Praktikum 20

Makefile, Shared Library & Source Packages

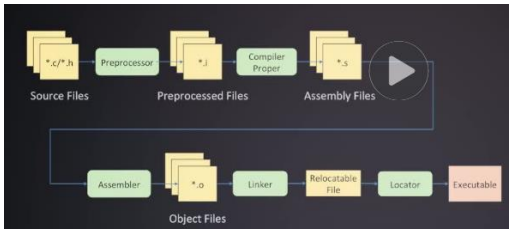
POKOK BAHASAN:

- ✓ Makefile
- ✓ Make

TUJUAN BELAJAR:

- ✓

DASAR TEORI:



Kompilasi Program C dan C++

Untuk melakukan kompilasi, anda dapat menggunakan sintax berikut :

```
cc program-source-code.c -o executable-file-name
```

atau

```
gcc program-source-code.c -o executable-file-name
```

atau

```
make executable-file-name
```

Opsi-opsi penting dari GCC

Opsi	Arti
-Wall	Memberi tahu compiler untuk menggunakan opsi Warning pada seluruh bagian code. Warning sangat berguna untuk melakukan debugging
-ansi	Memberi tahu compiler untuk menggunakan opsi bahasa ANSI. Akibatnya, beberapa fitur GCC yang tidak kompatibel dengan standar ANSI tidak bisa digunakan
-pedantic	Biasa digunakan bersama opsi -ansi, tujuannya agar compiler berpegang teguh pada standar ansi, semua code yang tidak kompatibel dengan ansi akan ditolak
-I {directory_name}	Memberi tahu linker lokasi include yang digunakan
-L {directory_name}	Memberi tahu linker lokasi library yang digunakan
-l {library}	Memberi tahu linker nama library yang digunakan dan menyuruh compiler mencari library tersebut di lokasi standar yang biasa digunakan untuk library dan lokasi yang ditunjukkan oleh -L
-o{file_name}	Memberi tahu compiler untuk menyimpan program hasil kompilasi dalam nama tertentu sesuai nama file yang diberikan setelah opsi tersebut
-c {file_name}	Memberi tahu compiler bahwa nama file sesudah opsi tersebut adalah file source code yang harus dicompile, namun jangan di-link

Instalasi package source

Jika anda pernah melakukan instalasi program dari source, anda pasti mengingat 3 perintah utama yang harus anda gunakan

```
$ ./configure
$ make
$ make install
```

Apa yang dilakukan 3 perintah tersebut ?

1. Configure : Konfigurasi software

   Skrip configure bertugas untuk menyiapkan sistem untuk melakukan building software. Dengan menggunakan skrip tersebut, semua dependensi yang dibutuhkan (serta bagaimana cara menggunakan dependensi tersebut) untuk proses building dan instalasi akan disiapkan. Program Unix seringkali ditulis dalam bahasa C, karena itu dibutuhkan compiler C untuk melakukan building. Pada kasus diatas, maka sistem akan mengecek ketersediaan kompiler C dan dimana letaknya. Setelah seluruhnya OK, maka configure akan mencari file Makefile.in dan mengubahnya menjadi file Makefile

2. Make : Melakukan building software

Begitu perintah configure dijalankan, anda dapat menggunakan perintah make untuk melakukan building (kompilasi) software.

3. Make Install : Menginstall software

   Ketika software telah dibuilding dan siap untuk di-running, file-file hasil kompilasi dapat dikopikan ke destinasi akhir. Perintah make install akan menyalin ke direktory di PATH anda, manual page akan disalin ke direktory MANPATH demikian juga untuk file-file lain, disimpan di tempat yang telah ditentukan.

Suite Autotools

Skrip configure dibuat lewat kumpulan program (atau disebut suite) autotools, yang terdiri dari autoconf, automake dan beberapa program. Tugas dari auto tools adalah membuat file-file : configure, Makefile.in, aclocal.m4, dan lain-lain. Dengan menggunakan suite autotools, user tidak harus menuliskan file-file tersebut secara manual. Dengan cara ini, apapun versi distro Linux anda, proses installasi program source akan tetap sama.

