

Praktikum21

Manajemen Kernel

POKOK BAHASAN:

- ✓ Kernel
- ✓ Kompilasi Kernel
- ✓ Optimasi Kernel

TUJUAN BELAJAR:

- ✓ Mahasiswa dapat melakukan kompilasi kernel
- ✓ Mahasiswa dapat melakukan optimasi kernel

TUGAS PENDAHULUAN:

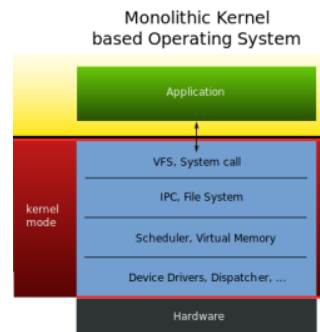
Siapkan linux anda di VMWare player yang hendak dikompilasi dengan space hard disk minimal 30GB, memory 2048, dan jumlah prosesor 4.

DASAR TEORI:

Konsep Kernel


Linux kernel pada dasarnya adalah monolithic kernel yang bersifat modular. Monolithic kernel adalah kernel yang semua bagiannya disimpan pada lokasi memory yang sama, sehingga mengurangi jumlah context switch dan messaging yang dibutuhkan. Akibatnya, proses booting dapat berjalan cepat dan optimal.

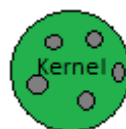
Pada monolithic kernel keseluruhan hardware, network dan file system terkompilasi dalam satu file image. Kelemahan dari monolithic kernel terletak besarnya jumlah code yang harus dilakukan di dalam kernel, sehingga sukar untuk di-debugging.



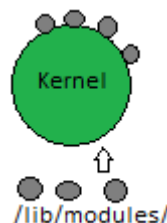
Linux kernel dikenal juga sebagai modular kernel karena beberapa drivernya dikompilasi sebagai file objek yang dapat dimuat dan dihapus oleh kernel sesuai dengan demand. Modul loadable ini disimpan dalam `/lib/modules`.

Pemisahan kernel dari modul ini sangat memudahkan developer untuk mendebug program, karena crash yang disebabkan oleh kernel atau modul lebih mudah dilokalisasi. Selain itu, kemudahan pengembangan modul menyebabkan menjamurnya jumlah device driver yang dikembangkan oleh para developer, yang mampu mendukung kernel Linux.

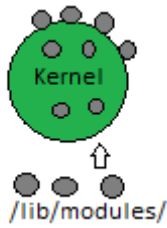
Mari kita lihat ilustrasi dibawah dimana Gambar a adalah monolithic kernel. Lingkaran abu-abu adalah modul  Modul. Kita dapat melihat bahwa modul dikompilasi didalam kernel. Akibatnya ukuran kernel menjadi besar dan berat, namun kernel ini lebih cepat.



Gambar b adalah modular kernel, dimana modul dikompilasi diluar kernel. Ukuran kernel menjadi kecil dan ringan, serta mudah didebug.



Linux menggabungkan kedua konsep diatas dalam kernel untuk mempercepat proses booting dan memudahkan debugging



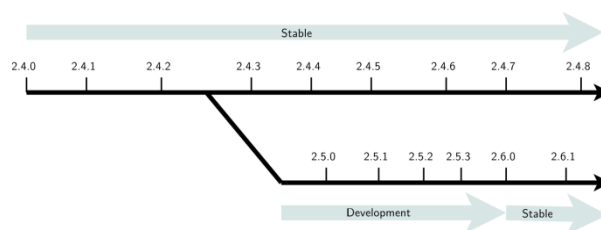
Versi Kernel Linux

Linux kernel dibagi menjadi dua versi:

- stable
- development

Stable kernel adalah release untuk level produksi dan cocok untuk dipasang di mana-mana. Versi stable kernel yang baru hanya akan direlease jika ada driver baru atau untuk menangani bug (bug-fix). Sebaliknya, versi baru dari development kernel akan terus berubah, karena developer bereksperimen dengan kernel baru, sehingga perubahan kernel yang drastis sering terjadi.

