ANALISIS DAN IMPLEMENTASI KONFIGURASI KERNEL 2.4.18 BERBASIS LINUX PADA JARINGAN TANPA *HARDDISK* (*DISKLESS*)

A.Fahrudin[1] (10100176), Y.S.Aripin[1], M.Siallagan[2], A.Heryandi[2]
[1)] Jurusan Teknik Informatika
[2)] Dosen Jurusan Teknik Informatika
Fakultas teknik
UNIVERSITAS KOMPUTER INDONESIA

Abstraksi

Jaringan Diskless merupakan jaringan Client/Server, yaitu client di boot menggunakan disket yang sudah terpasang kernel. Setelah boot, ia akan mencari server yang memiliki boot protocol (BOOTP atau DHCP). Server yang menangkap permintaan tersebut akan melihat terlebih dahulu apakah komputer client tersebut merupakan salah satu komputer yang boleh masuk. Proses ini dilakukan dengan memeriksa database server apakah ada informasi unik komputer client sudah terdaftar, informasi tersebut adalah nomor hardware pada Ethernet card yang biasa disebut dengan MAC addres. Jika ya, server akan memberikan IP addres, Netmask, Gateway, hostname, dan informasi lainnya. Selanjutnya, komputer client melakukan pengenalan perangkat keras komputer client dan akses root filesystem dilanjutkan exsekusi inittab yang menetukan tingkat run level penggunaan.

Kata kunci: *Diskless, Kernel, Compile, NFS*

I. Pendahuluan

Perkembangan teknologi komputasi berkembang dengan sangat pesat. Hal ini dibuktikan dengan lahirnya Personal Computer (PC) canggih generasi pentium IV dengan kecepatan mencapai 2.4 GHz ke atas dan media penyimpanan yang semakin besar kapasitasnya. Perkembangan ini diikuti pula dengan perkembangan perangkat lunak sehingga seolah-olah perangkat lunak tersebut dirancang untuk PC Pentium tersebut. Fenomena yang ada pada saat ini banyak orang yang ingin mendayagunakan komputernya meskipun kecepatan dan media penyimpanan minimum agar masih dapat dipergunakan dengan baik pada sebuah jaringan, baik untuk mengakses informasi lokal maupun informasi global, salah satu pendayagunaan komputer tersebut dapat diimplementasikan dengan menghubungkan komputer tersebut kedalam sebuah jaringan, jaringan yang dimaksud adalah jaringan tanpa harddisk (*Diskless*) berbasis Linux

Masalah yang menjadi dasar pertanyaan adalah bagaimana proses jaringan *diskless* yang diterapkan berjalan dengan baik sedangkan komputer *client* yang terhubung ke *server* tidak dilengkapi dengan media penyimpanan dan tidak memiliki sebuah sistem operasi yang biasa diproses dalam *harddisk*, untuk menyelesaikan persoalan ini, memerlukan suatu media untuk melakukan *boot* di komputer *client*, media tersebut adalah dengan menggunakan sebuah kernel, namun apakah kernel tersebut telah sesuai dengan perangkat keras komputer *client*, protocol dan prosedur *diskless* oleh karena itu kernel perlu dikonfigurasi dan di-*compile* ulang.

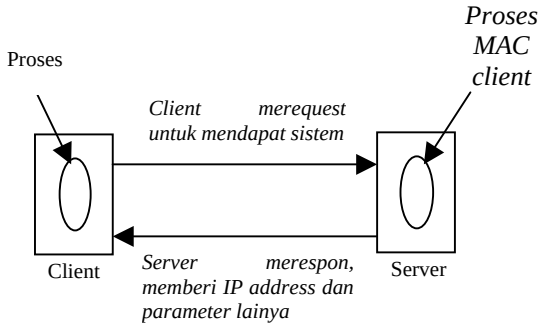
II. Metodologi

Dalam menyelesaikan penelitian skripsi ini penulis mengadakan penelitian di jurusan Teknik Informatika, Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia (UNIKOM) Jl. Dipatiukur No.112. Proses penelitian skripsi ini memerlukan waktu sekitar empat bulan terhitung mulai dari tanggal 1 Maret sampai dengan tanggal 1 Juli 2004.

Metode yang dilakukan untuk menyelesaikan laporan skripsi ini penulis menggunakan metode: Observasi/pengamatan langsung secara uji coba di lapangan, Wawancara melakukan tanya jawab kepada pihak yang menguasai dan memahami tentang masalah yang penulis teliti. Studi literature mencari sumber referensi baik dari buku, document atau internet yang berkaitan dengan masalah penelitian skripsi yang penulis kemukakan.