Membuat skript configure.ac

Skrip configure memang digenerate dari Autotools, namun untuk membuat file tersebut, Autotols membutuhkan file configure.ac. File tersebut ditulis dengan format m4sh, yaitu kombinasi antara makro m4 dan skrip shell POSIX. Makro tersebut bertugas menjelaskan apa yang harus dilakukan oleh file configure.

1. Langkah pertama untuk membuat skrip configure.ac adalah menginisialisasi autoconf dan melakukan setting informasi tentang program yang hendak dibuat packagingnya. Hal ini dilakukan lewat makro AC_INIT. Formatnya :

   AC_INIT (*package, version,* [*bug-report*], [*tarname*], [*url*])

   Misalkan, nama package kita bernama helloworld, versi 0.1 dan dimaintain oleh fitri@pens.ac.id. Kedua hal ini dilakukan dengan perintah AC_INIT.

```
AC_INIT([helloworld], [0.1], [fitri@pens.ac.id])
```

2. Kemudian, kita harus menginisialisasi penggunaan automake dengan makro AM_INIT_AUTOMAKE

```
AM_INIT_AUTOMAKE
```

3. Kemudian, autoconf disetting agar mengecek kompiler C. Untuk itu, kita menggunakan makro AC_PROG_CC.

```
AC_PROG_CC
```

4. Setelah langkah diatas, kita akan menggunakan makro AC_CONFIG_FILES untuk memberitahu autoconf bahwa skrip configure bertugas menemukan file makefile.in, menggantikan tag @PACKAGE_VERSION@ dengan nilai 0.1 dan menuliskannya ke Makefile

```
AC_CONFIG_FILES([Makefile])
```

5. Terakhir, dengan makro AC_OUTPUT, kita memberitahu autoconf bahwa seluruh informasi telah diberikan dan saatnya untuk mengoutputkan skrip

```
AC_OUTPUT
```

Setelah makro terakhir tesebut, skrip configure akan menghasilkan file makefile.in sepanjang 4737 baris.

Membuat skrip Makefile.am

1. Berbagai fitur automake ditentukan oleh options pada makro AUTOMAKE_OPTIONS . Pertama kita perlu memberitahu automake bahwa kita tidak menggunakan standar layout dari proyek GNU, namun menggunakan layout proyek foreign

```
AUTOMAKE_OPTIONS = foreign
```

2. Berikutnya untuk melakukan build terhadap program, Automake harus diberitahu nama binary, nama source file (file .c) dan nama librarynya. Untuk memberitahu nama file binary yang dihasilkan, digunakan sintax PROGRAM. Pada percobaan ini, nama file binary adalah helloworld

```
bin_PROGRAMS = helloworld
```

3. Setelah, mendeklarasikan PROGRAMS, berikutnya kita perlu memberitahu automake nama source file. Karena nama source file kita adalah main.c, maka :

```
helloworld_SOURCES = main.c
```

make all	Build programs, libraries, documentation, etc. (same as make).
make install	Install what needs to be installed, copying the files from the package's tree to system-wide directories.
make dist	Recreate package-version.tar.gz from all the source files.
make installcheck	Check the installed programs or libraries, if supported.

make check	Run the test suite, if any.
make distclean	Additionally erase anything ./configure created.
make clean	Erase from the build tree the files built by make all.
make install-strip	Same as make install, then strip debugging symbols. Some users like to trade space for useful bug reports..
make uninstall	The opposite of make install: erase the installed files. (This needs to be run from the same build tree that was installed.)

PERCOBAAN:

Percobaan 1 : Kompilasi file C dan C++

1. Update linux anda dan install paket build-essential dan manpages-dev

```
$ sudo apt-get update
$ sudo apt-get install build-essential manpages-dev
```

2. Untuk mengecek dimana binary gcc yang bertugas melakukan kompilasi dan dimana

```
$ whereis gcc
$ which gcc
$ gcc --version
```

3. Tulis program dibawah, simpan dengan nama demo.c

```
#include<stdio.h>
/* demo.c:  My first C program on a Linux */
int main(void)
{
 printf("Hello! This is a test program.\n");
 return 0
}
```

