PEMBANGUNAN SISTEM GNU/LINUX MENGGUNAKAN METODE *LINUX FROM SCRATCH*

BUILD GNU/LINUX SYSTEM USING LINUX FROM SCRATCH METHOD

Irham Imami Harahap[1], Dr. Purba Daru Kusuma S.T., M.T.[2], Anton Siswo Raharjo Ansori S.T., M.T.[3]

[1,2,3]Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom

[1]**irhamharahap@student.telkomuniversity.ac.id**, [2]**purbodaru@gmail.com**, [3]**raharjotelu@gmail.com**

Abstrak

GNU/Linux merupakan sistem operasi yang mirip dengan *Unix* dan mengadopsi Linux sebagai kernelnya. Perkembangan GNU/Linux yang sangat pesat tidak terlepas dari filosofi *open source*. Salah satu metode dalam mengembangkan sistem linux adalah Linux From Scratch. Metode Linux From Scratch merupakan pengembangan distribusi Linux dengan cara manual. Metode ini mengajarkan bagaimana membangun sebuah sistem Linux dasar yang dapat dimodifikasi sesuai keinginan penggunanya. Beberapa optimisasi yang dilakukan diharapkan meningkatkan performa dan kegunaan dari sistem linux yang dibangun.

Pengujian dilakukan dengan membandingkan sistem LFS 8.4 sebelum dan sesudah dioptimisasi. Data dari hasil pengujian menunjukkan sistem LFS yang telah dioptimisasi memiliki keunggulan dalam performa makefile sebesar 59,73 % .Penggunaan media penyimpanan juga berkurang sekitar 2018 MB dan optimisasi pada mode "*Performance"* adalah 5 %.

Kata Kunci : GNU/Linux. sistem operasi, *Linux From Scratch*

Abstract

GNU / Linux is an operating system similar to Unix and Linux as the kernel. The rapid development of GNU / Linux cannot be separated from the philosophy of open source. One method in developing a linux system is Linux From Scratch. The Linux Method From the Beginning was a development of the Linux distribution manually. This method teaches how to create a basic Linux system that can be approved as desired by the user. Some optimizations made are expected to improve the performance and usability of the built Linux system.

The test is done by comparing the LFS 8.4 system with an optimized system. Data from the test results show that the optimized LFS system has an advantage in makefile performance of 59.73%. The use of storage media is also reduced by around 2018 MB and optimization in "Performance" mode is 5%.

Key Word : GNU/Linux, operating system, Linux From Scratch

1. Pendahuluan

Metode *Linux From Scratch* dikembangkan oleh Gerard Beekmans. Metode ini memungkinkan pengembangan distribusi Linux dari *script* dengan mengeksekusi paket – paket dasar dan konfigurasi yang benar – benar manual. Menurut Gerard Beekmans, LFS memungkinkan pengguna untuk melakukan optimisasi terhadap sistem Linux yang dikembangkan, sehingga hanya menggunakan *resources* yang sedikit dan keamanan yang lebih baik. Metode LFS juga melahirkan beberapa distro yang cukup menarik dan memiliki banyak pengguna seperti NuTyX, CRUX, dan AryaLinux.

Untuk mengukur optimisasi sistem dalam metode LFS dapat dilakukan serangkaian pengujian dan membandingkannya dengan sistem LFS tanpa optimisasi. Sebagai optimisasi yang dilakukan antara lain konfigurasi *Makefile*, *stripping*, manajemen paket dan mode "*Performance*".

2. Kebutuhan Pembangunan Sistem

Sebagai persiapan untuk membangun sistem linux membutuhkan perangkat keras dan perangkat lunak. Untuk spesifikasi yang digunakan sebagai pengujian ini adalah sebagai berikut :

No	*Hardware/Software*	Keterangan
1	Prosesor	AMD Fx-7500 2,5 GHz 4MB L2 Cache
2	Arsitektur	64 bit
3	RAM	8 GB DDR3
4	LAN Card	Gigabit Ethernet LAN
5	Keyboard	US type Keyboard
6	Grafis	AMD Radeon R7 Graphic
7	Penyimpanan	HDD 750GB 5400RPM
8	Sistem Host	Gentoo Live cd
9	Versi LFS	LFS 8.4

3. Pembangunan *Linux From Scratch*

Perancangan sistem GNU/Linux yang akan dibangun memiliki beberapa tahapan yaitu :

1. Persiapan *Host System*

   Pada metode LFS, penting untuk memulai dengan sebuah *host system.* Host system digunakan untuk menambahkan partisi, menyediakan *mount point*, menyediakan compiler, dan mengunduh paket – paket sistem yang dibutuhkan LFS. Pada penelitian ini *host system* yang digunakan adalah *live-cd* dari Gentoo.
2. Pembuatan Partisi