III. Pembahasan dan Hasil

Pengertian *Diskless* adalah memungkinkan *client* yang tidak dilengkapi dengan media penyimpanan, seperti *Harddisk* untuk dapat mengaktifkan sistem operasi dalam hal ini adalah Linux. Proses *diskless* akan membantu komputer *Client* untuk dapat mengaktifkan sistem operasi tersebut berupa kernel system yang dieksekusi di sisi komputer *client* dengan exsport NFS dari server. Setelah proses *diskless* selesai, dilanjutkan exsekusi inittab yang menetukan tingkat *run level* penggunaan.



Gambar 1 Model jaringan *Client/Server*

Dalam proses penelitian penulis menggunakan beberapa file yang mendukung tahapan penelitian, file–file tersebut merupakan paket file konfigurasi untuk komputer *server* dan komputer *client* file-file tersebut dapat ditemui pada CD 1, CD 2 dan CD 3 distro linux

```
rpm –ivh kernel_source.2.4.18-6mdk.i586.rpm
rpm –ivh tftp-server-0.29-2mdk.i586.rpm
rpm –ivh dhcp-common–3.0.1rc8.1mdk.i586.rpm
rpm –ivh dhcp-server-3.0-rc8.mdk.i586.rpm
rpm –ivh ltsp-core-3.0.4-0.i386.rpm
rpm –ivh nfs-utils-0.3.3-3mdk
rpm –ivh nfs-utils-client-0.3.3-3mdk
```

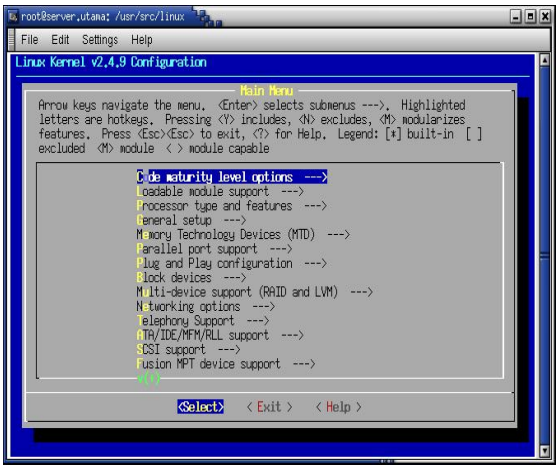
yang menandai bahwa kernel tersebut sudah dalam keadaan stabil atau masih dalam keadaan uji coba. Sebagai contoh, kernel versi 2.2.17, nomor pertama adalah *major number*, nomor dua adalah *minor nomor (patch level)* sedangkan nomor terakhir disebut sebagai *sub-level* dalam hal ini merupakan tambahan. Untuk kernel yang stabil nomor *minor* di tandai dengan bilangan genap seperti 2.2.xx dan 2.4.xx.

Penomoran versi kernel berada dalam file *makefile* pada direktori /linux yang dapat menentukan versi kernel sebelum mengkonfigurasi kernel. Pada bagian awal dari file tersebut terdapat variabel *version*, *patchlevel*, *sublevel.* Dalam perubahan *makefile* ini tidak ada ketentuan memasukan nomor versi misal versi kernel bisa tetap 2, patchlevel 4 menjadi 6 dan sublevelnya menjadi 9. Dalam hal ini tidak ada ketentuan khusus bila kernel tersebut untuk dikembangkan sendiri, tentunya hal tersebut akan sangat memudahkan untuk melakukan identifikasi kernel tersebut nantinya saat konfigurasi kernel. *Scrift makefile* sebagai berikut:

```
[root@SERVER linux]# vi makefile
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 18
KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SU
BLEVEL)
ARCH := $(shell uname -m | sed -e
s/i.86/i386/ -e s/sun4u/sparc64/ -e
s/arm.*/arm/ -e s/sa110/arm/)
KERNELPATH=kernel-$(shell echo
$(KERNELRELEASE) | sed -e "s/-//g")
CONFIG_SHELL := $(shell if [ -x "$$BASH" ];
then echo $$BASH; \
        else if [ -x /bin/bash ]; then
echo /bin/bash; \
        else echo sh; fi ; fi)
TOPDIR  := $(shell /bin/pwd)
HPATH          = $(TOPDIR)/include
FINDHPATH      = $(HPATH)/asm
```