4. Untuk melakukan kompilasi/building, bisa dipilih satu dari tiga perintah berikut

```
$ cc demo.c -o demo
```

atau

```
$ gcc demo.c -o demo
```

atau

```
$ make demo
```

5. Untuk melakukan debugging, tambahkan opsi -Wall

```
$ gcc -Wall demo.c -o demo
```

6. Cobalah untuk melakukan eksekusi

```
$ ./demo
```

7. Tulis program dibawah, simpan dengan nama demo2.c

```
#include "iostream"
// demo2.C - Sample C++ prgoram
int main(void)
{
   std::cout << "Hello! This is a C++ program.\n";
   return 0;
}
```

8. Untuk melakukan kompilasi/building, bisa dipilih satu dari dua perintah berikut

```
g++ demo2.c -o demo2
```

atau

```
make demo2
```

Percobaan 2: Membuat shared Library

1. Buatlah file main.c , letakkan di directory /home/<user>/Documents/shared

```
#include <stdio.h>
#include "test.h"

int main(void)
{
    printf("This is a shared library test...\n");
    test();
    return 0;
}
```

2. Buatlah file test.c , letakkan di directory /home/<user>/Documents/shared

```
#include <stdio.h>

void test(void)
{
   printf("Hello, I'm a shared library \n");
}
```

3. Buatlah file test.h , letakkan di directory /home/<user>/Documents/shared

```
#ifndef test_h__
#define test_h__

extern void test(void);

#endif  // test_h__
```

4. Sekarang, lakukan kompilasi file test.c

```
$ gcc -c -Wall -Werror -fPIC test.c
$ ls -al
```

5. Sekarang, buatlah file shared library. File shared library akan terbentuk dengan mana libtest

```
$ gcc -shared -fPIC -o libtest.so test.o
$ ls -al
```

6. Lakukan linking dengan shared library. Binary yang diouptkan gcc bernama : program

```
$ gcc -Wall -o program main.c -ltest
$ ls -al
```

Terjadi error karena, karena tidak dapat menemukan file library

7. Beritahu gcc dimana library berada

```
$ gcc -L /home/<user>/Documents/shared/ -Wall -o program main.c -ltest
$ ls -al
```

8. Berhasil !!! File program telah terbentuk.

```
$ cd /home/<user>/Documents/shared/
$ ls -al
```

9. Coba di-running

```
./program
```

10. Gagal ? Apa error yang diberikan ? Jika gagal lanjutkan langkah dibawah
11. Loader tidak dapat menemukan shared library. Hal ini disebabkan lokasi shared library tidak berada di lokasi standar, yaitu : /lib, /usr/lib dan /usr/local/lib. Untuk itu, kita dapat menggunakan environment variable LD_LIBRARY_PATH. PErtama mari kita lihat isi variabel tersebut

```
$ echo $LD_LIBRARY_PATH
```

12. Kosong ataukah ada isinya? Jika kosong, masukkan path library seperti berikut. Kemudian cek isinya.

```
$ LD_LIBRARY_PATH='/home/username/shared'
$ echo $LD_LIBRARY_PATH
```

Jika ada isinya, masukkan path library dengan cara berikut. Dengan cara ini, kita tidak menghapuskan path library sebelumnya

```
$ LD_LIBRARY_PATH='/home/username/shared':$LD_LIBRARY_PATH
$ echo $LD_LIBRARY_PATH
```

13. Lakukan eksekusi

```
$ ./program
```

14. Berhasilkah ? Pasti gagal, karena kita belum mengeksport LD_LIBRARY_PATH. Sekarang export LD_LIBRARY_PATH

```
$ export LD_LIBRARY_PATH
$ echo $LD_LIBRARY_PATH
```

15. Lakukan eksekusi

```
$ ./program
```

Percobaan 3 : Kompilasi Source Packages & mengatasi dependensi

1. Download software pidgin dari https://www.pidgin.im/download/linux/ atau minta ke instruktur
2. Install beberapa program berikut

```
#apt-get install build-essential graphviz checkinstall
```