Kita dapat membedakan dua versi kernel ini dengan melihat versi minor release dari kernel. Minor release dengan angka genap adalah stable kernel, sementara minor release dengan angka ganjil adalah development kernel. Contoh : Versi kernel 2.6.0 adalah stable kernel, karena angka 6 yang menunjukkan versi minor adalah genap.



Development kernel, misal 2.5.0 berkembang bertahap menjadi 2.5.1 s.d 2.5.3. Perubahan ini terjadi karena developer menambahkan fitur baru atau menyelesaikan issue baru yang dihadapi oleh versi kernel tersebut. Selang waktu, kernel menjadi semakin mature dan siap untuk diluncurkan. Tidak ada fitur baru yang ditambah. Setelah kernel dianggap cukup stabil, dilakukan code freeze pada versi kernel tersebut. Kemudian, developer hanya boleh melakukan bug fix saja untuk kernel. Bila dianggap cukup, kernel direlease sebagai versi pertama stable kernel, yaitu 2.6.0.

Semenjak kernel 2.6.0, developer terus menambahkan fitur baru sedikit demi sedikit, tanpa harus mengubah sistem secara drastis. Dari tahun 2003 sampai 2011, versi kernel yang

dikembangkan adalah 2.6.x. Linux kernel versi 3.0 direlease juli 2011 dan Linux kernel 4.0 mulai direlease april 2015

Versi Kernel

Linux 2.6.32-5-amd64 x86_64

dimana ,

- Linux - Nama kernel
- 2.6.32-5- Versi Nomor Kernel
 - 2 = Major , 6 = Minor, 32 = Micro, 5= Level Patch
- Seringkali :
 - dua digit pertama, major dan minor menunjukkan versi kernel
 - dua digit terakhir, micro dan patch menunjukkan versi revisi
- amd64 - prosesor amd, 64 bit
- x86_64 - Nama hardware mesin (64 bit)

Kernel Source Tree

Kernel source tree adalah direktori yang menyimpan semua source kernel. Biasanya diletakkan di `/usr/src/kernel -<versi kernel>`. Anda dapat membangun (building) kernel baru, menginstall dan mereboot Linux untuk menggunakan kernel hasil building tersebut (kernel rebuilt). Biasanya, tujuan dari rebuilding adalah untuk mendapatkan kernel yang lebih ramping atau untuk menambah device driver baru yang tidak disertakan pada kernel Linux sebelumnya.

Kernel source tree dibagi dalam beberapa direktori, yang memiliki beberapa sub direktori untuk tiap direktorinya.

Directory	Description
arch	Architecture-specific source
crypto	Crypto API
Documentation	Kernel source documentation
drivers	Device drivers
fs	The VFS and the individual file systems
include	Kernel headers
init	Kernel boot and initialization
ipc	Interprocess communication code
kernel	Core subsystems, such as the scheduler
lib	Helper routines
mm	Memory management subsystem and the VM
net	Networking subsystem
scripts	Scripts used to build the kernel

security	Linux Security Module
sound	Sound subsystem
usr	Early user-space code (called initramfs)

Kompilasi dari Kernel Terkustomisasi (Customized Kernel)

Customized kernel adalah kernel yang telah dirombak sesuai dengan kebutuhan pemakai.


Keuntungan dari customized kernel adalah :

1. Mendukung sejumlah driver hardware yang lebih bervariasi, termasuk yang paling baru
2. Menghapus driver tidak terpakai dari kernel
3. Boot time yang lebih cepat karena ukuran kernel yang lebih kecil
4. Penggunaan memory yang lebih rendah
5. Sekuritas yang lebih baik karena ada driver/fitur/modul yang ditambahkan atau dihapus
6. Anda dapat memilih menggunakan kernel yang paling baru dengan segala keuntungannya.
7. Anda dapat belajar mengenai kernel dan pemanfaatannya.

