CS 35L

LAB 8,
TA: Sucharitha Prabhakar
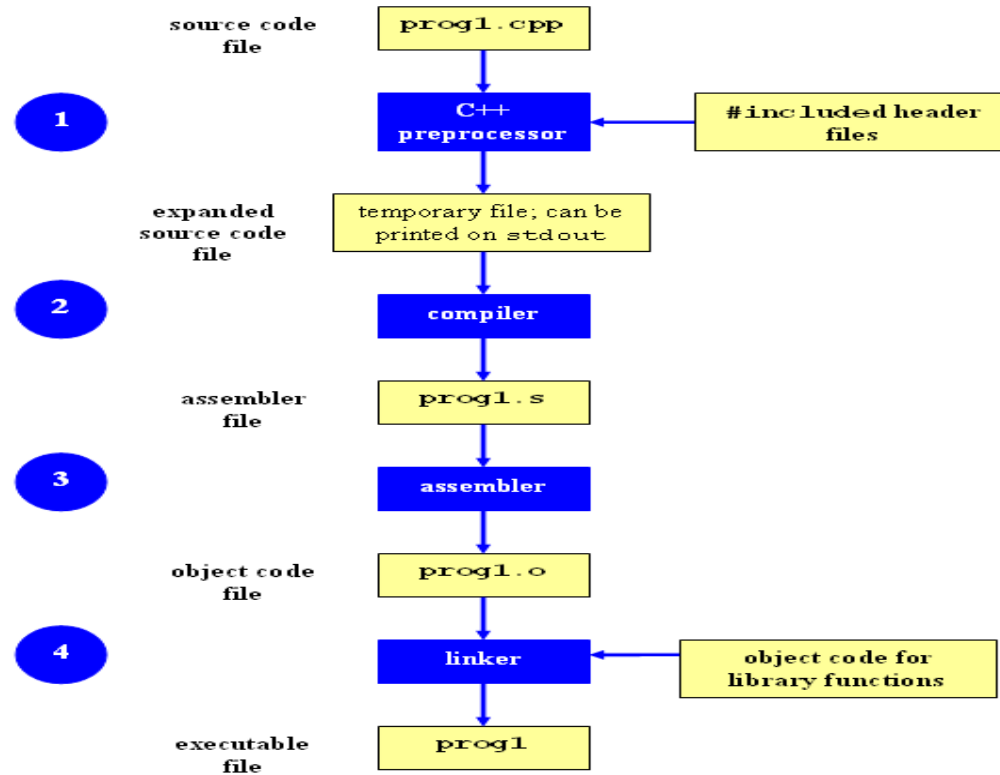EMAIL ID: prabhakarsucharitha@gmail.com

Outline

- Compilation process

- Make

- Automake

- Applying a patch

Compilation process

Compilation process

Preprocessing:

cc -E hello_world.c

Compilation:

cc -S hello_world.c

Assembly:

cc -c hello_world.c

hexdump hello_world.o

Linking

cc -o hello_world hello_world.c

Compilation

- shop.cpp

    - #includes shoppingList.h and item.h

- shoppingList.cpp

    - #includes shoppingList.h

- item.cpp

    - #includes item.h

- How to compile?

    - g++ -Wall shoppingList.cpp item.cpp shop.cpp –o shop

What if ?

- We change one of the header or source files?

  - Rerun command to generate new executable

- We only made a small change to item.cpp?

  - not efficient to recompile shoppinglist.cpp and shop.cpp

- Solution: avoid waste by producing a separate object code file for each source file

  - g++ -Wall –c item.cpp… (for each source file)

 - g++ item.o shoppingList.o shop.o –o shop (combine)

What if ?

- We change item.h?

- Need to recompile every source file that includes it & every source file that includes a header that includes it. Here: item.cpp and shop.cpp

- Difficult to keep track of files when project is large

  - Windows 7 ~40 million lines of code

  - Google ~2 billion lines of code

- **MAKE**

Make

GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files.

