[0]->{'line'});
        if (&passfiles_type() == 2) {
                &replace_file_line($config{'shadow_file'}, $_[0]->{'sline'});
                }
        }
}
```

Untuk melakukan administrasi pada group masih sama seperti diatas, informasi tentang group, pass, gid, members disimpan dalam array hashtable bernama *GROUP*.

```
# list_groups()
sub list_groups
{
local(@rv, $lnum, $_, %idx, $g, $i, $j);
$lnum = 0;
open(GROUP, $config{'group_file'});
while(<GROUP>) {
        s/\r|\n//g;
        if (/\S/ && !/^[#\+\-]/) {
                @gr = split(/:/, $_, -1);
                push(@rv, { 'group' => $gr[0],'pass' => $gr[1],
```

```
                        'gid' => $gr[2],'members' => $gr[3],
                        'line' => $lnum,'num' => scalar(@rv) });
            $idx{$gr[0]} = $rv[$#rv];
            }
    $lnum++;
    }
close(GROUP);
```

Untuk fungsi-fungsi yang lain seperti, menambah group baru, edit group, atau menghapus group pada prinsipnya sama seperti yang terjadi pada user. Perbedaannya hanya terdapat pada ulist dan glist nya saja.

c. Administrasi Printer

Dengan memanfaatkan **PRINCAP** (Printer Capability Database) dapat dibuat fungsi-fungsi yang berguna untuk menambah, menghapus maupun edit printer. Princap berfungsi untuk menerjemahkan line printer. Berikut adalah program untuk mendapatkan printer sekaligus untuk mengetahui status printer tersebut.

```
#Sub_get_printer
# found the printer.. get info from printcap
$prn{'name'} = $n[0];
if (@n > 2) { $prn{'alias'} = [ @n[1..$#n-1] ]; }
if (@n > 1) { $prn{'desc'} = $n[$#n]; }
$prn{'iface'} = $l->{'if'};
$prn{'banner'} = !defined($l->{'sh'});
$prn{'dev'} = $l->{'lp'};
$prn{'rhost'} = $l->{'rm'};
$prn{'rqueue'} = $l->{'rp'};
$prn{'msize'} = $l->{'mx#'};
# call lpc to get status
$out = `lpc status $prn{'name'} 2>&1`;
$prn{'accepting'} = ($out =~ /queuing is enabled/);
$prn{'enabled'} = ($out =~ /printing is enabled/);
# call lpq to get print jobs
open(LPQ, "lpq -P$prn{'name'} |");
```

```
while(<LPQ>) {
chop;
if (/^Rank\s+Owner\s+/) { $doneheader++; }
elsif ($doneheader && /^(\S+)\s+(\S+)\s+(\d+)\s+(.*\S)\s+(\d+)\s+(\S+)$/) {
local(%job);
$job{'id'} = $3;
$job{'user'} = $2;
$job{'size'} = $5;
$job{'file'} = $4;
$job{'printing'} = ($1 eq "active");
push(@jobs, \%job);
```

Untuk menambah account printer baru dalam direktori /etc/printcap, panggil file handle CAP simpan dalam config{'printcap_file'} cetak CAP dan jalankan subrutin make_printcap, maka data baru yang diinputkan oleh user akan disimpan dalam printcap_file.

```
# create_printer(&details)
sub create_printer
{
local(%cap);
$cap{'sd'} = "$config{'spool_dir'}/$_[0]->{'name'}";
mkdir($cap{'sd'}, 0755);
open(CAP, ">> $config{'printcap_file'}");
print CAP &make_printcap($_[0], \%cap),"\n";
close(CAP);
&apply_status($_[0]);
}
```

Untuk merubah account yang sudah ada dapat mengambil daftar printer dalam list_printcab yang kemudian disimpan dalam sebuah array. Nama printer yang akan dihapus merupakan input dari user. Bila nama yang diinginkan tidak terdapat dalam daftar printer akan ditampilkan pesan kesalahan. Namun bila printer ditemukan akan ditampilkan.