3. File yang diperoleh adalah pidgin-2.12.0.tar.bz2 . Lakukan un-tar pada file tersebut. Perintah ini akan melakukan un-kompresi terhadap file bz2. Direktory baru akan terbentuk

```
#tar -jxvf pidgin-2.12.0.tar.bz2
```

4. Masuk ke direktory tersebut dan lakukan konfigurasi. Perintah ini akan mengecek hardware dari sistem dimana software akan diinstall. Kemudian mengecek dependensi dari paket. Jika ada yang kurang, maka proses ini akan gagal. Jika berhasil, akan terbentuk Makefile.

```
#cd pidgin-2.12.0
#./configure --disable-plugins
```

5. Banyak terjadi masalah dependensi. Menginstall software satu-persatu bukan solusi. Gunakan perintah berikut

```
# apt-get build-dep pidgin
```

6. Lakukan konfigur kembali

```
#./configure --disable-plugins
```

7. Berhasilkah ? jika iya, akan terbentuk Makefile pada lokasi yang sama. Make akan melakukan kompilasi terhadap paket pidgin untuk disiapkan sebeluin diinstal. proses ini disebut juga dengan building packages

```
#make
```

8. Jika berhasil, lanjutkan dengan make install. Perintah ini akan menginstal pidgin ke sistem

```
#make install
```

9. Coba kita jalankan pidgin

```
#pidgin
```

Percobaan 4 : Membuat file untuk Configure, Make , Make Install

1. Buatlah file bernama main.c dan ketikkan baris berikut. Kemudian simpan di direktory /home/<user>/Documents/hello

```
#include <stdio.h>

int main(int argc, char* argv[])
{
   printf("Hello world\n");
   return 0;
}
```

2. Buatlah file configure.ac dan ketikkan baris berikut. Ganti fitri@pens.ac.id dengan alamat email anda. Nama package : helloworld, versi:0.1, dan alamat email programmer fitri@pens.ac.id. Kemudian simpan di direktory /home/<user>/Documents/hello

```
AC_INIT([helloworld], [0.1], [fitri@pens.ac.id])
AM_INIT_AUTOMAKE
AC_PROG_CC
```

```
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

3. Buatlah file Makefile.am dan ketikkan baris berikut.

```
AUTOMAKE_OPTIONS = foreign
bin_PROGRAMS = helloworld
helloworld_SOURCES = main.c
```

4. Langkah pertama yang kita lakukan adalah menggunakan program aclocal untuk men-generate aclocal.m4 dan autom4te.cache

```
#aclocal
```

5. Sekarang, mulailah men-generate file configure dari file configure.ac dengan menggunakan program autoconf. Perintah ini akan men-generate file configure.

```
#autoconf
```

6. Kemudian mulailah men-generate file Makefile.in dengan menggunakan perintah automake. Opsi --add-missing digunakan untuk mencegah error karena ada file yang kurang. Dengan menggunakan opsi ini, automake akan membuat symbolic link ke dirinya sendiri untuk file yang hilang/tidak ada. Perintah ini akan men-generate file Makefile.in dari file Makefile.am

```
#automake --add-missing
```

7. Berikutnya, kita melakukan configure, untuk mendeteksi kebutuhan program , termasuk diantaranya build-environment, gcc, executable file, dependensi, dan lain-lain. Perintah ini akan men-generate beberapa program, yaitu config.log, config.status, direktori .deps dan yang paling penting : file Makefile (dari file Makefile.in)

```
# ./configure
```

8. File yang dibutuhkan untuk package helloworld telah lengkap. Berikutnya kita akan membuat file tar.gz dari package tersebut. Selain membuat package tar.gz, perintah tersebut juga men-generate file : compile, depcomp, install-sh dan missing.

```
#make dist
```

9. Sekarang, tinggal melakukan cek terhadap package .tar.gz agar dapat diinstall dalam berbagai kondisi. Berhasilkah ?

```
make distcheck
```

Jika tidak, catat error. Error ditandai oleh baris yang mengandung kata ERROR di bagian bawah output.

10. Sekarang, kita harus melakukan rekonfigurasi karena terjadi error saat melakukan make distcheck. Autoreconf akan menjalankan autoconf, autoheader, aclocal, automake, libtoolize dan autopoint untuk mengupdate hasil build.