Make gets its knowledge of how to build your program from a file called the makefile, which lists each of the non-source files and how to compute it from other files.
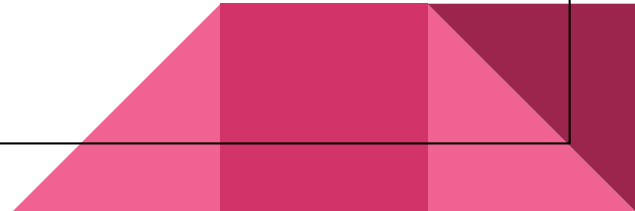
Advantages of Make

Make enables the end user to build and install your package without knowing the details of how that is done

Make figures out automatically which files it needs to update, based on which source files have changed. It also automatically determines the proper order for updating files, in case one non-source file depends on another non-source file.
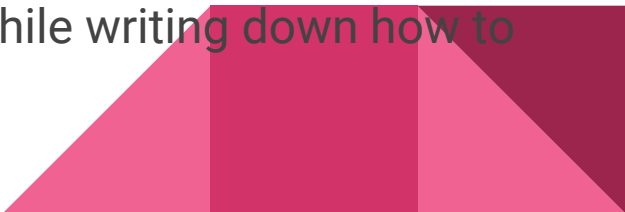
As a result, if you change a few source files and then run Make, it does not need to recompile all of your program.

Advantages of Make

Make is not limited to any particular language. For each non-source file in the program, the makefile specifies the shell commands to compute it. These shell commands can run a compiler to produce an object file, the linker to produce an executable, ar to update a library, or TeX or Makeinfo to format documentation.

Make is not limited to building a package. You can also use Make to control installing or uninstalling a package, generate tags tables for it, or anything else you want to do often enough to make it worth while writing down how to do it.

Build Process

- Configure

  - Script that checks details about the machine before installation

  - Dependency between packages

  - Creates 'Makefile'

- make

  - Requires 'Makefile' to run

  - Compiles all the program code and creates executables in current temporary directory

- make install

Automake

GNU Automake is a tool to automate parts of the compilation process.

It eases usual compilation problems. For example, it points to needed dependencies.

It automatically generates one or more *Makefile.in* from files called *Makefile.am*.

Each *Makefile.am* contains, among other things, useful variable definitions for the compiled software, such as compiler and linker flags, dependencies and their versions, etc.

Automake

GNU Automake is a tool to automate parts of the compilation process.

It eases usual compilation problems. For example, it points to needed dependencies.

It automatically generates one or more *Makefile.in* from files called *Makefile.am*.

Each *Makefile.am* contains, among other things, useful variable definitions for the compiled software, such as compiler and linker flags, dependencies and their versions, etc.

Automake

Makefile.am in <tests/project>

SUBDIRS = src

Makefile.am in <tests/project/src>

bin_PROGRAMS = helloworld

AM_CXXFLAGS = $(INTI_CFLAGS)

helloworld_SOURCES = main.cc helloworld.cc helloworld.h

helloworld_LDADD = $(INTI_LIBS)

Lab Assignment

Installing Software

- Windows

  - Installshield

  - Microsoft/Windows Installer

- OS X

  - Drag and drop from .dmg mount -> Applications folder

- Linux

  - rpm(Redhat Package Management)

  - RedHat Linux (.rpm)

Decompressing files

- Generally, you receive Linux software in the tarball format (.tgz) or (.gz)

- Decompress file in current directory:

  - $ tar –xzvf filename.tar.gz

  - Option –x: --extract

  - Option –z: --gzip

  - Option –v: --verbose

  - Option –f: --file

Problem

- Coreutils 7.6 has a problem

- Different users see different date formats

- $ ls −l /bin/bash

  - -rwxr-xr-x 1 root root 729040 2009-03-02 06:22 /bin/bash

  - -rwxr-xr-x 1 root root 729040 Mar 2   2009 /bin/bash

- Why?

  - Different locales

- Want the traditional Unix format for all users

Setup

- Download coreutils-7.6 to your home directory

    - Use 'wget'

- Untar and Unzip it

    - tar –xzvf coreutils-7.6.tar.gz

- Make a directory ~/coreutilsInstall in your home directory (this is where you'll be installing coreutils)

    - mkdir ~/coreutilsInstall

Setup

- Go into coreutils-7.6 directory. This is what you just unzipped.