```
# modify_printer(&details)
sub modify_printer
{
local(@old, $o, $old, @cap);
@old = &list_printcap();
foreach $o (@old) {
	$o->{'name'} =~ /^([^\|]+)/;
	if ($1 eq $_[0]->{'name'}) {
		# found current details
		$old = $o;
		last;
		}
	}
if (!$old) { &error("Printer '$_[0]->{'name'}' no longer exists"); }
open(CAP, $config{'printcap_file'});
@cap = <CAP>;
close(CAP);
splice(@cap, $old->{'line'},
	$old->{'eline'} - $old->{'line'} + 1, &make_printcap($_[0], $old)."\n");
open(CAP, "> $config{'printcap_file'}");
print CAP @cap;
close(CAP);
&apply_status($_[0]);
}
```

Setelah daftar printer (list_printcap) yang diketahui sistem ditampilkan, user dapat melakukan proses penghapusan data yang ada dalam daftar tersebut melalui *interface* yang telah disediakan. Setelah itu data yang terdapat dalam CAP disimpan lagi dalam config{'printcap_file'}. Dengan demikian otomatis data CAP yang lama akan ditimpa (*overwrite*) dengan CAP yang baru.

```
# delete_printer(name)
sub delete_printer
{
local(@old, $o, $old, @cap);
@old = &list_printcap();
foreach $o (@old) {
	$o->{'name'} =~ /^([^\|]+)/;
	if ($1 eq $_[0]) {
```

```
                # found current details
                $old = $o;
                last;
                }
        }
if (!$old) { &error("Printer '$_[0]' no longer exists"); }
open(CAP, $config{'printcap_file'});
@cap = <CAP>;
close(CAP);
splice(@cap, $old->{'line'}, $old->{'eline'} - $old->{'line'} + 1);
open(CAP, "> $config{'printcap_file'}");
print CAP @cap;
close(CAP);
}
```

d. Schedule Cron Job

Untuk mengetahui nama user yang memiliki cronjob, dibuat file handle bernama TAB. Tab berisi configurasi dari sistem_crontab dari direktori /etc/crontab.

```
sub list_users
{
local($f, @rv, @uinfo);
opendir(DIR, $config{cron_dir});
while($f = readdir(DIR)) {
        if ($f =~ /\./) { next; }
        if (@uinfo = getpwnam($f)) {
                push(@rv, $f);
                }
        }
           open(TAB, $config{'system_crontab'});

# read_crontab(user)
# Return an array containing the lines of the cron table for some user
sub read_crontab
{
local(@tab);
open(TAB, "$config{cron_dir}/$_[0]");
@tab = <TAB>;
close(TAB);
```

Untuk mendapatkan daftar cronjob yang dimiliki oleh tiap-tiap user file handle TAB harus dibuka terlebih dahulu. Informasi dari Tab ditempatkan dalam array untuk dibuat daftar list job .

```
sub list_jobs
{
local(@rv, $_);
open(TAB, "$config{cron_dir}/$_[0]");
while(<TAB>) {
	chop;
		# Keep this line..
		push(@rv, $_);
		}
	}
close(TAB);
	open(TAB, $config{'system_crontab'});
			push(@rv, "$1$4");
			}
		}
	close(TAB);
	}
return @rv;
}
```

Untuk membuat cronjob baru, file handle Tab yang telah dibuka menerima input dari form yang telah diisi oleh user kemudian isi dari tab akan disimpan (ditambahka) pada TAB yang ada.

```
sub create_job
{
local(@tab);
@tab = &read_crontab($_[0]);
open(TAB, "> $cron_temp_file");
print TAB @tab;
print TAB ($_[1] ? "" : "# "),join(' ' , @_[2 .. @_-1]),"\n";
close(TAB);
&copy_crontab($_[0]);
}
```

Merubah cronjob yang telah ada sebelumnya. Dengan menampilkannya terlebih dahulu. Setelah user melakukan perubahan Tab akan disimpan kembali menimpa data Tab yang lama sehingga Isi dari Tab telah berubah menjadi Tab yang baru.