Untuk melakukan perombakan ini, anda harus mendownload Linux kernel source tree secara penuh, melakukan instalasi dan kustomisasi kernel. Melakukan kompilasi kernel memiliki kelebihan dan kelemahan. Untuk mengubah kernel, anda harus melakukan kompilasi dan rebooting ke kernel baru. Kebanyakan fungsionalitas pada kernel linux tersimpan dalam modul yang dapat diloading dan dihapus dari kernel secara dinamis sesuai kebutuhan.


The Linux Kernel Archives

[About](#)
[Contact us](#)
[FAQ](#)
[Releases](#)
[Signatures](#)
[Site news](#)



Protocol
[HTTP](#)
[GIT](#)
[RSYNC](#)

Location
<https://www.kernel.org/pub/>
<https://git.kernel.org/>
<rsync://rsync.kernel.org/pub/>

Latest Stable Kernel:

4.10.11

mainline:	4.11-rc7	2017-04-16	[tar.xz]	[pgp]	[patch]	[view diff]	[browse]
stable:	4.10.11	2017-04-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.9.23	2017-04-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.4.62	2017-04-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.1.39	2017-03-13	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.18.49 [EOL]	2017-04-18	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.16.43	2017-04-04	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.12.73	2017-04-13	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.10.105	2017-02-10	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.4.113	2016-10-26	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	3.2.88	2017-04-04	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
linux-next:	next-20170418	2017-04-18					[browse]

Patch

Patch adalah dokumen teks berukuran kecil yang mengandung perubahan kecil diantara dua versi kernel yang berbeda dari satu source tree. Patch diciptakan lewat program diff. Proses menginstall patch pada kernel disebut patching. Agar patching dapat diaplikasikan, anda

perlu mengetahui base dari mana patch tersebut di-generate dan apa efek perubahan yang disebabkan oleh patch tersebut pada source tree. Informasi ini didapat dari file metadata patch atau lewat filename patch.

Kenapa Patching kernel ?

Mengapa anda membutuhkan patch ?

- Seringkali anda membutuhkan driver untuk hardware baru, namun sayangnya driver tersebut tidak didukung oleh kernel yang ada.
- Atau, misalkan anda membutuhkan hendak menerapkan teknologi virtualisasi terbaru namun ternyata teknologi tersebut belum didukung oleh kernel. Untuk itu anda memerlukan patch pada source kernel.

Untuk melakukan patching, anda harus mendownload patch yang dibutuhkan, melakukan ekstraksi dan meletakkan patch tersebut pada direktori /usr/src/linux.

Prepatching

Patches dapat juga dilakukan untuk kernel source. Misalkan anda membutuhkan fitur yang ada pada kernel 2.6.19-rc4 , namun kernel tersebut belum direlease, maka anda dapat mengaplikasikan patch-2.6.19-rc4.bz2 untuk kernel source 2.6.18. Teknik ini disebut sebagai pre-patching. Namun jangan lakukan prepatch untuk kernel 2.6.18.1 or 2.6.18.2, dsb. Ini dijelaskan pada <http://kernel.org/patch-types/pre.html>.

Patching Kernel 4.x

Kernel dengan versi ini termasuk release stabil. Kernel dengan versi tertinggi adalah kernel yang paling terupdate. Jika terjadi error atau ketidak stabilan, maka patch akan direlease untuk base kernel tersebut (4.x). Selain itu, jika ada kernel 4.x baru yang direlease, maka patch kernel baru tersebut dapat digunakan. Patch tersebut berisi perbedaan delta antara kernel sebelumnya dengan kernel baru