- Read the INSTALL file on how to configure "make", especially --prefix flag

- Run the configure script using the prefix flag so that when everything is done, coreutils will be installed in the directory ~/coreutilsInstall

- Compile it: make

- Install it: make install

Reproduce bug

Reproduce the bug by running the version of 'ls' in coreutils 7.6

If you just type ls at CLI it won't run 'ls' in coreutils 7.6

> Why? Shell looks for /bin/ls

To use coreutils 7.6: $ ./ls -lrt

> This manually runs the executable in this directory (ls is in src directory)
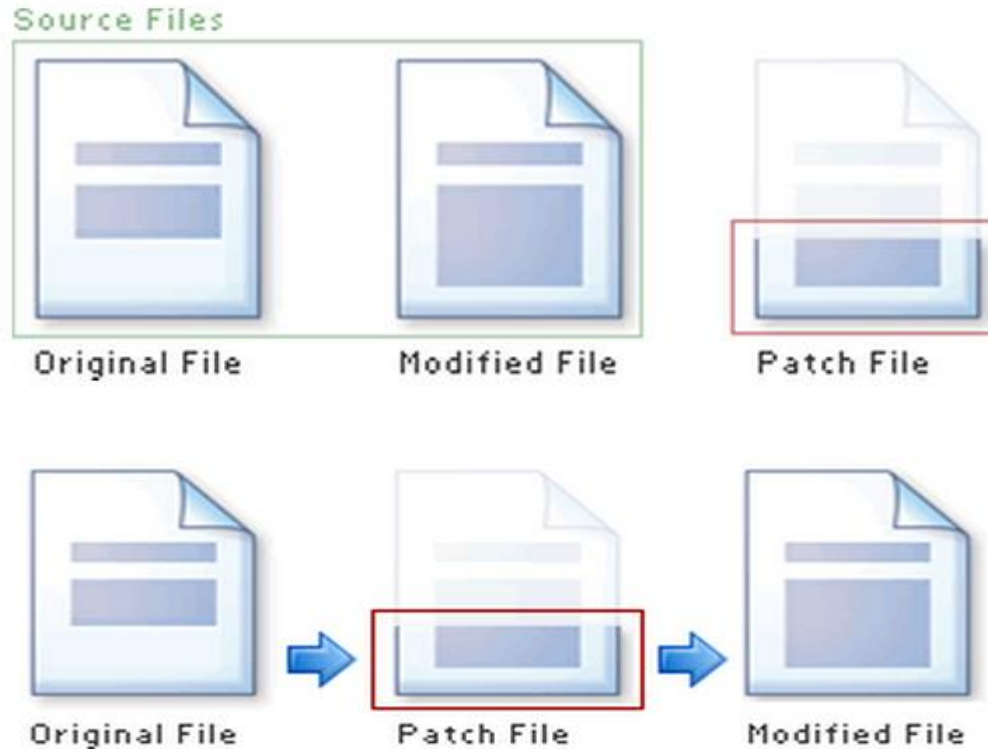
Patch

- A patch is a piece of software designed to fix problems with or update a computer program

- It's a diff file that includes the changes made to a file

- A person who has the original (buggy) file can use the patch command with the diff file to add the changes to their original file

Applying a Patch

diff Unified Format

diff −u original_file modified_file

--- path/to/original_file

+++ path/to/modified_file

@@ -l,s +l,s @@

@@: beginning of a chunk

l: beginning line number

s: number of lines the change chunk applies to for each file

A line with a:

Patching and Building

cd coreutils-7.6

vim or emacs patch_file: copy and paste the patch content

patch –pnum < patch_file

    'man patch' to find out what pnum does and how to use it

cd into the coreutils-7.6 directory and type make to rebuild patched ls

Don't install!!

Testing fix

- Test the following:

  - Modified ls works

  - Installed unmodified ls does NOT work

- Test on:

  - A file that has been recently modified

    - Make a change to an existing file or create a new file

  - A file that is at least a year old

    - touch –t 201401210959.30 test_file