```
sub change_job
{
local(@tab, $i, $_);
@tab = &read_crontab($_[0]);
open(TAB, "> $cron_temp_file");
$i = 0;
foreach(@tab) {
	chop;
	if (/^(#+)?\s*[0-9\-\*\/,]+\s+[0-9\-\*\/,]+\s+[0-9\-\*\/,]+\s+[0-9\-\*\/,]+\s+[0-
9\-\*\/,]+\s+.*$/ && $i++ == $_[1]) {
		# Found line to replace
		print TAB ($_[2] ? "" : "# "),join(' ' , @_[3 .. @_-1]),"\n";
		$found = 1;
		}
	else { print TAB $_,"\n"; }
	}
close(TAB);
if ($found) { &copy_crontab($_[0]); }
elsif ($config{'vixie_cron'} && $_[1] >= $i) {
	# need to update the system crontab file instead
	open(TAB, $config{'system_crontab'});
	@tab= <TAB>;
	close(TAB);
	open(TAB, "> $config{'system_crontab'}");
	foreach (@tab) {
		chop;
		# Found line to replace
			print TAB ($_[2] ? "" : "# "),
				"$_[3] $_[4] $_[5] $_[6] $_[7] $_[0] $_[8]\n";
			}
		else { print TAB $_,"\n"; }
		}
	close(TAB);
	}
}
```

Untuk menghapus cronjob, selanjutnya sistem crontab akan diupdate atau diganti dengan sistem yang baru, dengan cara menimpa sistem yang lama.

```
sub delete_job
{
local(@tab, $i, $_);
@tab = &read_crontab($_[0]);
open(TAB, "> $cron_temp_file");
$i = 0;
foreach (@tab) {
	chop;
	# This is not the line to delete..
		print TAB $_,"\n";
		}
	else { $found = 1; }
	}
close(TAB);
if ($found) { &copy_crontab($_[0]); }
elsif ($config{'vixie_cron'} && $_[1] >= $i) {
	# need to update the system crontab file instead
	open(TAB, $config{'system_crontab'});
	@tab= <TAB>;
	close(TAB);
	open(TAB, "> $config{'system_crontab'}");
	foreach (@tab) {
		chop;
		# dont delete this line
			print TAB $_,"\n";
			}
		}
	close(TAB);
	}
}
```

e. NFS Export

Untuk membuat daftar *export* file dibutuhkan sebuah data yang berfungsi untuk menyimpan informasi tentang export file.

```
# list_exports()
sub list_exports
```

```
{
local (@rv, $pos, $lnum, $h, $o);
return @list_exports_cache if (@list_exports_cache);
open(EXP, $config{'exports_file'});
$lnum = 0;
while(<EXP>) {
	s/\s+$//g;
	s/#.*$//g;
	if (/^\s*(\S+)\s+(.*)$/) {
		local $dir = $1;
		local $rest = $2;
		$pos = 0;
		while($rest =~ /^([^\s+\(\)]*)\(([^\)]*)\)\s*(.*)$/ ||
		      $rest =~ /^([^\s+\(\)]+)\s*()(.*)$/) {
			local %exp;
			$exp{'dir'} = $dir;
			$exp{'host'} = $1;
			local $ostr = $2;
			$rest = $3;
			while($ostr =~ /^([a-z_]+)=([0-9,\-]+)\s*,\s*(.*)$/ ||
			      $ostr =~ /^([a-z_]+)=([0-9,\-]+)(.*)$/ ||
			      $ostr =~ /^([a-z_]+)=([^,\s]+),(.*)$/ ||
			      $ostr =~ /^([a-z_]+)=([^,\s]+)(.*)$/ ||
			      $ostr =~ /^([a-z_]+)()\s*,\s*(.*)$/ ||
			      $ostr =~ /^([a-z_]+)()(.*)$/) {
				if ($2 ne "") { $exp{'options'}->{$1} = $2; }
				else { $exp{'options'}->{$1} = ""; }
				$ostr = $3;
				}
			$exp{'line'} = $lnum;
			$exp{'pos'} = $pos++;
			$exp{'index'} = scalar(@rv);
			push(@rv, \%exp);
			}
		}
	$lnum++;
	}
close(EXP);
@list_exports_cache = @rv;
return @list_exports_cache;
}
```

Untuk menciptakan export file baru, perlu membuka EXP file terlebih dahulu kemudian simpan EXP dalam export file dan menjalankan subrutin make export.

Daftar export baru tersebut akan ditambahkan dalam daftar export yang tersimpan dalam export_file.