   Pembuatan partisi pada LFS memungkinkan sistem linux yang dibangun untuk tetap berada pada hardware yang digunakan. Partisi juga berfungsi sebagai penyimpanan untuk sistem. Pembuatan sistem linux LFS dibangun pada partisi */dev/sda*. Partisi yang ditambahkan untuk sistem LFS yaitu partisi bertipe *swap* sebesar 4 GB dan partisi utama bertipe *linux filesystem* sebesar 60 GB.
3. Penambahan *Environtment* $LFS

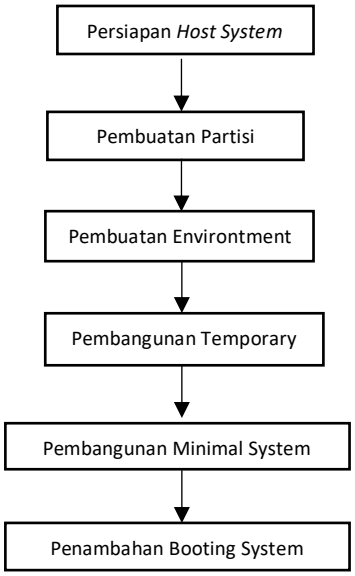
   Environtment $LFS dibuat untuk mendefinisikan direktori yang digunakan sebagai wadah pembuatan sistem LFS kedepannya. $LFS adalah sebuah direktori /lfs yang di *mount* pada direktori `/mnt` *host system*. Selain itu direktori `/tools` dan `/sources` juga ditambahkan pada *environtment* $LFS. Fungsi dari direktori `/tools` sebagai penyedia *temporary system* dan direktori `/sources` sebagai tempat seluruh paket – paket yang dibutuhkan proses pembangunan LFS. Sebuah group dan user lfs juga ditambahkan untuk dijadikan sebagai pemilik environtment $LFS ditahapan berikutnya.
4. Pembangunan *Temporary System*

   Setelah menambahkan *Environtment* $LFS, selanjutnya membangun sebuah *temporary system* (sistem sementara) pada direktori */mnt/lfs/tools. Temporary system* menyediakan *toolchain* untuk membangun sistem LFS yang sebenarnya. *Toolchain* yang dimaksud adalah *compiler, assembler, linker, libraries,* dan *utilities. Temporary system* beguna untuk menjaga kemurnian sistem minimal LFS yang akan dibangun dari konfigurasi *host system* karena proses pemasangan paket selanjutnya tidak menggunakan *tools* pada *host system.*
5. Pembangunan *Minimal System* LFS

   Untuk mulai membangun sistem akhir yang minimal dari LFS, kita harus melakukan ***`chroot`***. ***`Chroot`*** merupakan command yang menjadikan environment $LFS menjadi direktori `/root` menggantikan `/root` pada *host system.* Seluruh *temporary system* yang dibangun sebelumnya juga ditambahkan pada `/root` yang baru. Tahapan ini menjadikan lfs sebagai *superuser* yang memiliki kewenangan penuh terhadap `/root` yang baru. Kemudian saatnya membangun sebuah struktur direktori pada sistem yang baru. Struktur direktori pada LFS mengikuti model FHS. Pada tahapan ini seluruh paket berupa *compiler, assembler, linker, libraries*, dan *utilities* seperti *security, network* dan lainnya dipasangkan kedalam *environment $LFS* sesuai dengan FHS.
6. Penambahan *Booting System*

   Jika minimal sistem telah selesai, langkah berikutnya adalah melakukan

konfigurasi agar sistem *bootable* atau dapat melakukan proses *booting*. *Booting* adalah proses memasuki sistem operasi pada saat komputer pertama kali dihidupkan. LFS memiliki *bootscript* untuk mengontrol start dan stop pada sistem. Untuk bootloader sistem LFS menggunakan *GRUB* 2.0, kegunaan *bootloader* salah satunya adalah memuat kernel saat sistem dihidupkan. Pada tahapan ini beberapa konfigurasi juga dibutuhkan seperti konfigurasi kernel, konfigurasi *locale*, konfigurasi grub dan konfigurasi jaringan.



Gambar 3.2 Diagram Perancangan Sistem

4. Pengujian Optimisasi

1. Optimisasi *Makefile*

Optimisasi yang dilakukan dengan menambahkan *command* `export MAKEFLAGS= "-j$(nproc)".` Pengujian adalah waktu dari tiga kali percobaan dalam melakukan "`make`" saat pemasangan paket *Curl* pada sistem LFS sebelum dan sesudah optimisasi. Hasil pengujian adalah sebagai berikut.