Dalam tahap mengkonfigurasi kernel memiliki 3 cara diantaranya :

1. *make config*, akan menjalankan program konfigurasi kernel berbasis teks
2. *make menuconfig*, menjalankan program konfigurasi kernel menggunakan antarmuka menu
3. *make xconfig* menjalankan program konfigurasi kernel dengan antarmuka berbasis grafis



Gambar 2 Konfigurasi kernel menggunakan *menuconfig*

Konfigurasi kernel memiliki beberapa option pilihan setting diantaranya seperti di perlihatkan pada table berikut

Tabel Option konfigurasi kernel

Setting	Nilai	Penjelasaan
Y	Ya	Jika kita memilih *option* ini. Kita langsung membuat setting tersebut *built-in* ke dalam sistem
M	Module	Jika kita memilih menggunakan *module*, *option* pada kernel diaktifkan sebagai *module*
N	No	Pilihan ini digunakan, jika *option* pada kernel tidak akan diaktifkan sama sekali
?	Help	Pilihan ini akan memberikan penjelasan/petunjuk (help)

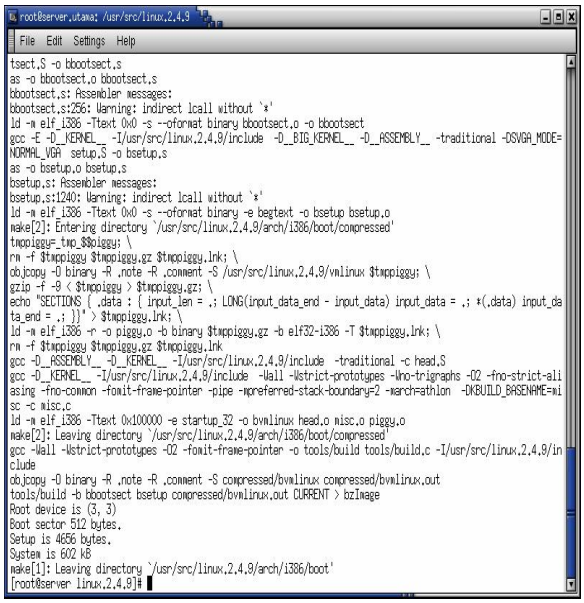
Pada tahap penelitian ini penulis mengkonfigurasi kernel untuk keperluan *diskless* dan keperluan komputer *client* maka hal yang dikonfigurasi pada kernel adalah sebagai berikut

- Loadable Module support (Enable load modul, kernel module loader)
- Processor type and feature (Athlon/duron/K7, math emulation)
- General setup (Network support, PCI support, system V IPC)
- Plug and play configuration (Plug and play support)
- Block device (Normal floppy, Ram disk, initial RAM disk support)
- Network option (Unix domain, TCP/IP, kernel autoconfig, DHCP)
- Network device (Ethernet 10 or 100 Mbit, Realtek 8139)
- Character device (virtual terminal, system console, unix 89 pty support)
- File system (/proc, second extended fs, NFS file support, root file system)
- Console driver (VGA text driver)

Kernel yang telah dikonfigurasi dan telah disesuaikan dengan dukungan perangkat keras komputer *client* dan jaringan tanpa *harddisk* masih belum dapat berfungsi sebagai *boot system* karena perlu di-*compile* terlebih dahulu, proses *compile* ini memerlukan beberapa waktu yang cukup lama sekitar delapan sampai sepuluh menit. Perintah sederhana untuk melakukan *compile* kernel adalah sebagai berikut

```
[root@SERVER linux]# echo "Mulai: $(date)"
> waktu; make dep && make clean && make
bzImage && make modules; echo "Selesai:
$(date)" >> waktu
```

- *make dep* : membuat depedensi file yang terkait sebelumnya
- *make clean* : memeriksa konfigurasi yang konflik antara kernel lama dan kernel baru setelah dikonfigurasi
- *make bzImage :* merupakan perintah compile kernel image
- *make modules* : mengkonfigurasi kernel dalam option module



Gambar 3 Hasil *compile* kernel

Konfigurasi Protocol Diskless

***Dynamic Host Configuration Protocol* (DHCP)**

Implemntasi kernel pada jaringan diskless perlu tahapan konfigurasi beberapa protocol yang terkait jaringan diskless diantaranya adalah mengkonfigurasi protocol DHCP, komputer *client* membutuhkan IP *adders* yang akan digunakan oleh dirinya untuk melakukan akses dan komunikasi dalam jaringan, untuk itu *client* harus mendapatkan IP *addres* dari komputer *server.*

