

# Membangun Sendiri Distro Minimalis

Artikel ini menyajikan cara untuk membuat distro Linux seminimal mungkin. Minimal di sini berarti ukuran dan kemampuan distro sangat minim. Namun, distro ini dapat menjadi dasar bagi Anda yang berminat untuk mengembangkan *embedded distro* (misalnya distro untuk router).

Untuk membuat distro yang ukurannya sangat kecil ini, Anda dapat menggunakan partisi kosong harddisk Anda, atau harddisk komputer lama yang akan dibuat menjadi sistem *embedded* ataupun USB Flash Card Anda. PERINGATAN: jangan buat distro ini di partisi Linux Anda yang terpakai.

Apapun media yang Anda pilih, pastikan motherboard Anda *men-support booting* dari media tersebut. Sedikit catatan tambahan, floppy disk mungkin tidak bisa digunakan (karena ukuran terkecil distro yang pernah penulis buat adalah 1,7 MB).

Untuk membuat distro ini, penulis menggunakan dua buah komputer. Komputer pertama (*host*) digunakan untuk *men-download* source code, *meng-compile* dan menyusun distro. Komputer kedua (*target*) digunakan sebagai komputer aplikasi dan testing. Komputer target inilah yang nantinya berperan sebagai sistem *embedded*, setelah distro selesai dibangun. Komputer host penulis adalah AMD Athlon 1800XP dengan Mandrake 9.2, sedangkan untuk komputer target menggunakan Cyrix C3 800 MHz. Model *host-target* ini memudahkan penulis untuk melakukan *compiling* dan testing distro.

Untuk media distro, penulis menggunakan SanDisk CardFlash 128 MB. CardFlash yang penulis gunakan dihubungkan ke komputer target dengan IDE-CF adapter. Dengan demikian, komputer target akan menganggap CardFlash ini sebagai harddisk (*/dev/hda*).

Untuk menulis ke CardFlash, penulis menggunakan komputer host yang dilengkapi dengan 6-in-1 card reader adapter. 6-in-1 card reader di komputer host terhubung di SCSI pertama (*/dev/sda*).

Sebelum proses pembuatan distro dimulai terlebih dahulu kita pelajari proses booting yang terjadi di Linux.

## Proses Booting Linux BIOS (Basic Input Output System)

Ketika komputer dinyalakan, komputer akan memanggil BIOS dari flash ROM motherboard. BIOS berisi program (*routine*) untuk menginisialisasi komputer dan menyediakan *interface software-hardware* melalui *basic interrupt*.

Inisialisasi yang kali pertama dilakukan BIOS adalah POST (*Power On Self Test*). POST bertujuan untuk memastikan agar semua devais komputer berjalan dengan semestinya. Jika hasil POST baik, BIOS akan menjalankan rutin bootstrap loader.

Bootstrap loader bertugas untuk *me-load* boot sector yang bootable ke alamat memory 0x7C00. Bootable boot sector adalah boot sector pertama devais yang ditandai dengan angka khusus 0xAA55 di byte ke-510 (0x1FE, dua byte terakhir dari sektor). Selain rutin untuk meload boot sector, bootstrap loader juga berisi tabel devais yang diperkirakan mempunyai boot sector. Oleh karena itu, bootstrap loader menentukan jenis devais mana yang dapat di-booting di komputer Anda.

## Boot Loader

*Bootable boot sector* dapat berisi kernel image ataupun boot loader. Jika kernel image ditemukan di boot sector, kernel tersebut akan ditaruh di alamat 0x7C00 dan dijalankan. Proses booting akan berlanjut dengan inisialisasi kernel.