```
# create_export(&export)
sub create_export
{
open(EXP, ">>$config{'exports_file'}");
print EXP &make_exports_line($_[0]);
close(EXP);
}
```

Data export yang diperoleh dari subrutin list_exports disimpan dalam bentuk array. Isi dari array tersebut akan ditampilkan dalam form. Apabila tidak ada perubahan, maka direktori tidak akan berubah. Namun apabila terjadi perubahan, maka data-data yang berubah akan disimpan menggantikan data yang lama.

```
sub modify_export
{
local @exps = &list_exports();
local @same = grep { $_->{'line'} eq $_[1]->{'line'} } @exps;
if ($_[0]->{'dir'} eq $_[1]->{'dir'} || @same == 1) {
	# directory not changed, or on a line of it's own
	splice(@same, &indexof($_[1],@same), 1, $_[0]);
	&replace_file_line($config{'exports_file'}, $_[1]->{'line'},
			   &make_exports_line(@same));
	}
else {
	# move to a line of it's own
	splice(@same, &indexof($_[1],@same), 1);
	&replace_file_line($config{'exports_file'}, $_[1]->{'line'},
			   &make_exports_line(@same));
	open(EXP, ">>$config{'exports_file'}");
	print EXP &make_exports_line($_[0]);
	close(EXP);
```

Sama dengan yang modul yang lain menghapus export file adalah sama dengan menghapus daftar export file.

```
sub delete_export
{
local @exps = &list_exports();
local @same = grep { $_ ne $_[0] && $_->{'line'} eq $_[0]->{'line'} } @exps;
if (@same) {
	# other exports on the same line.. cannot totally delete
	&replace_file_line($config{'exports_file'}, $_[0]->{'line'},
				&make_exports_line(@same));
	}
else {
	# remove export line
	&replace_file_line($config{'exports_file'}, $_[0]->{'line'});
	}
}
```

f. Konfigurasi network

Pada modul ini ada 4 (empat) tugas atau *task* yang termasuk dalam konfigurasi *network* yaitu *Network Interfaces, Routing and Gateways, DNS Client, Host Addresses.*

Untuk memperoleh daftar host maka dibuat data bernama HOSTS untuk menyimpan configurasi host file dalam /etc/host. Data yang ada didalam hosts yang berupa address, hosts, line, index ditampung dalam array bernama rv (reserve). Demikian seterusnya sampai selesai hingga terbentuk sebuah *list host*

```
# list_hosts()
# Parse hosts from /etc/hosts into a data structure
sub list_hosts
{
local @rv;
local $lnum = 0;
open(HOSTS, $config{'hosts_file'});
while(<HOSTS>) {
```

```
        s/\r|\n//g;
        s/#.*$//g;
        s/\s+$//g;
        if (/(\d+\.\d+\.\d+\.\d+)\s+(.*)$/) {
                push(@rv, { 'address' => $1,
                            'hosts' => [ split(/\s+/, $2) ],
                            'line', $lnum,
                            'index', scalar(@rv) });
                }
        $lnum++;
        }
close(HOSTS);
return @rv;
}
```

HOST dapat digunakan untuk menciptakan sebuah host address baru dengan menerima input dari user berupa hosts dan *address* selanjutnya disimpan dalam variabel scalar config{'host_file'}.

```
# create_host(&host)
# Add a new host to /etc/hosts
sub create_host
{
open(HOSTS, ">>$config{'hosts_file'}");
print HOSTS $_[0]->{'address'},"\t",join(" ",@{$_[0]->{'hosts'}}),"\n";
close(HOSTS);
}
```

Subrutin replace_file_line akan menggantikan data yang ditampilkan dalam form dengan data baru yang ditulis oleh user.

```
# modify_host(&host)
# Update the address and hosts of a line in /etc/hosts
sub modify_host
{
&replace_file_line($config{'hosts_file'},
                   $_[0]->{'line'},
                   $_[0]->{'address'}."\t".join(" ",@{$_[0]->{'hosts'}})."\n");
}
# delete_host(&host)
```

```
# Delete a host from /etc/hosts
sub delete_host
{
&replace_file_line($config{'hosts_file'}, $_[0]->{'line'});
}
```

Untuk mendapatkan konfigurasi dns, perlu untuk dibuat file dengan nama resolv yang berisi dari /etc/resolv.conf. Dari file tersebut didapat nama server, domain dan disimpan dalam array *hashtable.*

