Lab 3

Modifying Programs

How to Install Software

- Windows
  - Installshield
  - Microsoft/Windows Installer
- Linux
  - rpm(Redhat Package Management)
    - RedHat Linux (.rpm)
  - apt-get(Advanced Package Tool)
    - Debian Linux, Ubuntu Linux (.deb)
  - **Good old build process**
    - **configure, make, make install**

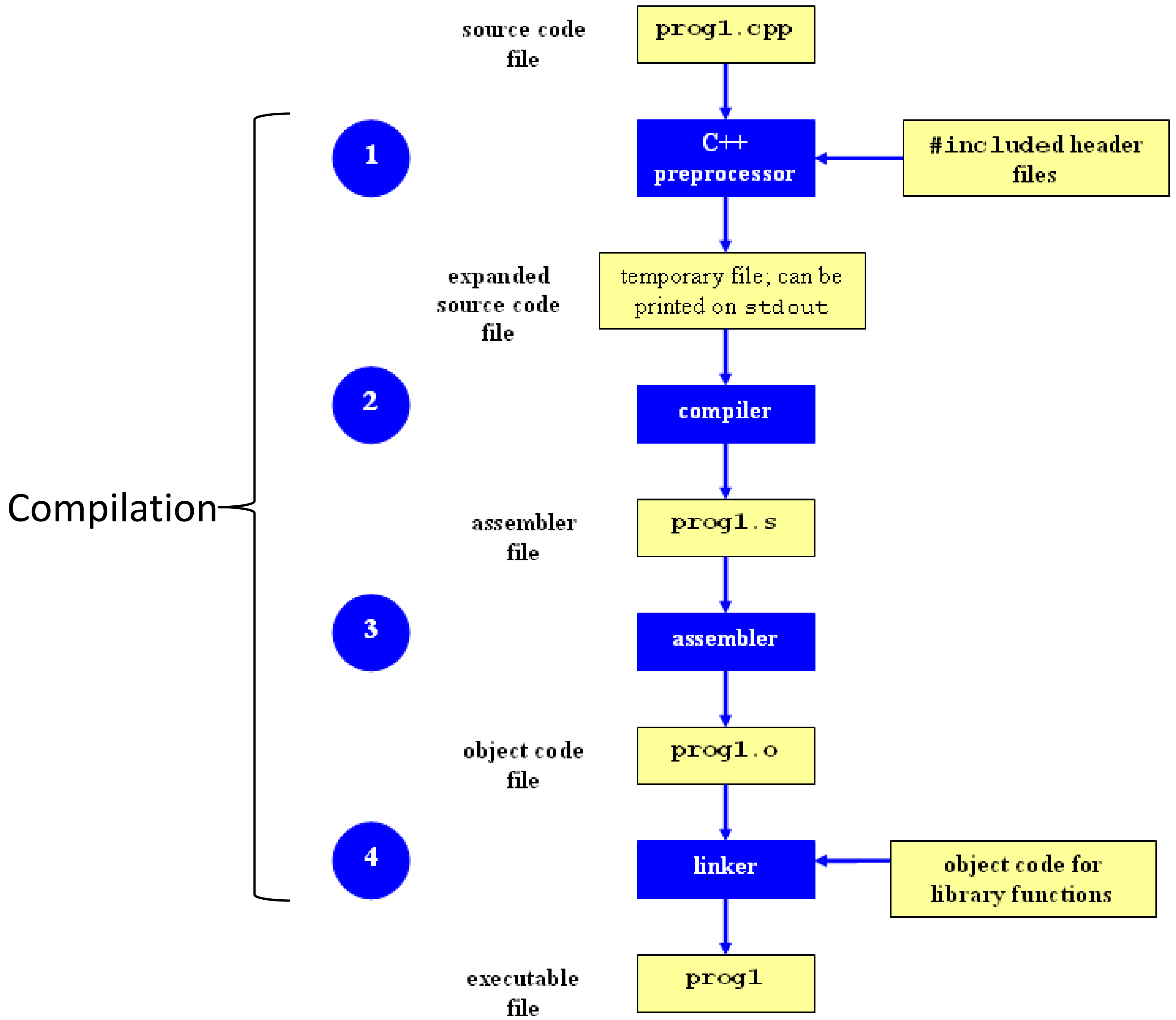
Decompressing Files

- Generally, you would get 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

Compilation Process



Command-Line Compilation

- shop.c
  - #includes shoppingList.h and item.h
- shoppingList.c
  - #includes shoppingList.h
- item.c
  - #includes item.h
- How to compile?
  - **gcc shoppingList.c item.c shop.c –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.c?**
  - not efficient to recompile shoppinglist.c and shop.c
  - ⇒Solution: avoid waste by producing a separate object code file for each source file
    - gcc –c item.c… (for each source file)
    - gcc item.o shoppingList.o shop.o –o shop (combine)
    - Less work for compiler, saves time but more commands

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.c and shop.c
  - Difficult to keep track of files when project is large

=> Make

Make

- Utility for managing large software projects
- Compiles files and keeps them up-to-date
- Efficient Compilation (only files that need to be recompiled)

Makefile Example

```
# Makefile - A Basic Example
all : shop  #usually first
item.o : item.c item.h
        gcc -c item.c
shoppingList.o : shoppingList.c item.h shoppingList.h
        gcc -c shoppingList.c
shop.o : shop.c item.h shoppingList.h
        gcc -c shop.c
shop : item.o shoppingList.o shop.o
        gcc -o shop item.o shoppingList.o shop.o
clean :
        rm -f item.o shoppingList.o shop.o shop
```

$\}$ Rule

🟩 Comments

🟦 Targets $\}$ Dependency Line

🟪 Prerequisites

🟫 Commands

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**
  - make utility searches for a label named install within the Makefile, and executes only that section of it
  - executables are copied into the final directories (system directories)

Lab 3

- 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
- Fix the ls program

Getting Set Up (Step 1)

- 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

Building coreutils (Step 2)

- Go into coreutils-7.6 directory. This is what you just unzipped.
- Read the INSTALL file on how to configure the project, especially with **--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 (won't work on Linux server without proper prefix!)