Tabel 3 Optimisasi *Makefile*

NO	Sebelum Optimisasi	Sesudah Optimisasi
1	21,314 detik	9,807 detik
2	26,757 detik	10,227 detik
3	26,605 detik	10,039 detik
Rata[2]	24,892 detik	10,024 detik

Teroptimisasi : (24,892-10,024)/24,892 x 100% = 59,73%

Konfigurasi pada `MAKEFLAGS` berguna untuk memaralelkan proses "`make`" dan menyesuaikan dengan jumlah *cpu/core* yang terdapat pada *hardware*. Hasil pengujian menunjukkan waktu yang digunakan sistem setelah dioptimisasi lebih cepat sebesar 59,73 %. Maka dari itu dapat disimpulkan optimisasi yang dilakukan berhasil.

2. *Stripping*

Proses *stripping* yang disediakan dalam buku panduan LFS dapat mengurangi penggunaan *disk* sekitar 1,2 GB namun setelah pemasangan kernel jumlah penggunaan *disk* meningkat kembali menjadi 4,6 GB. Penggunaan *disk* dapat dikurangi dengan menghapus *file* `tools_lfs_8.4_linux.tar.xz` yang sebelumnya digunakan untuk *temporary*

system. Kemudian kita dapat menghapus direktori `/sources/` yang sebelumnya digunakan untuk menyimpan semua *file archive* dari paket yang akan dipasang pada sistem. Terakhir kita dapat menghapus direktori `/x86_64-lfs-linux-gnu/` yang sebelumnya juga digunakan untuk *temporary system* LFS. Penggunaan *disk* yang tercatat setelah *stripping* adalah sebagai berikut.

Tabel 4 Optimisasi *Stripping*

NO	Keterangan	Sebelum Stripping (MB)	Setelah Stripping (MB)
1	`rm -Rf /sources/`	4648	2941
2	`rm -Rf ../x86_64-lfs-linux-gnu/`	2941	2929
3	`rm ../tools_lfs_8.4_linux.tar.xz`	2929	2630
∑	4648 – 2630 = 2018 MB		

Optimisasi pemakaian disk : (2018/4648) x 100% = 43,42%

Dengan menghapus *file* dan direktori diatas, sistem LFS dapat dikurangi pemakaian *disk*nya sebanyak 2018 MB. Maka dari jumlah tersebut, pemakaian *disk* teroptimisasi sebesar 43,42% dibanding sistem LFS yang tidak melakukan *stripping* diatas.

3. Manajemen Paket

Sistem manajemen paket merupakan salah satu optimisasi yang dibutuhkan pada LFS. Secara umum kebutuhan manajemen paket adalah untuk memasang, melepas, dan memperbarui paket – paket dalam sistem operasi. Beberapa distro Linux menyediakan manajemen paket yang lebih fungsional seperti Apt, RPM, Pacman dan lain – lain. Sebagai bagian dari optimisasi, penguji memasang manajemen paket untuk sistem LFS yaitu Scratchpkg. Scratchpkg adalah sebuah manajemen paket yang dibuat khusus untuk sistem LFS. Sistem kerjanya menggunakan metode DESTDIR yang akan membangun dan memasang paket ke direktori sementara kemudian dikompresi kedalam bentuk *tar*. Kemudian paket dalam bentuk tar tersebut yang dipasangkan kedalam sistem utama. Scratchpkg juga otomatis menyediakan kebutuhan dependensi dari paket yang akan dipasang karena telah dikonfigurasi dan diindeks kedalam direktori khusus. Walaupun paket yang disediakan Scratchpkg tidak selengkap Apt ataupun RPM namun pengguna dapat mengatur sendiri manajemen paket tersebut seperti menambahkan paket dan dependensi yang dibutuhkan , mengubah *url* paket, dan beberapa fungsi lain sesuai kebutuhan. Scratchpkg merupakan proyek *open source* yang dapat diunduh dengan *link* berikut https://github.com/venomlinux/scratchpkg.git.

4. Mode *"Performance"*

Mode "*Performance*" yang dimaksud adalah subsistem dari Kernel Linux yang dapat mengatur penskalaan kinerja CPU yang terdiri dari tiga lapisan yaitu *core*, pengatur penskalaan (*governors*) dan penggerak penskalaan (*drivers*) menjadi paling maksimal pada frekuensi CPU yang digunakan. Mode ini akan menaikan voltase dari CPU dan mengonsumsi daya lebih besar dibanding dengan mode lainnya. Untuk masuk kedalam mode "*Performance*", perlu menu konfigurasi Kernel linux dan mengikuti panduan seperti berikut.