Untuk berpindah dari kernel 4.6 ke 4.7, anda tinggal melakukan patching dengan patch 4.7. Sedangkan jika anda hendak berpindah dari kernel 4.6.1 ke 4.7, anda harus melakukan konversi mundur ke base kernel 4.6.1, yaitu 4.6 dengan patch 4.6.1. Setelah itu anda dapat mengaplikasikan patch 4.7.

```
# moving from 4.6 to 4.7
$ cd ~/linux-4.6      # change to kernel source dir
$ patch -p1 < ../patch-4.7  # apply the 4.7 patch
$ cd ..
$ mv linux-4.6 linux-4.7  # rename source dir

# moving from 4.6.1 to 4.7
$ cd ~/linux-4.6.1    # change to kernel source dir
$ patch -p1 -R < ../patch-4.6.1 # revert the 4.6.1 patch
# source dir is now 4.6
$ patch -p1 < ../patch-4.7  # apply new 4.7 patch
$ cd ..
$ mv linux-4.6.1 linux-4.7  # rename source dir
```

Patching Kernel 4.x.y

Kernel dengan versi 3 digit adalah kernel stabil. Kernel tersebut berisi perubahan (fix) yang paling sedikit dan biasa diaplikasikan untuk mengatasi masalah sekuritas atau regresi pada kernel 4.x. Versi kernel ini direkomendasikan untuk user yang menginginkan kernel terbaru yang stabil dan tidak tertarik untuk melakukan eksperimen. Jika tidak ada versi kernel 4.x.y yang tersedia, maka, versi kernel 4.x tertinggi adalah yang paling stabil. Jika anda ingin menggunakan patch 4.7.3 ke kernel 4.7.2, maka anda harus melakukan konversi mundur ke base 4.7, kemudian baru mengaplikasikan patch 4.7.3

```
$ cd ~/linux-4.7.2      # change to the kernel source dir
$ patch -p1 -R < ../patch-4.7.2 # revert the 4.7.2 patch
$ patch -p1 < ../patch-4.7.3  # apply the new 4.7.3 patch
$ cd ..
$ mv linux-4.7.2 linux-4.7.3  # rename the kernel source dir
```

Patching Kernel -rc (release candidate)

Kernel -rc atau kernel release candidate adalah kernel experimental yang direlease oleh Linus. Kernel rc bukan kernel yang stabil, akibatnya anda harus siap menghadapi kemungkinan terjadinya crash atau error selama dijalankan.

Walaupun begitu, kernel rc adalah kernel paling stabil dari pengembangan kernel yang baru dan merupakan cikal bakal dari kernel baru yang stabil. Karena itu, penting bagi kernel -rc untuk di-testing oleh publik. Berikut ini contoh siklus development kernel 2.6.38

January 4	2.6.37 stable release
January 18	2.6.38-rc1, merge window closes
January 21	2.6.38-rc2
February 1	2.6.38-rc3
February 7	2.6.38-rc4
February 15	2.6.38-rc5
February 21	2.6.38-rc6
March 1	2.6.38-rc7
March 7	2.6.38-rc8
March 14	2.6.38 stable release

Kernel 4.8-rc5 berarti kernel tersebut adalah kandidat release ke 5 untuk calon kernel baru versi 4.8. Ketika patching dilakukan, maka patch pertama diterapkan pada kernel 4.7. Berikut ini diberikan contoh patching kernel-rc

```
# first an example of moving from 4.7 to 4.8-rc3
```

```
$ cd ~/linux-4.7          # change to the 4.7 source dir
$ patch -p1 < ../patch-4.8-rc3  # apply the 4.8-rc3 patch
$ cd ..
$ mv linux-4.7 linux-4.8-rc3    # rename the source dir
```

```
# now let's move from 4.8-rc3 to 4.8-rc5
```

```
$ cd ~/linux-4.8-rc3      # change to the 4.8-rc3 dir
$ patch -p1 -R < ../patch-4.8-rc3  # revert the 4.8-rc3 patch
$ patch -p1 < ../patch-4.8-rc5    # apply the new 4.8-rc5 patch
$ cd ..
$ mv linux-4.8-rc3 linux-4.8-rc5   # rename the source dir
```

```
# finally let's try and move from 4.7.3 to 4.8-rc5
```

```
$ cd ~/linux-4.7.3        # change to the kernel source dir
$ patch -p1 -R < ../patch-4.7.3    # revert the 4.7.3 patch
$ patch -p1 < ../patch-4.8-rc5    # apply new 4.8-rc5 patch
$ cd ..
$ mv linux-4.7.3 linux-4.8-rc5    # rename the kernel source dir
```