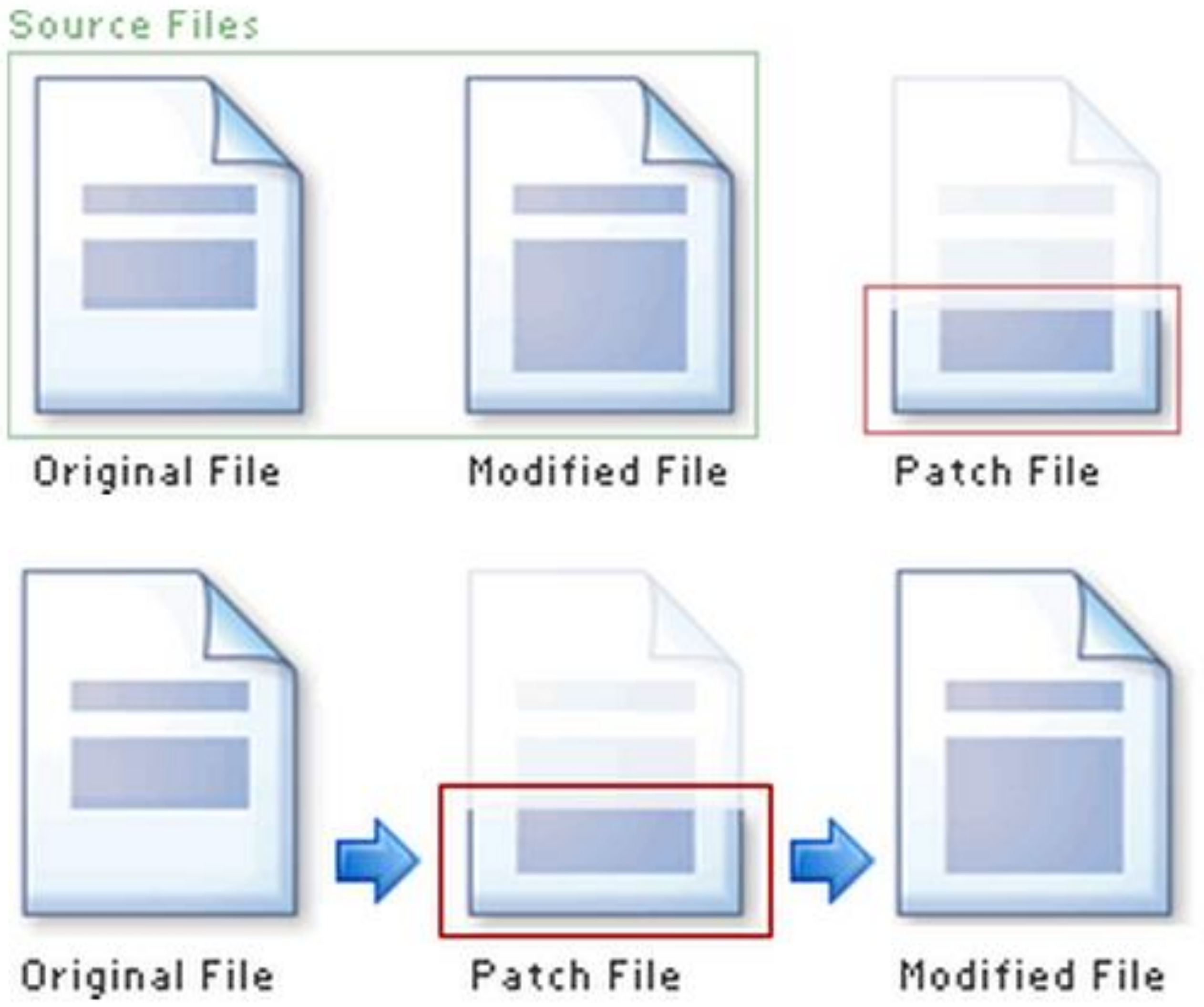
Reproduce Bug (Step 3)