Selain IP *address* yang diberikan, komputer *client* juga mendapatkan beberapa informasi, informasi yang didapatkan oleh kompter *client* diantaranya adalah *subnetmask*, *broadcast*, *hostname*, IP *address server*, MAC *address* LAN *card* komputer *client*, bergunakan untuk menetukan *client* yang diperbolehkan untuk akses *diskless* terhadap komputer *server*, part root dan lokasi kernel, *scrift* DHCP server adalah sebagai berikut:

```
[root@server etc]# vi dhcpd.conf
# once you adjusted this file and copied it
to /etc/dhcpd.conf.
ddns-update-style             none;
default-lease-time            21600;
max-lease-time                21600;

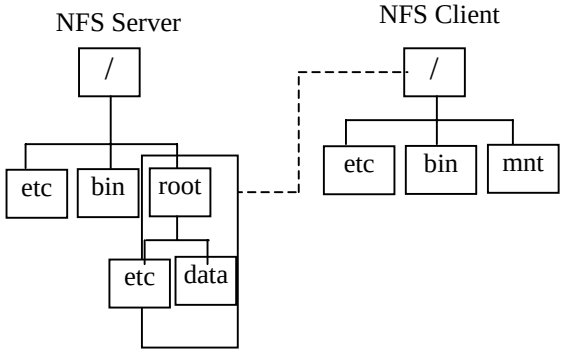
option subnet-mask            255.255.255.0;
option broadcast-address      192.168.0.255;
option routers                192.168.0.254;
option domain-name-servers    192.168.0.254;
option root-path
"192.168.0.254:/tftpboot/diskless/root";

option option-128 code 128 = string;
option option-129 code 129 = text;

shared-network WORKSTATIONS {
    subnet 192.168.0.0 netmask 255.255.255.0
{
    }
}
group   {
    use-host-decl-names       on;
    option log-servers        192.168.0.254;
  host unikom1 {
    hardware ethernet     00:90:47:01:9e:d6;
    fixed-address         192.168.0.1;
    filename      "/diskless/bzImage.2.4.9";
    }
}
```

***Network File System* (NFS)**

Network File System (NFS) adalah sebuah protocol yang dipakai untuk mengoperasikan sebuah komputer *server* agar dapat bekerja sebagai penyedia file atau mendistribusikan sumberdaya kepada komputer *client* sehingga komputer *client* dapat menjalankan isi file sistem operasi yang di-*export*,. File yang di-*exports* diinisialisasi sebagai *root* direktori yang nantinya akan di-*mount* dan berjalan pada komputer *client* yang disimpan pada memori.



Gambar 4 *Network File Sistem*

```
[root@server etc]# vi exports
#
/tftpboot/diskless/root
192.168.0.0/255.255.255.0(ro,no_root_squash)
/home
192.168.0.254/255.255.255.0(rw,no_root_squash)
```

Konfigursi *Trivial File Transfer Protocol* (TFTP)

Trivial File Transfer Protocol adalah sebuah protocol sederhana dengan menggunakan *User Datagram Protocol* (UDP) yang bekerja secara blok perblok untuk transfer file antar komputer yang sama maupun berbeda jaringan tanpa memerlukan pengenalan (*authentication*) terhadap user yang memakai. TFTP di gunakan untuk menghubungkan komputer *client* yang tidak memiliki media penyimpanan yaitu *harddisk* ke komputer *server* untuk *mendownload system operasi* untuk *booting* dengan menggunakn alamat IP yang ditentukan oleh dhcpd komputer *server*. File ditransfer hanya dengan menunjukan nama file, beberapa *Administrator* tidak mengaktifkannya atau membatasi jenis file yang dapat ditransfer. Untuk mengaktifkannya memerlukan perubahan pada file script tftp pada direktori */etc/xinet.d*

```
[root@server xinetd]# vi tftp
service tftp
{
        disable       = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server          = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
}
```

Yang harus dirubah scrift protocol tftp ini adalah *line* atau baris disable yang tadinya yes menjadi no yang menandakan protocol tftp dapat digunakan atau diakses untuk men-*downlod* kernel image pada komputer *server*

```
[root@server init.d]# service xinetd restart
Stopping xinetd:                [  OK  ]
Starting xinetd:                [  OK  ]
[root@server init.d]#
```