```
#autoreconf -ivf
```

11. Sekarang kita lakukan make distcheck lagi

```
#make distcheck
```

Ada error ? Harusnya tidak, karena autoreconf telah menangani error sebelumnya

12. Selamat, anda berhasil membuat package tar.gz baru !!!
13. Sekarang copykan package helloworld-0.1.tar.gz tersebut ke direktory /home/<user>/Downloads, kemudian cobalah mengekstrak package tersebut dan cobalah melakukan kompilasi

```
#cp  helloworld-0.1.tar.gz     /home/<user>/Downloads/
#cd /home/<user>/Downloads
#tar -xvzf helloworld-0.1.tar.gz
```

14. Langkah pertama adalah ./configure

```
# ./configure
```

15. Langkah kedua adalah make

```
#make
```

16. Langkah ketiga dan terakhir adalah make install

```
#make install
```

Apakah terjadi error? Jika iya, belum tentu ini error, coba cek letak file binary di /usr/local/bin. Apakah anda dapat melihat file helloworld ? Jika iya, coba kita eksekusi

```
#cd /usr/local/bin
#ls -al | grep helloworld
#./ helloworld
```

17. Berhasilkah ?

Percobaan 5: Uninstall Sourge Package

Setelah melakukan instalasi source binary helloworld, berikutnya kita akan melakukan uninstall

1. Cara yang paling mudah adalah melakukan make uninstall

```
$cd /home/<user>Downloads/helloworld-0.1
$make uninstall
```

Terlihat bahwa yang dikerjakan oleh Linux adalah melakun cd /usr/local/bin (yaitu direktori dimana binary helloworld disimpan) dan melakukan rm-f untuk binary helloworld. Periksalah kembali direktori tersebut dan cek apakah binary tersebut sudah hilang atau belum

```
#cd /usr/local/bin
#ls -al
```

2. Cara lain adalah dengan menggunakan tool checkinstall. Kita dapat menggunakan tool tersebut dengan mengganti make install dengan check install. Cukup tekan y, saat pertama kali dan lakukan enter terus-menerus sewaktu menggunakan check install. Tool checkinstall akan membentuk .deb dari source package kita.

```
$cd /home/<user>Downloads/helloworld-0.1
```

```
$./configure
$make
$checkinstall --type=debian --install=yes
```

Lakukan instalasi dengan dpkg

```
#dpkg -i helloword-01..._i386.deb
```

Sekarang coba lakukan remove

```
#dpkg -r helloworld
```

Cek kembali /usr/local/bin, lihat apakah helloworld sudah terhapus

```
#cd /usr/local/bin
#ls -al
```

TUGAS:

1. Ambil salah satu paket yang ada di Linux dan lakukan instalasi dengan menggunakan ./configure, make, make install

2. Cobalah membuat makefile untuk file helloworld

LAPORAN:

DAFTAR PUSTAKA:

1. https://tssurya.wordpress.com/2014/08/19/adding-a-hello-world-system-call-to-linux-kernel-3-16-0/
2. https://medium.com/@ssreehari/implementing-a-system-call-in-linux-kernel-4-7-1-6f98250a8c38
3. https://www.ibm.com/developerworks/library/l-system-calls/
4. http://cs.carleton.edu/faculty/jondich/courses/cs307_f05/documents/syscall.html
5. https://robots.thoughtbot.com/the-magic-behind-configure-make-make-install
6. http://mrbook.org/blog/tutorials/make/
7. https://www.gnu.org/software/make/manual/html_node/Introduction.html#Introduction
8. https://linuxacademy.com/blog/linux/troubleshooting-configure-make-and-make-install-tutorial/
9. https://www.cybercit,i.biz/tips/linux-shared-library-management.html
10. http://www.thegeekstuff.com/2012/06/linux-shared-libraries/
11. https://www.gnu.org/software/automake/manual/automake.html#amhello_0027s-configure_002eac-Setup-Explained
12. http://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html
13. http://www.thegeekstuff.com/2011/10/c-program-to-an-executable/
14. http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/