- Reproduce the bug by running the version of ‘ls’ in coreutilsInstall

Patching

- 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

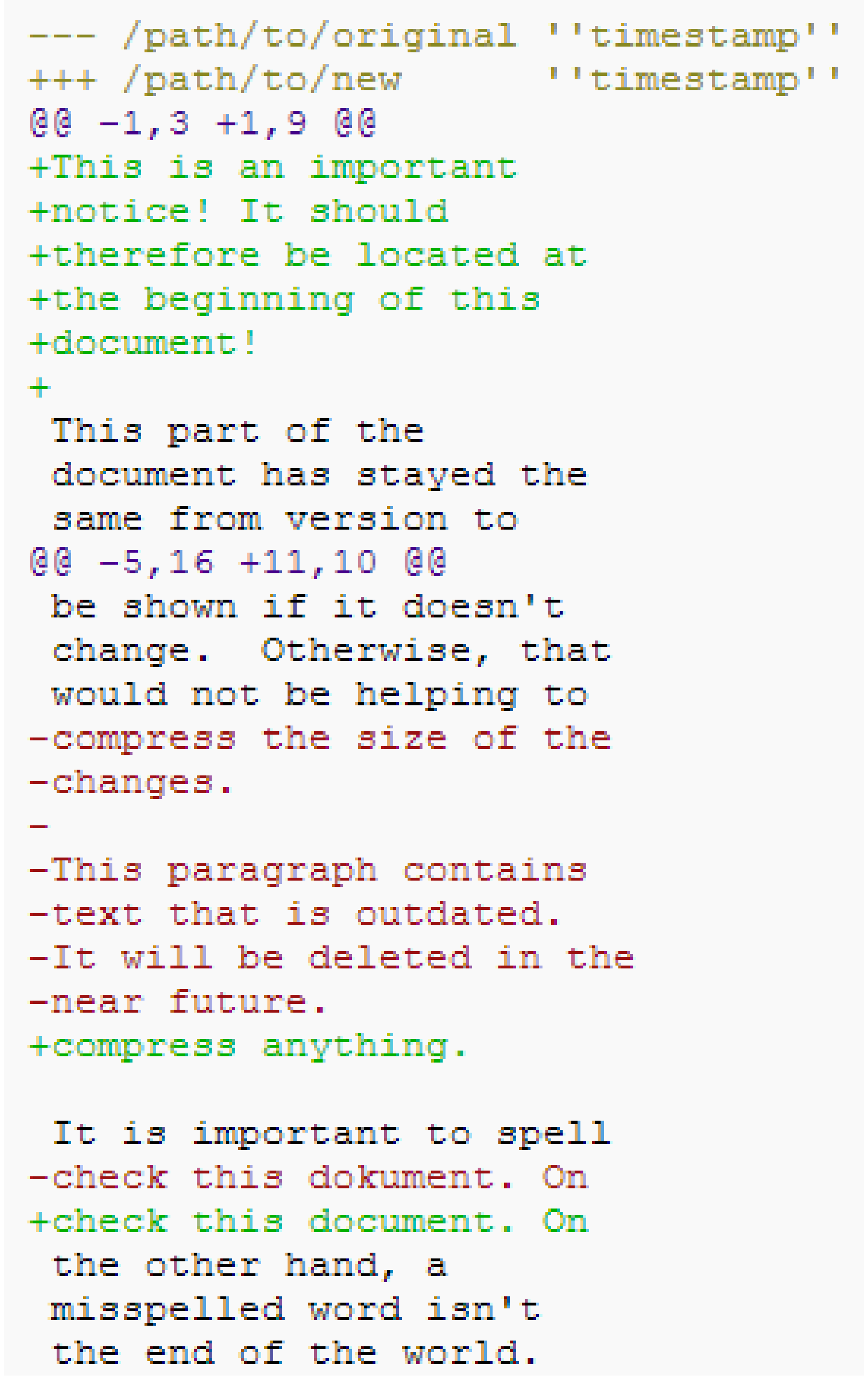


diff Unified Format

- diff –u original_file modified_file

- --- path/to/original_file
- +++ path/to/modified_file

- @@ -l,s +l,s @@
  - @@: beginning of a hunk
  - l: beginning line number
  - s: number of lines the change hunk applies to for each file
  - A line with a:
    - - sign was deleted from the original
    - + sign was added to the original
    -    stayed the same



```diff
--- /path/to/original ''timestamp''
+++ /path/to/new      ''timestamp''
@@ -1,3 +1,9 @@
+This is an important
+notice! It should
+therefore be located at
+the beginning of this
+document!
+
 This part of the
 document has stayed the
 same from version to
@@ -5,16 +11,10 @@
 be shown if it doesn't
 change.  Otherwise, that
 would not be helping to
-compress the size of the
-changes.
-
-This paragraph contains
-text that is outdated.
-It will be deleted in the
-near future.
+compress anything.
 
 It is important to spell
-check this dokument. On
+check this document. On
 the other hand, a
 misspelled word isn't
 the end of the world.
```

Patching and Building (Steps 4 & 5)

- 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 p**num** does and how to use it
- `cd` into the coreutils-7.6 directory and type make to rebuild patched ls.c. Don’t install!!

Testing Fix (Step 6)

- Test the following:
  - Modified ls works
  - Installed unmodified ls does NOT work
- Test on:
  - 1) a file that has been recently modified
    - Make a change to an existing file or create a new file
  - 2) a file that is at least a year old
    - touch –t 201504100959.30 *test_file*

- Answer Q1 and Q2