PERCOBAAN:

Percobaan 1. Melihat versi kernel

1. Untuk melihat keseluruhan informasi mengenai kernel linux anda

```
$ uname -a
```

2. Untuk melihat release kernel saja

```
$ uname -r
```

3. Untuk melihat versi kernel saja

```
$ uname -v
```

Percobaan 2. Melihat distribusi Linux

```
$ lsb_release -a  
#cat /etc/debian_version  
#cat /etc/*release
```

Percobaan 3. Kompilasi Kernel

1. Update repo anda

```
#apt-get update
```

2. Install beberapa software berikut :

```
# apt-get install fakeroot build-essential ncurses-dev libncurses5-dev xz-utils libssl-dev bc
```

3. Lihat versi kernel linux anda. Catat versi kernel linux anda

```
#uname -r
```

4. Lihat beberapa direktory berikut

```
#ls -al /boot  
#ls -al /lib/modules
```

5. Cek konfigurasi grub.

```
#cat /boot/grub/grub.cfg
```

Cari baris berikut : menuentry 'Debian GNU/Linux'-- class

Dibawahnya, catat versi dari vmlinuz. Linux /boot/vmlinuz-3.16.....

6. Download kernel baru yang hendak dikompilasi di <http://www.kernel.org>. Cari versi terakhir yang stabil. Download versi .tar.xz. Jika anda tidak terkoneksi ke internet, minta source kernel dari dosen anda

7. Simpan kernel source di /home/<user>/Downloads, kemudian lakukan un-tar pada file tersebut

```
#tar -xJf linux-4.10.10.tar.xz
```

8. Sekarang kita mulai melakukan konfigurasi kernel. Hal yang paling mudah yang untuk mengkonfigurasi adalah menggunakan konfigurasi kernel baru yang semirip mungkin dengan konfigurasi kernel yang kita punya.

```
#cd /home/<user>/Downloads/linux-4.10.10  
#cp /boot/config-$(uname -r) .config
```

9. Mulailah melakukan konfigurasi

```
# make oldconfig
```

10. Seringkali file .config ini tidak up-to-date, sehingga banyak fitur baru yang hendak ditambahkan ke kernel. Sebab itu, saat melakukan konfigurasi, anda akan ditanya mengenai pilihan-pilihan baru tersebut. Sementara ini anda cukup menekan tombol enter sampai selesai

11. File konfigurasi kernel yang baru telah terbentuk. Anda bisa melihat file tersebut

```
#less .config
```

12. Buka file Makefile pada /home/<user>/Downloads/linux-4.10.10

```
#cd /home/<user>/Downloads/linux-4.10.10  
#nano Makefile
```

13. Perhatikan baris paling atas berikut. Isi EXTRAVERSION = <nama_anda>. Jangan lupa simpan

```
VERSION = 4  
PATCHLEVEL = 10  
SUBLEVEL = 10  
EXTRAVERSION = <nama_anda>  
NAME = Fearless Coyote
```

14. Sekarang lakukan kompilasi. Kompilasi akan memakan waktu cukup lama (bisa 3 jam). Jika proses kompilasi selesai, anda akan mempunyai kernel baru beserta modul-modulnya. Opsi -j digunakan untuk menambah kecepatan kompilasi.

```
#make -j 4
```