```
# get_dns_config()
sub get_dns_config
{
local $dns;
open(RESOLV, "/etc/resolv.conf");
while(<RESOLV>) {
	s/\r|\n//g;
	if (/nameserver\s+(.*)/) {
		push(@{$dns->{'nameserver'}}, split(/\s+/, $1));
		}
	elsif (/domain\s+(\S+)/) {
		$dns->{'domain'} = [ $1 ];
		}
	elsif (/search\s+(.*)/) {
		$dns->{'domain'} = [ split(/\s+/, $1) ];
		}
	}
close(RESOLV);
```

Untuk menyimpan kembali file dns_config, perlu membuka kembali file handle resolv yang berisi configurasi dns dalam file /etc/resolv.conf. tempatkan resolv dalam sebuah array, kemudian simpan hasilnya satu persatu dalam /etc/resolv.conf. secara langsung file handle resolv akan mengganti isi dari file /etc/resolv.conf

```
# save_dns_config(&config)
```

```
sub save_dns_config
{
open(RESOLV, "/etc/resolv.conf");
local @resolv = <RESOLV>;
close(RESOLV);
open(RESOLV, ">/etc/resolv.conf");
foreach (@{$_[0]->{'nameserver'}}) {
	print RESOLV "nameserver $_\n";
	}
if ($_[0]->{'domain'}) {
	if ($_[0]->{'domain'}->[1]) {
		print RESOLV "search ",join(" ", @{$_[0]->{'domain'}}),"\n";
		}
	else {
		print RESOLV "domain $_[0]->{'domain'}->[0]\n";
		}
	}
foreach (@resolv) {
	print RESOLV $_ if (!/^\s*(nameserver|domain|search)\s+/);
	}
close(RESOLV);
```

Untuk menyimpan kembali hostname, perlu membuka kembali file handle host yang berisi data hostname dalam file /etc/hostname. kemudian simpan hasilnya dalam file network_config. File handle host akan mengganti isi dari file /etc/sysconfig/network.

```
# save_hostname(name)
sub save_hostname
{
local %conf;
system("hostname $_[0] >/dev/null 2>&1");
open(HOST, ">/etc/HOSTNAME");
print HOST $_[0],"\n";
close(HOST);
&read_file("$network_config", \%conf);
$conf{'HOSTNAME'} = $_[0];
&write_file("$network_config", \%conf);
}
```

Static route config didapat dari /etc/sysconfig/static-routes dalam sistem, tempatkan file tersebut dalam sebuah variabel dan simpan isinya dalam file handle STATIC.

```
# get static routes
open(STATIC, $static_route_config);
while(<STATIC>) {
	if (/(\S+)\s+net\s+(\S+)\s+netmask\s+(\S+)\s+gw\s+(\S+)/) {
		push(@st, [ $1, $2, $3, $4 ]);
		}
	}
close(STATIC);
```

Perintah ifconfig digunakan untuk mengatur kernel-resident *network interface*. dieksekusi pada saat boot untuk mengatur *interface* yang ada dalam sistem, setelah itu hanya dibutuhkan ketika debuging atau ketika tuning sistem dibutuhkan.

ifconfig -a menampilkan status dari seluruh interface. Data tersebut kemudian dismpan dalam IFC. Dari ifc data tersebut dibuat daftar untuk ditampilkan ketika dibutuhkan.

```
# active_interfaces()
sub active_interfaces
{
local(@rv, @lines, $l);
open(IFC, "ifconfig -a |");
while(<IFC>) {
	s/\r|\n//g;
	if (/^\S+/) { push(@lines, $_); }
	else { $lines[$#lines] .= $_; }
	}
close(IFC);
foreach $l (@lines) {
	local %ifc;
	$l =~ /^([^:\s]+)/; $ifc{'name'} = $1;
```