Jika boot loader yang ditemukan, boot loader akan meload tabel sistem operasi/

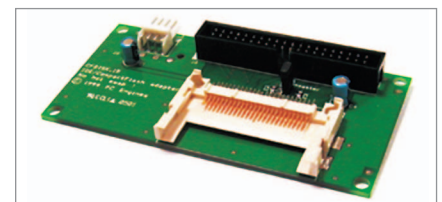
kernel yang dapat digunakan untuk booting. Setelah itu, boot loader umumnya akan menanyakan pengguna sistem operasi (*kernel*) mana yang dikehendaki untuk dijalankan. Kernel sistem operasi yang dipilih oleh pengguna kemudian di-load ke memory.

## Inisialisasi kernel Inisialisasi default

Fungsi utama kernel adalah manajemen sumber daya komputer. Pada kernel Linux, setelah kernel di-load ke memory, kernel akan menginisialisasi driver (yang di-*compile built-in* dalam kernel) dan struktur data internalnya (contohnya struktur data virtual memory). Kemudian, kernel akan mencari dan memount root filesystem dengan hak akses *read-only*. Root filesystem berisi program-program utama sistem. Program */sbin/init* (atau program lain yang didefinisikan di parameter kernel *init=namaProgram*) di root filesystem kemudian dijalankan oleh kernel.

## Inisialisasi Alternatif

Driver *built-in* kernel diperlukan agar kernel dapat mengakses hardware. Pada saat inisialisasi kernel, driver yang dibutuhkan kernel adalah driver jenis filesystem (contoh: ext2, ext3, *raisersfs*, *vfat*, dan seterusnya) dan driver untuk interface hardware (contoh: SCSI, IDE/ATA, USB). Jika jenis root filesystem



Gambar 1. IDE-CF Adapter.

adalah filesistem yang drivernya tidak built-in dalam kernel ataupun root filesistem terletak di media hardware yang driver-nya tidak built-in dalam kernel, kernel akan *hang* dan memunculkan pesan error: kernel panic : VFS : Unable to mount root fs at XX:YY. Untuk mengatasinya, driver filesistem dan interface hardware untuk root filesistem harus di-compile built-in kernel.

Cara lainnya adalah dengan membuat sebuah root filesistem yang memuat semua driver yang dibutuhkan di dalam ramdisk. Ramdisk adalah RAM komputer yang diperlakukan seolah-olah sebuah filesistem oleh kernel. Kernel akan mengakses ramdisk melalui /dev/ramdisk.

Metode ini umum dipergunakan untuk driver-driver yang tidak dapat di-compile built-in kernel (hanya dapat dikompilasi sebagai modul—umumnya driver yang *proprietary* atau source code-nya tidak diberikan). Root filesistem ini dikenal dengan nama *initrd* dan dibuat dengan software tool *mkinitrd*. *Initrd* umumnya bertipe *cram fs* (atau dapat juga bertipe *ext2* atau *ext3* yang dikompresi dengan *gunzip*).

Jika *initrd* ditemukan, kernel akan memount *initrd* ke dalam RAM dengan hak akses *read-only*. Kernel kemudian memanfaatkan driver-driver yang terdapat di *initrd* untuk mencari dan memount root filesistem sesungguhnya. Kemudian, program /sbin/init (ataupun program lain yang didefinisikan di parameter kernel *init=namaProgram*) di root filesistem dijalankan oleh kernel.

### SysVinit (/sbin/init)

*Init* (/sbin/init) adalah program pertama yang secara default akan dijalankan oleh kernel. Tugas utama program ini adalah menginisialisasi sistem. Program *init* yang umum dipakai di UNIX/Linux adalah *SysVinit*. Selain *SysVinit*, banyak program lain yang menawarkan program *init*, contohnya *BusyBox*.