15. Lakukan instalasi kernel beserta modul-modulnya. Opsi -j digunakan untuk menambah kecepatan kompilasi.

```
# make modules_install -j 4  
# make install -j 4
```

16. Langkah terakhir, begitu semua modul dan kernel telah terinstall, maka sekarang, kita akan menggunakan kernel baru saat booting awal

```
# update-initramfs -c -k 4.10.10<nama_anda>
```

Perintah diatas akan men-generate file /boot/initrd.img-4.10.10<nama-anda>. Jika tidak, coba kita reinstall kembali udev. Jika anda berhasil, skip perintah dibawah

```
#apt-get install --reinstall udev
```

17. Sekarang, lakukan update terhadap grub, untuk menambahkan kernel baru kita.

```
#update-grub
```

18. Restart system

19. Setelah login cek dengan perintah uname. Apakah versi kernel anda telah berganti ?

```
#uname -r
```

Percobaan 4. Update kernel tanpa melakukan kompilasi

1. Catat versi kernel Linux anda

```
$ uname -r
```

2. Masukkan baris berikut di /etc/apt/sources.list di abris paling bawah.

```
deb http://kebo.vlsm.org/debian jessie-backports main
```

3. Cari kernel image yang available

```
#apt-get update  
# apt-cache search linux-image
```

4. Instal kernel image

```
# apt-get install linux-image-x.x.x-xx
```

Pilihlah kernel-image yang anda inginkan. Perhatikan bahwa tidak semua kernel image sesuai dengan kernel linux lama. Biasanya instalasi akan gagal, jika kernel tersebut menimbulkan masalah pada kernel lama. Pada percobaan ini, coba install kernel-image-4.9.0-0.bpo.3-686-pae.

5. Berhasilkah ? Jika kernel anda memiliki dependensi dengan linux-base, maka coba lakukan hal berikut

```
# apt-get install linux-base -t jessie-backports  
# apt-get install linux-image-4.9.0-0.bpo.3-686-pae
```

6. Reboot OS anda

7. Pada saat layar GRUB muncul, klik opsi Debian GNU Linux seperti yang biasa anda lakukan. Secara otomatis kernel image akan terupdate sesuai dengan versi yang anda install.

8. Pilih kernel yang anda ingin booting. Pilih kernel yang baru saja anda install, yaitu kernel-image-4.9.0-0.bpo.3-686-pae.

9. Coba cek apakah kernel anda sudah berubah

```
#uname -r
```

10. Reboot kembali linux anda

11. Pada saat layar GRUB muncul, klik opsi Advanced options for Debian GNU/Linux'

12. Pilih kernel yang anda ingin gunakan dan bootinglah pada versi kernel lama sesuai hasil perintah 1.

13. Coba cek apakah kernel anda sudah berubah menjadi kernel yang lama.

```
#uname -r
```

Percobaan 5. Membuat File Patch

File patch menunjukkan perbedaan antara dua file yang berbeda. Program diff dapat digunakan untuk mengkomparasi file asal dan file baru dan menuliskan perbedaan tersebut berbentuk file, yang disebut sebagai file patch. Program/tool patch dapat mengaplikasikan file patch ke file asal, sehingga isi file lama akan sama persis dengan isi file baru.

1. Buat file bernama file_lama dengan menggunakan nano.

```
$ nano /home/student/file/file_lama
```

Ketikkan baris berikut

```
Ini
adalah
file
sederhana
```

2. Buatlah file bernama file_baru dengan menggunakan nano.

```
$ nano /home/student/file/file_baru
```

Ketikkan baris berikut

```
Ini
adalah
file
yang
lebih
kompleks
```

3. Sekarang, kita akan menggunakan program diff untuk membandingkan kedua file tersebut

```
$ cd /home/student/file
$ diff -uNr file_lama file_baru > patchfile
$ nano patchfile
```

Lihatlah tanda + dan - pada kata sederhana, yang, lebih dan kompleks. Tanda - menunjukkan bahwa kata sederhana tidak ada di file_baru. Sedang tanda + menunjukkan bahwa kata yang, lebih dan kompleks harus ditambahkan pada file_lama.