```
        $l =~ /^(\S+)/; $ifc{'fullname'} = $1;
        if ($l =~ /^(\S+):(\d+)/) { $ifc{'virtual'} = $2; }
        if ($l =~ /inet addr:(\S+)/) { $ifc{'address'} = $1; }
        else { next; }
        if ($l =~ /Mask:(\S+)/) { $ifc{'netmask'} = $1; }
        if ($l =~ /Bcast:(\S+)/) { $ifc{'broadcast'} = $1; }
        if ($l =~ /HWaddr (\S+)/) { $ifc{'ether'} = $1; }
        if ($l =~ /MTU:(\d+)/) { $ifc{'mtu'} = $1; }
        $ifc{'up'}++ if ($l =~ /\sUP\s/);
        $ifc{'edit'} = ($ifc{'name'} !~ /^ppp/);
        $ifc{'index'} = scalar(@rv);
        push(@rv, \%ifc);
        }
return @rv;
}
```

g. Disk & Network file system

Berfungsi untuk menampilkan filesistem yang ada dan menunjukkan status mount. Untuk mendapatkan daftar mount dari file sistem perlu dipanggil subrutin list_mounts yang membuka **fstab** file pada direktori /etc/fstab. Informasi yang diperoleh dari fstab berupa *directory, device, type, options, fsck order, mount at boot* akan disimpan dalam array **mlist** untuk dibuat daftar mount (FSTAB) yang ada dan ditampilkan pada user.

```
sub get_mount
{
local(@mlist, $p, $d);
@mlist = &list_mounts();
for($i=0; $i<@mlist; $i++) {
        $p = $mlist[$i];
        if ($_[0] eq "*" && $p->[1] eq $_[1]) {
                # found by match on device
                return $i;
                }
        elsif ($_[1] eq "*" && $p->[0] eq $_[0]) {
                # found by match on directory
                return $i;
                }
```

```
        elsif ($p->[0] eq $_[0] && $p->[1] eq $_[1]) {
                # found by match on both
                return $i;
                }
        }
return -1;
```

Untuk menciptakan mount baru pada fstab file perlu membuka kembali FSTAB yang berisi informasi dari fstab file. Setelah user memberi input pada form yang telah disediakan, FSTAB akan ditulis kembali beserta dengan informasi tambahan dalam fstab file.

```
# create_mount(directory, device, type, options, fsck_order, mount_at_boot)
sub create_mount
{
local(@mlist, @amd, $_); local($opts);
        # Adding a normal mount to the fstab file
        open(FSTAB, ">> $config{fstab_file}");
        print FSTAB "$_[1]  $_[0]  $_[2]";
        close(FSTAB);
        }
}
```

Untuk mengganti mount sistem yang sudah ada dalam sistem file perlu dibuat daftar mount. FSTAB harus dibuka terlebih dahulu, kemudian simpan dalam array. Dari daftar mount akan dicari mount sistem manakah yang akan diganti sesuai dengan masukan dari user. Setelah baris informasi dalam daftar mount yang akan diganti ditemukan, FSTAB akan disimpan lagi dalam fstab file. Sehingga fstab file yang lama akan diganti dengan fstab yang baru, yang berisi informasi baru dari user.

```
# change_mount(num, directory, device, type, options, fsck_order,
mount_at_boot)
# Change an existing permanent mount
sub change_mount
```

```
{
local($i, @fstab, $line, $opts, $j, @amd);
$i = 0;

# Update fstab file
open(FSTAB, $config{fstab_file});
@fstab = <FSTAB>;
close(FSTAB);
open(FSTAB, "> $config{fstab_file}");
foreach (@fstab) {
	chop; ($line = $_) =~ s/#.*$//g;
	if ($line =~ /\S/ && $line !~ /\signore\s/ && $i++ == $_[0]) {
		# Found the line to replace
		print FSTAB "$_[2]  $_[1]  $_[3]";
		$opts = $_[4] eq "-" ? "" : $_[4];
		if ($_[6] eq "no") {
			$opts = join(',' , (split(/,/ , $opts) , "noauto"));
			}
		if ($opts eq "") { print FSTAB "  defaults"; }
		else { print FSTAB "  $opts"; }
		print FSTAB "  0  ";
		print FSTAB $_[5] eq "-" ? "0\n" : "$_[5]\n";
		}
	else { print FSTAB $_,"\n"; }
	}
close(FSTAB);
```

Untuk menghapus mount sistem yang sudah ada dalam sistem file perlu dibuat daftar mount. FSTAB harus dibuka terlebih dahulu, kemudian simpan dalam array. Daftar mount tersebut akan ditampilkan pada user. Setelah user memilih mount sistem mana yang akan dihapus FSTAB akan disimpan kembali dalam fstab file menimpa (overwrite) fstab file lama.