### Loading

Program *init*/*SysVinit* bergantung kepada fungsi-fungsi yang disediakan oleh library *glibc*. Langkah pertama yang dilakukan oleh program *init* adalah meminta kernel untuk meload library *glibc* ke memory. Library *glibc* umumnya terletak di /lib, dengan nama *libc-versi.so*

### /etc/inittab

Program *init* kemudian akan membaca file /etc/inittab. File ini berisi nama script yang harus dilaksanakan untuk inisialisasi sistem dan daftar nama script yang harus dilaksanakan untuk tiap runlevel. Script adalah file yang berisi kumpulan perintah yang akan dilaksanakan oleh shell. Runlevel adalah mode operasi Linux. Umumnya, terdapat enam runlevel Linux, yaitu:

- 0 : Halt
- 1 : Single-user
- 2 : Multi-user (Without NFS)
- 3 : Multi-user
- 4 : Unused
- 5 : Multi-user with graphical login
- 6 : Reboot

Script inisialisasi sistem umumnya berisi perintah untuk me-mount seluruh filesistem yang ada di sistem (dengan menggunakan data yang tersimpan di /etc/fstab). Root filesistem yang semula di-mount *read-only* oleh kernel akan di-mount lagi sesuai keterangan di /etc/fstab. Selain itu, script inisialisasi juga mengaktifkan berbagai program daemon. Contohnya adalah *klogd* (kernel log daemon, untuk mencatat pesan error kernel) dan *syslogd* (system log daemon, untuk mencatat pesan error sistem).

### Getty dan Login

Setelah menjalankan script inisialisasi dan berbagai script inisialisasi runlevel, *init* akan menjalankan program *getty* dengan mode *respawn*. Artinya, jika program ini mati, program ini akan dijalankan kembali oleh *init*. *Getty* digunakan untuk menampilkan pesan selamat datang di sistem (pesan yang ditampilkan berdasarkan file /etc/issue) dan mengambil input nama user dan password dari terminal. Input yang diterima *getty* kemudian diteruskan ke program *login*. *Login* kemudian memverifikasi nama user dan password. Jika valid, user akan masuk ke sistem dan menjalankan shell. Sampai di sini, Linux dapat digunakan secara normal.

## Pembuatan distro minimalis Software yang dibutuhkan

Untuk membuat distro minimalis ini, Anda membutuhkan source code *BusyBox* dan Linux kernel. *BusyBox* adalah paket yang dapat menyediakan semua kebutuhan dasar

sistem Unix/Linux dengan sebuah file executable. Perintah-perintah seperti *cat*, *echo*, *ls*, *chmod*, *su*, dan sebagainya dapat digunakan dengan melakukan soft link perintah tersebut ke *BusyBox*.

### Mempersiapkan media

#### Harddisk

Jika Anda telah memiliki sebuah harddisk atau partisi kosong, format dengan perintah berikut:

```
$ mke2fs /dev/hdb1 (sesuaikan dengan lokasi harddisk Anda)
```

#### Partisi Harddisk

Jika mempunyai banyak space kosong di harddisk, Anda dapat menggunakannya untuk partisi baru. Gunakan program *cfdisk* ataupun *fdisk* (atau *diskDrake* jika Anda menggunakan *Mandrake*). Berhati-hatilah dengan program-program partisi ini. Backup data Anda sebelum Anda melakukan partisi harddisk Anda.

Setelah Anda mempartisinya, format partisi baru tersebut:

```
$ mke2fs /dev/hda6 (sesuaikan dengan nama partisi baru Anda).
```

### USB Disk ataupun CardFlash

Umumnya jenis disk ini telah diformat dengan tipe VFAT (untuk DOS/Windows). Agar dapat digunakan untuk Linux, format ulang ke *ext2*:

```
$ mke2fs /dev/sda1 (sesuaikan dengan letak media Anda)
```

Bila ternyata USB disk / CardFlash belum diformat sama sekali (*mke2fs* akan menolak memformat USB disk / CardFlash ini), siapkan dulu USB disk/ CardFlash:

```
$ dd if=/dev/zero of=/dev/sda bs=1k count=ukuran_CardFlash_ Anda,
```

Contoh :

```
$ dd if=/dev/zero of=/dev/sda bs=1k count=128000
```

(128000 sebenarnya bukan angka yang tepat untuk 128MB, tetapi bukan masalah).