4. Sekarang kita akan melakukan patching file lama dengan file baru.

```
$ patch -p4 < /home/student/file/patchfile
```

Jika anda ditanya file mana yang akan dipatching, jawab saja file_lama

5. Sekarang coba anda lihat isi dari file_lama

```
$ nano /home/student/file/file_lama
```

Bandingkan dengan isi file_baru.

Apakah isi file_lama sama atau berbeda dengan isi file_baru ?

Percobaan 6. Patching Software

1. Download software openvpn terbaru dan versi sebelumnya. Sampai bulan mei 2017, software openvpn terbaru adalah openvpn -2.4.2, sedangkan versi sebelumnya adalah

openvpn-2.3.15. Mintalah pada instruktur jika anda tidak terkoneksi langsung dengan internet. Simpan pada directory /home/student/Downloads

2. Lakukan dekompresi paket untuk kedua software

```
$ cd /home/student/Downloads
$ mkdir backup
$ cp openvpn-2.4.1.tar.gz backup/
$ tar -xvzf openvpn-2.4.2.tar.gz
$ tar -xvzf openvpn-2.4.1.tar.gz
$ tar -xvzf openvpn-2.4.1.tar.gz
```

3. Buat direktory backup yang berisi file openvpn-2.4.1.tar.gz. Lakukan dekompresi.

```
$ mkdir backup
$ cp openvpn-2.4.1.tar.gz backup/
$ cd backup
$ tar -xvzf openvpn-2.4.1.tar.gz
```

4. Gunakan tool diff untuk membuat patch pada direktori /home/student/Downloads

```
$ cd ..
$ diff -Naur openvpn-2.4.1 openvpn-2.4.2 > openvpn.patch
```

5. Coba buka openvpn.patch

```
$ nano openvpn.patch
```

Perhatikan baris -baris awal dari openvpn.patch. File ini berisi perbedaan antara /openvpn-2.4.1 dan /openvpn-2.4.2.

```
diff -Naur /home/student/Downloads/openvpn-2.4.1/aclocal.m4 --dst--
--- /home/student/Downloads/openvpn-2.4.1/aclocal.m4 --dst--
+++ /home/student/Downloads/openvpn-2.4.1/aclocal.m4 --dst--
-# generated automatically by aclocal 1.14.1 -*- Autoconf -*-
+# generated automatically by aclocal 1.13.4 -*- Autoconf -*
```

Coba buka file aclocal.m4 pada openvpn-2.4.1 dan openvpn-2.4.2

```
$ cd /home/student/Downloads
#cat openvpn-2.4.1/aclocal.m4 | grep "generated automatically"
#cat openvpn-2.4.2/aclocal.m4 | grep "generated automatically"
```

6. Sekarang aplikasikan openvpn.patch agar openvpn-2.4.1 sama dengan openvpn-2.4.2. Mengapa p4? Karena direktory yang akan dipatch (openvpn-2.4.1) terletak pada direktori /home/student/Downloads/. Perhatikan bahwa ada 4 buah karakter / (slash) pada /home/student/Downloads/.

```
$ patch -p4 < /home/student/Downloads/openvpn.patch
```

7. Sekarang bandingkan isi file aclocal.m4 pada openvpn-2.4.1 dan openvpn-2.4.2

```
$ cd /home/student/Downloads
#cat openvpn-2.4.1/aclocal.m4 | grep "generated automatically"
#cat openvpn-2.4.2/aclocal.m4 | grep "generated automatically"
```

Sama atau tidak ? Jika patching berhasil, maka seharusnya outputnya akan sama

8. Sekarang kita membuat file patch dengan bantuan diff

```
$ diff -Naur openvpn-2.4.1 openvpn-2.4.2 > openvpn.patch2
```

Coba buka file openvpn.patch2

```
$ nano openvpn.patch2
```

Apakah isinya kosong ? Jika kosong berarti /openvpn-2.4.1 dan /openvpn-2.4.2 sudah sama. Jika ada isinya, berarti patching kita sebelumnya gagal, karena kedua direktory masih belum serupa.