```
# delete_mount(index)
# Delete an existing permanent mount
sub delete_mount
{
local($i, @fstab, $line, $opts, $j, @amd);
$i = 0;
```

```
# Update fstab file
open(FSTAB, $config{fstab_file});
@fstab = <FSTAB>;
close(FSTAB);
open(FSTAB, "> $config{fstab_file}");
foreach (@fstab) {
	chop; ($line = $_) =~ s/#.*$//g;
	if ($line !~ /\S/ || $line =~ /\signore\s/ || $i++ != $_[0]) {
		# Don't delete this line
		print FSTAB $_,"\n";
		}
	}
close(FSTAB);
```

h. Internet services dan protocols

Dengan mengambil file /etc/inetd.conf dan /etc/services dapat diketahui konfigurasi file dari inetd. Untuk membuat daftar service dapat menggunakan file service yang berada dalam /etc/services yang kemudian disimpan dalam file handle bernama SERVICE. Data berebentuk baris tersebut berisi informasi tentang nama, port, protocol, alias. Data-data yang tersimpan dalam SERVICE tersebut kemudian dibuka dan diambil satu persatu dengan menggunakan array untuk ditampilkan pada form.

```
sub list_services
{
local(@rv, $l);
$l = 0;
open(SERVICES, $config{services_file});
while(<SERVICES>) {
	chop; s/#.*$//g;
	if (/^(\S+)\s+([0-9]+)\/(\S+)\s*(.*)$/) {
		push(@rv, [ $l, $1, $2, $3, $4 ]);
		}
	$l++;
	}
close(SERVICES);
return @rv;
```

```
}
```

Apabila user telah mengaktifkan tombol save atau create pada form pembuat service baru maka akan dijalankan program save_serv.cgi yang akan memanggil subrutin create_service. Sehingga data-data yang telah diisikan oleh user akan ditambahkan dalam service file

```
# create_service(name, port, proto, aliases)
# Add a new service to the list
sub create_service
{
open(SERVICES, ">> $config{services_file}");
print SERVICES "$_[0]\t$_[1]/$_[2]",$_[3] ? "\t$_[3]\n" : "\n";
close(SERVICES);
}
```

Daftar service dapat dirubah tetap dengan memanfaatkan SERVICE. Dengan memanfaatkan form (edit_inet.cgi) yang sama user dapat melakukan perubahan pada daftar service. Perbedaannya terletak pada standar input yang digunakan. Pada subrutin modify_service digunakan SERVICES yang berisi file baru akan menimpa services_file lama, sehingga data service_file akan berubah.

```
# modify_service(line, name, port, proto, aliases)
# Change an existing service
sub modify_service
{
local(@serv);
open(SERVICES, $config{services_file});
@serv = <SERVICES>;
close(SERVICES);
$serv[$_[0]] = "$_[1]\t$_[2]/$_[3]".($_[4] ? "\t$_[4]\n" : "\n");
open(SERVICES, "> $config{services_file}");
print SERVICES @serv;
close(SERVICES);
}
```

Pada subrutin delete_service, hampir sama dengan subrutin modify_service. File handle SERVICE melakukan *overwrite* pada services_file. Form yang digunakan masih sama dengan form yang digunakan untuk merubah service.