### Membuat struktur direktori

Linux mempunyai banyak direktori (/bin, /boot, /etc, /dev, dll). Tiap direktori ini

mempunyai fungsi tersendiri (contohnya direktori /etc digunakan untuk menyimpan file-file konfigurasi). Fungsi direktori ini diatur oleh kebijakan pembuat distro. Bila Anda ingin mengetahui bagaimana contoh kebijakan penggunaan direktori ini, silahkan lihat ke <http://www.debian.org/doc/debian-policy>.

Akan tetapi, struktur direktori linux / UNIX sebenarnya telah distandardisasi dengan FHS (File Hierarchy Structure). Kita akan mengikuti struktur FHS.

Mount media yang Anda gunakan:

```
$ mount -t ext2 /dev/sda1
/mnt/cf (sesuaikan dengan letak
media Anda)
```

Buat struktur direktori :

```
$ cd /mnt/cf
$ mkdir bin boot dev etc home
lib mnt root proc sbin tmp usr
var
$ mkdir var/lock var/log var/
run var/spool
$ mkdir usr/bin usr/include
usr/lib usr/local usr/sbin usr/
share usr/src
$ mkdir usr/share/man
$ mkdir usr/share/man/
man{1,2,3,4,5,6,7,8,9}
$ ln -s usr/share/man usr/man
```

### Menginstal BusyBox

Sebagaimana telah dijelaskan di atas, busybox adalah sebuah file executable yang dapat melakukan apa saja. Untuk melakukan berbagai fungsi yang berbeda, Anda hanya perlu mengubah parameter busybox atau meng-softlink-kan fungsi dengan BusyBox (contoh : `ln -s /bin/BusyBox /bin/lis`). Akan tetapi hal ini akan dilakukan secara otomatis dengan `make instal`.

Untuk menginstal busybox, Anda memerlukan source code-nya. Jika telah mendapatkannya, ekstrak source code tersebut.

```
$ tar -xvzf busybox-version.
tar.bz2 atau
$ tar -xvzf busybox-version.
tar.gz
```

Setelah itu, Anda harus menentukan fungsi apa yang diperlukan dalam distro Anda.

```
$ cd busybox-version
$ make menuconfig
```

Petunjuk penulis: semakin banyak fungsi yang Anda tambahkan, semakin besar ukuran busybox. Program-program wajib yang harus Anda compile adalah `init` (support reading an inittab file), `mount` (support for a real /etc/mntab), dan `shell` (`ash` / `hush` / `lash` / `msh`). Bila Anda memerlukan network, compile juga `ifconfig` dan `route` (serta `udhcpc --dhcp client--` ataupun `udhcpd --dhcp server--` untuk koneksi dhcp).

Setelah selesai menentukan fungsi yang Anda perlukan, compile busybox tersebut:

```
$ make
```

Setelah di-compile, instal dengan :

```
$ make PREFIX=/direktori/
media/anda install
```

Contoh :

```
$ make PREFIX=/mnt/cf install
```

### Menginstal Library untuk BusyBox

BusyBox, seperti layaknya program lain, membutuhkan rutin program tambahan yang didapat dari library (kecuali Anda mengcompile BusyBox secara static).