9. Sekarang kita kembalikan lagi /openvpn-2.4.2 seperti semula dengan mereverse patching

```
$ patch -p4 -R < /home/student/Downloads/openvpn.patch
```

10. Sekarang bandingkan isi file aclocal.m4 pada openvpn-2.4.1 dan openvpn-2.4.2

```
$ cd /home/student/Downloads
#cat openvpn-2.4.1/aclocal.m4 | grep "generated automatically"
#cat openvpn-2.4.2/aclocal.m4 | grep "generated automatically"
```

Apakah isi file aclocal.m4 pada openvpn-2.4.1 dan openvpn-2.4 sama atau berbeda ? Seharusnya berbeda, karena ketika anda melakukan reverse patching dengan menggunakan opsi -R, berarti kita mengembalikan /openvpn-2.4.1 seperti awal.

Patching Kernel

1. Update repo anda

```
#apt-get update
```

2. Lihat versi kernel linux anda

```
#uname -v
```

3. Download patch yang hendak dikompilasi di <http://www.kernel.org>. Sesuaikan dengan versi kernel yang anda punya. Download versi .tar.xz. Jika anda tidak terkoneksi ke internet, minta patch dari dosen anda. Misalnya kernel yang baru kita gunakan adalah 3.16. Kita akan berpindah ke kernel 4.10.11. Maka kita membutuhkan patch

4.

```
#cd /usr/src/linux
#xz -d patch-4.10.6.xz
```

5.

```
#patch -p1 < patch-4.10.6.xz
```

Daftar Pustaka

1. https://www.howtoforge.com/kernel_compilation_ubuntu_p1&p2
2. <https://www.kernel.org/doc/Documentation/applying-patches.txt>
3. <http://kernel-handbook.alioth.debian.org/index.html>
4. <http://blog.zedroot.org/linux-kernel-debugging-using-kdump-and-crash/>
5. <http://free-electrons.com/doc/training/linux-kernel/>
6. <http://www.tutorialsdaddy.com/2015/09/linux-kernel-modules/>
7. <http://www.tutorialsdaddy.com/2015/07/writing-hello-world-driver/>
8. <https://medium.freecodecamp.com/building-and-installing-the-latest-linux-kernel-from-source-6d8df5345980>
9. <http://www.linux.org/threads/linux-kernel-reading-guide.5384/>
10. <http://www.brokenwire.net/bw/Various/120/fix-virtualbox-causes-unidentified-network-on-vista-and-windows-7>
11. <http://courses.linuxchix.org/kernel-hacking-2002/01-check-materials.html>
12. <http://free-electrons.com/doc/training/linux-kernel/linux-kernel-slides.pdf>
13. <https://www.kernel.org/doc/html/latest/process/applying-patches.html>

14. <https://raphaelhertzog.com/2012/08/08/how-to-use-quilt-to-manage-patches-in-debian-packages/>
15. <http://julio.meroh.net/2013/11/patch-management-with-quilt.html>
16. <https://debian-handbook.info/browse/stable/sect.kernel-compilation.html>
17. <https://01.org/linuxgraphics/gfx-docs/drm/process/2.Process.html>
18. <http://www.stevesdebianstuff.org/Kernel.htm#Update>
19. <https://courses.linuxchix.org/kernel-hacking-2002/11-creating-applying-and-submitting-patches.html>
20. <http://www.stevesdebianstuff.org/Kernel.htm>
21. <http://www.thegeekstuff.com/2014/12/patch-command-examples>