Kemudian lakukan pemetaan dan pembukaan *port* yang akan digunakan dengan mengetikan perintah *portmap* sebagi berikut

```
[root@server init.d]# service portmap start
Starting portmapper:              [  OK  ]
[root@server init.d]#
```

Portmap digunakan oleh *Network File System* dan dipakai untuk mengambil kernel *system diskless* pada saat proses *booting*, berikut *routine port contro*, *port* yang dibuka seperti berikut.

```
[root@server root]# rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp   1026  mountd
    100005    1   tcp   1026  mountd
    100005    2   udp   1026  mountd
    100005    2   tcp   1026  mountd
    100005    3   udp   1026  mountd
    100005    3   tcp   1026  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
```

Nilai port number untuk modul protocol TCP dan UDP pada system linux secara keseluruhan dapat dilihat pada file : */etc/services* pada system operasi linux

Inittab Run Level

File yang akan memproses tingkat run level pada konfigurasi *lts.conf* adalah sebuah file inittab. Isi script file inittab ini yang menetukan proses berjalannya file lts.conf dengan menentukan tingkat run level kemudian menjalankan program yang ada pada direktori */etc/rc* berdasarkan *run level*, isi file inittab ini terdiri dari beberapa baris hanya satu baris yang dieksekusi sebagai penentu tingkat run level dengan format sebagai berikut

id:runlevels:action:process

id adalah unik character nomor untuk memproses run level, runlevel adalah daftar tingkatan run level yang akan diexsekusi, action merupakan bagian proses. misal respawn: merupakan proses restart bila proses pertama telah gagal. Once : proses akan mengexsekusi satu kali berdasarkan run level yang dimasukan

Run level	Penjelasan
0	Halt
1	Singel user mode
2	*Multi user*, Tidak ada NFS
3	Full multiuser mode
4	*Unused* (tidak digunakan) untuk *run level* sendiri
5	X11 (Grafik window)
6	Reboot

***Root device* Kernel**

Root device pada komputer umumnya mengarah ke *harddisk* yang memiliki *filesystem* misalnya *harddisk* partisi 3 yang menjadi *root filesystem* adalah : */dev/hda3*. Pada penelitian ini *root device filesystem* komputer server pada *harddisk* partisi 3 sehingga perlu perubahan informasi *root* pada kernel baru yang telah di-*compile.*

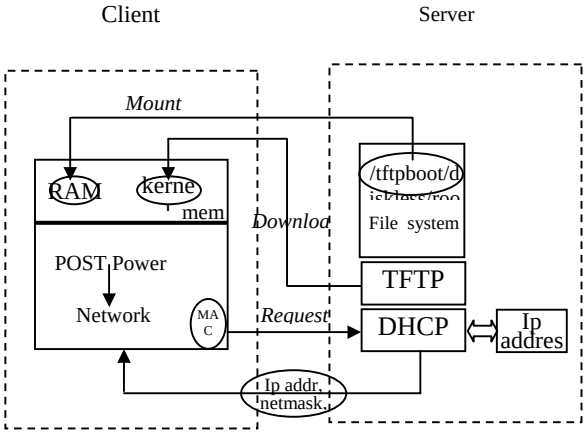
Sehingga untuk komputer *diskless* yang tidak dilengkapi dengan media penyimpanan yaitu *harddisk* maka dimungkinkan pirantinya mengarah ke *network file system* (NFS) yang memiliki *root* yang telah di *export* dan didisbrusikan oleh komputer server. Berikut ini merupakan perintah untuk mengosongkan *root device* di kernel baru untuk mengarah ke *network File System* dan *root* yang akan digunakan adalah *root* NFS pada kernel yang telah dikonfigurasi pada modus *built in.* untuk membuat parameter *root* tersebut perlu dibuat *dummy device* pada system linux dengan *block device* 0 dan *major number* 255

```
[root@SERVER linux]# mknod /dev/nfs b 0 255
```

Setelah mengosongkan *root* kernel di perlukan perubahan informasi *root device* di dalam kernel yaitu *root* yang baru dibuat pada direktori *root nfs* yang akan di *exsport* dengan perintah:

```
[root@SERVER linux]# rdev
/tftpboot/diskless/bzImage.2.4.9 /dev/nfs
```

root device disini sangat diperlukan untuk mengaktifkan NFS *device* namun bukan benar-benar *device* yang real, ini hanya *synonym* untuk kernel mengetahui NFS disaat *boot* untuk *boot network*.