-> Power management and ACPI options

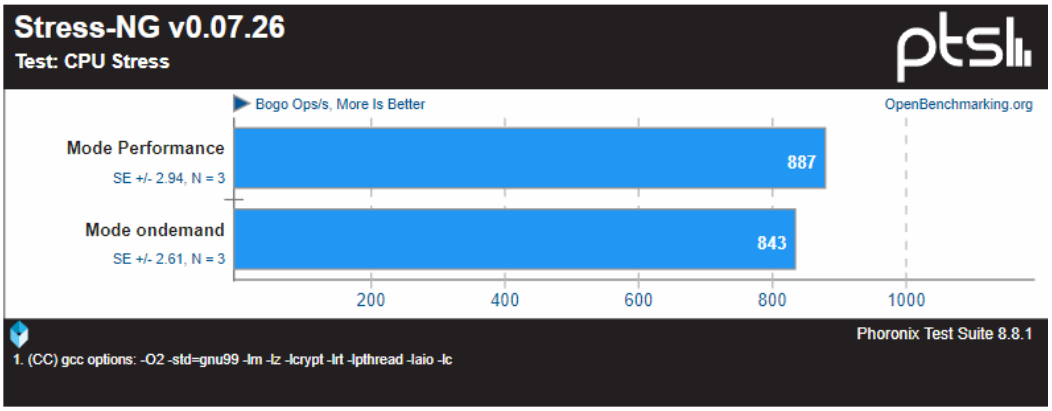
-> CPU Frequency scaling

-> CPU Frequency scaling (CPU_FREQ [=y])

-> Default CPUFreq governor (<choice> [=y])

Selects: CPU_FREQ_GOV_PERFORMANCE [=y]

Untuk menguji performa dari mode tersebut, dilakukan CPU stress test menggunakan Phoronix Test Suite. Pengujian dilakukan dengan membandingkan mode "*Performance*" dengan mode "*Ondemand*". Hasil pengujian adalah sebagai berikut.



Gambar 4.1 Pengujian *Stress-NG*

Berikut adalah tabel analisa hasil pengujian dari Phoronix Test Suite yang didapatkan.

Tabel 5 Hasil Pengujian Stress-NG

Hijau = Hasil pengujian yang terbaik Merah = Hasil pengujian yang terburuk **Hitam** = Hasil pengujian diantara keduanya	**Mode Performance**	**Mode Ondemand**
Test: CPU Stress (Bogo Ops/s ↑)	**887**	**843**
Normalisasi Nilai	100 %	95,04 %
Jumlah Sampel	3	3
Standar Deviasi	5,0971	4,5292
Standar Error	2,9328	2,5149

Pengaplikasian mode "*Performance*" menunjukkan bahwa performa CPU menjadi lebih baik dibandingkan dengan mode "*Ondemand*". Karena frekuensi CPU yang digunakan selalu di set menjadi maksimum maka mode "*Performance*" menunjukkan hasil lebih baik dibandingkan mode "*Ondemand*" yang dinamis sesuai beban proses. Optimisasi dengan mode "*Performance*" meningkatkan performa sekitar 5% lebih baik.

5. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, seluruh langkah, tahapan, serta proses dalam membangun sistem operasi Linux dengan metode LFS dapat diketahui secara jelas. Sistem operasi yang dihasilkan dapat digunakan dalam kehidupan sehari-hari. LFS mengajarkan bagaimana sistem operasi Linux dibangun. LFS memberikan pengetahuan dan pemahaman yang dalam tentang penggunaan sistem operasi Linux. Data dari hasil pengujian menunjukkan sistem LFS yang telah dioptimisasi memiliki keunggulan dalam performa makefile sebesar 59,73 % .Penggunaan media penyimpanan juga berkurang sekitar 2018 MB dan optimisasi pada mode "*Performance"* yaitu 5 %.

6. Daftar Pustaka

[1] Beekmans, G.. Linux From Scratch.http://www.linuxfromscratch.org/lfs/downloads/8.4/LFS-BOOK-8.4.pdf.

[2] Esteve, J.J. & Boldrito, R.S. GNU/Linux Advanced Administration. Eureca Media, SL (2009).[3] Hicks, A.. Slackware Linux Essentials. Slackware Linux, Inc. (2005).

[3] Masrurkhah, A. A., Danesh, A. S. & Taklimi, S. N. G. A Survey on Implementation of A Linuxbased Operating System Using LFS Method. International Journal of Computer Science Issues9, 170-174 (2012).

[4] Silberschatz, A., Galvin, P.B., & Gagne, G. Operating System Concepts Essentials. John Wiley & Sons Inc. (2011).

[5] Stallman, R. M.. Free Software, Free Society : Selected Essays of Richard M. Stallman. GNU Press (2002).