Untuk mengetahui library apa saja yang dipakai oleh BusyBox, ketikkan:

```
ldd /path/to/busybox, contoh :
$ ldd /mnt/cf/bin/busybox
libcrypt.so.1 => /lib/
libcrypt.so.1 (0x40023000)
libc.so.6 => /lib/i686/
libc.so.6 (0x40050000)
/lib/ld-linux.so.2 =>
/lib/ld-linux.so.2 (0x40000000)
```

Busybox akan mencari library yang diperlukan di /lib. Untuk itu, kopi semua library yang tertera di atas ke direktori /lib di media tempat distro Anda.

```
$ cp /lib/libcrypt.so.1 /mnt/
cf/lib
$ cp /lib/libc.so.6 /mnt/cf/lib
$ cp /lib/ld-linux.so.2 /mnt/
cf/lib
```

Sedikit catatan, di komputer penulis terdapat dua `libc.so.6` (sebenarnya hanya ada satu buah `libc.so.6`, yang satunya lagi merupakan soft link). Maksud `libc.so.6 => /lib/i686/libc.so.6` adalah busybox menggunakan `libc.so.6` yang terdapat di `/lib/i686/libc.so.6`. Namun, secara default `libc.so.6` akan dicari

di /lib, jadi Anda tidak perlu membuat direktori /lib/i686.

### Menginstal kernel

Untuk menginstal kernel, Anda membutuhkan source code kernel (bisa Anda dapatkan dari CD distro Anda).

Unpack / ekstrak source code kernel:

```
$ rpm -i kernel-version.rpm
```

atau

```
$ tar -xvzf kernel-version.
tar.bz2
```

atau

```
$ tar -xzvf kernel-version.
tar.gz
```

Atur konfigurasi kernel:

```
$ cd /path/ke/lokasi/kernel
$ make menuconfig
```

Sedikit catatan penulis, pastikan kernel Anda di-compile untuk arsitektur yang tepat (jangan compile ke i586 bila komputer Anda hanya 486). Pastikan juga modul interface di komputer target Anda (IDE / SCSI / USB) dan filesistem (`proc` dan `ext2`) ter-compile built-in kernel. Bila Anda menggunakan USB disk, pastikan juga driver untuk USB mass storage support tercompile built-in kernel.

Setelah selesai mengonfigurasi kernel, save dan keluar dari menuconfig, lalu ketik

```
$ make dep
$ make clean
$ make bzImage
```

Kernel hasil compile akan terdapat di `/path/to/kernel/source/arch/i386/boot/bzImage`. Copy kernel ini beserta file `System.map` (di `/path/to/kernel/source`) ke direktori /boot media.

```
$ cp arch/i386/boot/bzImage
/mnt/cf/boot/vmlinuz
$ cp System.map /mnt/cf/boot
```

Bila Anda mengkonfigurasi beberapa device driver sebagai modul, edit, dan save Makefile utama kernel.

```
$ vim Makefile
```

Lalu cari baris `export INSTALL_MOD_PATH`. Tambahkan `INSTALL_MOD_PATH=/tempat/media/Anda`, contoh:

```
export INSTALL_MOD_PATH=/mnt/
cf
```

Setelah itu, compile dan install modul

```
$ make modules
```

```
$ make modules_install
```

### Membuat file-file device

File device yang terletak di /dev adalah jenis file yang merepresentasikan hardware di linux. Beberapa file device yang Anda butuhkan adalah:

- /dev/console—merupakan device yang digunakan kernel untuk menampilkan output
- /dev/ttyx (x=0,1,2,dst)—virtual terminal, diperlukan shell untuk job control

Selain file device di atas, Anda harus menginstall beberapa file device untuk mengakses media Anda di host dan di target (/dev/sda, /dev/hdb, /dev/hdb5, dll.). Bila Anda ingin menggunakan dhcp, pastikan /dev/urandom juga ada di /dev media Anda.