Gambar 5 Ilustrasi Proses boot Diskless

1. Komputer *client booting* dengan *floppy disk* yang sudah diisi kernel *image* kemudian melakukan *network boot*
2. Kode *boot* mencari *Card LAN* dan melakukan pembacaan terhadap alamat fisik MAC addres 00:90:47:01:9E:D6 *Network Interface Card*
3. *Card LAN* me-*request* DHCP kepada komputer *server* kemudian komputer *server* menangkap permintaan *client* dan mencari IP *addres* dari *databases* berdasarkan MAC *addres* yang di ajukan
4. Komputer *server* mengirim balasan yang berisi informasi IP *addres client* 192.168.0.1 netmask 255.255.255.0, lokasi kernel system yang akan di *download* dan melakukan konfigurai TCP/IP pada *card LAN* yang diterima
5. Dengan menggunakan protocol TFTP komputer *client* men-*download* kernel system dari komputer *server* yang akan disimpan dalam memori
6. Kernel yang berada dalam memori melakukan inisialisasi perangkat keras komputer *client* yang ditemukan

7. Komputer *client* melakukan mount terhadap *Network file system* melaului NFS *protocol* pada direktori */tftpboot/diskless/root*
8. Pada saat *file system* berada pada ramdisk, dikontrol dan dimount pada direktori */tmp* oleh komputer *server*, pada direktori ini *server* dapat me-*mount* file ke dalamnya sebagai file *shere*
9. init kemudian membaca *file system /sbin/init*
10. file inittab ini berisi tingkat *run level* sehingga proses yang berjalan tergantung tingkatan bila *run level* 3 maka komputer *client* akan masuk pada *modus teks*

pada runlevel 3 maka pada komputer *client* akan tampil *bash prompt* menandakan *diskless* proses telah berhasil. Pada *run level* 3 ini diharapkan komputer *client* dapat mengakses komputer *server* dengan modus teks untuk langkah awal menemui kesalahan sehingga untuk kedepannya adalah tahap pengembangan yang lebih sempurna yaitu dengan menggunakan x86config.

IV. Kesimpulan

- Salah satu upaya pendayagunaan komputer yang memiliki kecepatan dan media penyimpanan minimum yaitu dengan menghubungkannya kedalam jaringan tanpa *harddisk* linux.
- Proses *diskless* ini memerlukan media untuk *boot* pada komputer *client* tanpa *harddisk*, media tersebut diantaranya adalah kernel. Kernel yang ada perlu tahapan pengkonfigurasian yaitu penyesuaian perangkat keras, protocol dan prosedur *diskless* kemudian meng-*compile* ulang.
- Tahapan implementasi merupakan upaya mengimplementasikan kernel yang telah dibuat perlu juga tahapan penyesuaian beberapa protocol pada komputer server seperti file protocol: inisialisasi root, NFS, TFTP, DHCPD, initialize, exports dan initttab.
- Kemudian memasukan kernel ke dalam disket, bila komputer server sudah siap kemudian memboot komputer *client* dengan *boot floppy* sehingga client dapat mengaktifkan *system operasi* berdasarkan tingkat *run level* yang telah ditentukan yang menandakan *diskless* proses telah berhasil.

Daftar Pustaka

1. **Abdul Kadir, fetra syahbana**, " *Sistem Operasi UNIX* " elex media komputindo Jakarta 1995
2. **Budi susanto**, "*Linux untuk praktikum Administrasi Jaringan* " Gava Media Jogyakarta 2003
3. **Budhi Irawan**, " *Pengatar Jaringan Komputer*". Universitas Komputer Indonesia 2002.
4. **Mulyana Y B**, "*Linux Semudah Windows"* Elexmedia Komputindo Jakarta 2002
5. **R.Kresno Aji**, " *Optimasi PC Tua menggunakan diskless system* " elex media komputindo Jakarta 2002
6. **Ricardus indrajit, bambang prastowo**, " *Desain dan implementasi Sistem Operasi Linux*" elexmedia komputindo jakarta 2002
7. **Ricardus indrajit, Hasan lamatungga**, " *Membangun jaringan diskless berbasis linux*" elexmedia komputindo jakarta 2002
8. **Ricardus indrajit, bambang prastowo, dudy rudianto**, " *Administrasi sistem linux redhat*" elexmedia komputindo jakarta 2002
9. www.linuxdoc.org/howto
10. www.justlinux.com
11. www.linux.org