```
# delete_service(line)
sub delete_service
{
local(@serv);
open(SERVICES, $config{services_file});
@serv = <SERVICES>;
close(SERVICES);
splice(@serv, $_[0], 1);
open(SERVICES, "> $config{services_file}");
print SERVICES @serv;
close(SERVICES);
}
```

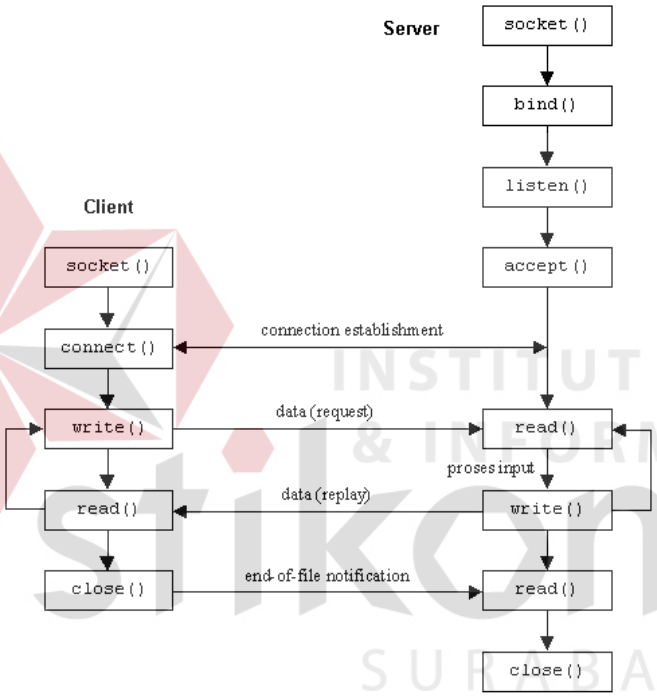
Untuk membuat daftar protocols digunakan file protokol yang terdapat dalam /etc/protocols. File protocol akan disimpan dalam file handle bernama PROT. Dimana isi dari PROT akan disimpan dalam array untuk kemudian ditampilkan pada form.

```
# list_protocols()
# Returns a list of supported protocols on this system
sub list_protocols
{
local(@rv);
open(PROT, $config{protocols_file});
while(<PROT>) {
	chop; s/#.*$//g;
	if (!/\S/) { next; }
	/^(\S+)\s+/;
	push(@rv, $1);
	}
close(PROT);
return @rv;
}
```

```
%prot_name = ("ip", "Internet Protocol",
              "tcp", "Transmission Control Protocol",
              "udp", "User Datagram Protocol");
```

3.4. Structure chart

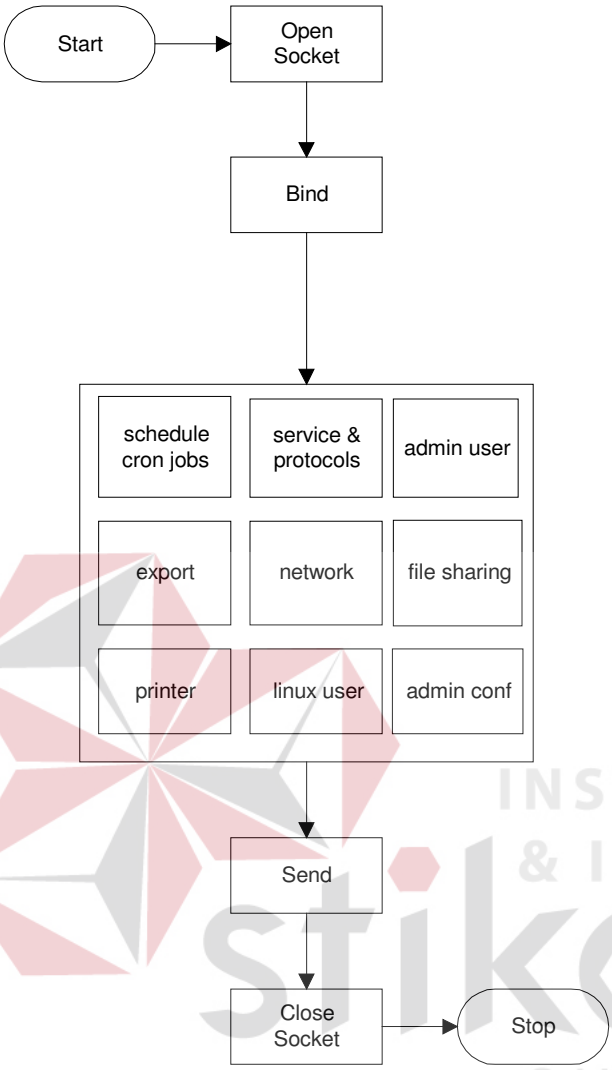
Berikut ini adalah structure chart yang menggambarkan hubungan antara client dan server dari uraian diatas.



Gambar 3.1. Structure Chart

3.5. Flow Chart

Dalam diagram alir ini digambarkan urut-urutan cara kerja program, diawali dari start sampai stop. Proses yang terjadi dalam modul tidak digambarkan alurnya sebab proses tersebut bersifat pasif. Fungsi-fungsi atau task yang berada di dalam modul akan dianggap sebagai menu.



Gambar 3.2. Diagram Alir