Untuk membuat file device, anda membutuhkan mknod. Berikut contoh penggunaannya :

```
$ cd /mnt/cf/dev
$ ls -l /dev/hda
brw----- 1 root root
3, 0 Jan 1 1970 /dev/hda
$ mknod hda b 3 0 (lihat dari
output ls)
```

Atau dapat menggunakan cara mudah:

```
$ cp -dpR /dev/hda /mnt/cf/dev
```

### Menginstal Boot Loader

Boot loader yang digunakan dalam distro ini adalah LiLo (Linux Loader). Untuk menginstall lilo, diperlukan file lilo.conf yang terletak di /etc media. Berikut adalah isi lilo.conf penulis :

```
boot=/dev/sda #Tempat media
Anda terpasang di Host, pastikan
/dev/sda ada di /dev media Anda
disk=/dev/sda # Tempat media
Anda terpasang di Host
bios=0x80
default=bfeLinux
image=/boot/vmlinuz
```

```
label=bfeLinux
root=/dev/hda #Tempat media
Anda terpasang di target, pas
takan /dev/hda ada di /dev media
Anda
read-only
```

Setelah itu, install ke MBR media Anda dengan

```
$ lilo -r /mnt/cf -C etc/lilo.
conf
```

### Membuat file konfigurasi

#### /etc/inittab

Inittab diperlukan untuk mengarahkan init ke script inisialisasi yang kita butuhkan. Satu hal yang perlu diperhatikan, init versi busybox tidak mendukung multiple runlevel. Karena itu, kita tidak membutuhkan script inisialisasi untuk tiap level. Inittab yang digunakan penulis adalah sebagai berikut:

```
::sysinit:/etc/rc.sysinit
::respawn:/bin/ash
```

#### /etc/rc.sysinit

/etc/rc.sysinit akan digunakan sebagai script untuk menginisialisasi sistem (lihat inittab). Penulis menggunakan rc.sysinit berikut:

```
#!/bin/sh
# Echo harus dikompile di
busyBox agar perintah ini dapat
berfungsi
echo "Welcome to Minimum
Distro"
echo "Remount root filesistem
as read-write"
mount -n -o rw,remount /
echo "Memount semua filesistem
di /etc/fstab"
mount -a
echo "Mengaktifkan loopback"
/sbin/ifconfig lo 127.0.0.1 up

# Bagian ini optional. Khusus
bila Anda ingin mengaktifkan
network interface di mesin tar
get Anda
# Bila anda meggunakan stat
ic IP, ganti $IP_address dan
$IP_mask di bawah
/sbin/ifconfig eth0 inet
$IP_Address netmask $IP_Mask up
# Setting gateway
# ganti $gateway_address
```

```
dengan IP gateway Anda
/sbin/route add default gw
$gateway_address
# Setting DNS yang Anda pakai
# ganti $domain dengan domain
tempat komputer target berada
echo "search $domain" >> /etc/
resolv.conf
# ganti $DNS_IP dengan IP DNS
Anda
echo "nameserver $DNS_IP" >>
/etc/resolv.conf
# Bila anda menggunakan dhcp,
pastikan udhcpd terkompile dalam
busybox
# dan packet socket option
terkompile di kernel
# Uncomment bagian ini untuk
mengaktifkan dhcp
# /sbin/udhcpd -n -i eth0 -s
/etc/udhcp.script
```

**Tambahan:** /etc/udhcp.script didapat dari source busybox. Anda hanya perlu mengkopinya dan menambahkan hak akses executable.

```
$ cp /home/kunil/busybox/
examples/udhcp/simple.script
/mnt/cf/etc/udhcp.script
$ chmod +x /mnt/cf/etc/udhcp.
script
```

```
/etc/mstab atau /etc/fstab
```

Perintah mount akan melihat file /etc/mtab ataupun /etc/fstab, tergantung konfigurasi Anda di busybox. Bila Anda tidak mengonfigurasi ini, secara umum mount akan memount filesistem yang terdapat di /proc/mounts.

/etc/fstabminimum yang digunakan penulis:

```
/dev/hda / default 1 1
proc /proc proc 1 0
```

Banyak proses busybox yang bergantung pada /proc. Karena itu pastikan filesistem proc ter-compile di kernel Anda.

### Testing

Nah, selesailah sudah pembuatan distro minimalis. Silakan reboot komputer dan coba jalankan!

**Daniel Widyanto**