CS 35L- Software Construction Laboratory

Fall 18

TA: Guangyu Zhou

Lab 3

Announcement

- Signup for Assignment 10 Presentation (No later than Oct 21, 11:55pm)
  - Use **UCLA** account to register at the following link
  - https://docs.google.com/spreadsheets/d/1L2IeP7WRbCmAygSPN5pq7SU2pTddKNooUteBGetde2M/edit?usp=sharing
  - Topic on recent research in computer science
    - **Technical** content is required
  - 1 or 2 people
  - ~10 minutes talk in class (~12 min for teams)
  - Use slides and upload to CCLE before presentation
  - Participation in Q&A
  - Brief Research report (due in the last week)

Week 2 Review

- Unix wildcards, basic regular expressions
- More advanced commands (e.g., grep, find)
- Text editing tools (tr, sed)
- Pipelines and redirection
- Simple shell scripting

Some Useful Exercises for HW2

- How to write script to see if file1 and file2 are same?
  - cmp file1 file2
- How to obtain the return value or exist status of previous command?
  - output=$(cmp file1 file2); echo "$output"
  - cmp file1 file2; echo $?
- What's the difference between ', ", and `:
  - date=20021226
  - echo '$date'
  - echo "$date"
  - echo "`date`"

Regular expression exercises

- Which one would match “Gogle”, “Google” and “Gooogle” but not “Ggle”?
- – Answer: “Go+gle”

- Which regular expression would match any version of the word “Google” that has an even number of o’s?
- – Answer: “G(oo)+gle”

Regular expression exercises

- Which line(s) would this regular expression match? “^T.+e$”
- A. The ; B. Te ; C. Three ; D. Then
- Answer: The, Three (ERE)

- Which regular expression(s) would match the words “Ted”, “Ned” and “Sed”?
- A. (T|N|S)ed ; B. [T N S]ed ; C. .ed ; D. [L-U]?ed ; E. *ed
- Answer:
- A., B., C.,
- D., E. (ERE)

Some regular expression examples

- Credit card matching:
  - All Visa card numbers start with a 4. New cards have 16 digits. Old cards have 13.

```
^4[0-9]{12}(?:[0-9]{3})?$
```

awk command

- awk [options] [program file]
  - **-F fs** To specify a file separator.
  - **-f file** To specify a file that contains awk script.
  - **-v var=value** To declare a variable.
- Tasks:
  - Define variables.
  - Use string and arithmetic operators.
  - Use control flow and loops.
  - Generate formatted reports.

tr vs sed vs awk

- sed is a **stream** editor. It works with streams of characters on a per-line basis.
  - It has a primitive programming language that includes goto-style loops and simple conditionals (in addition to pattern matching and address matching).
  - There are essentially only two "variables": pattern space and hold space. Readability of scripts can be difficult. Mathematical operations are extraordinarily awkward at best.
- tr perform character based transformation but sed perform string based transformation.
  - echo I am a good boy | tr 'good' 'test’
  - echo I am a good boy | sed 's/good/best/g'
- awk is oriented toward **delimited** fields on a per-line basis.
  - It has much more robust programming constructs including if/else, while, do/while and for (C-style and array iteration).
  - Mathematical operations resemble those in C.
  - It has printf and functions.

Awk basic usage

- With awk, you can process text files. Awk assigns some variables for each data field found:
  - $0 for the whole line.
  - $1 for the first field.
  - $2 for the second field.
  - $n for the nth field.
- The whitespace character like space or tab is the default separator between fields in awk.
- $ awk '{print $1}' myfile

```
likegeeks@likegeeks-VirtualBox ~/Desktop $ cat myfile
This is a test.
This is the second test.
This is the thrid test.
This is the fourth test.
likegeeks@likegeeks-VirtualBox ~/Desktop $ awk '{print $1}' myfile
This
This
This
This
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Awk basic usage

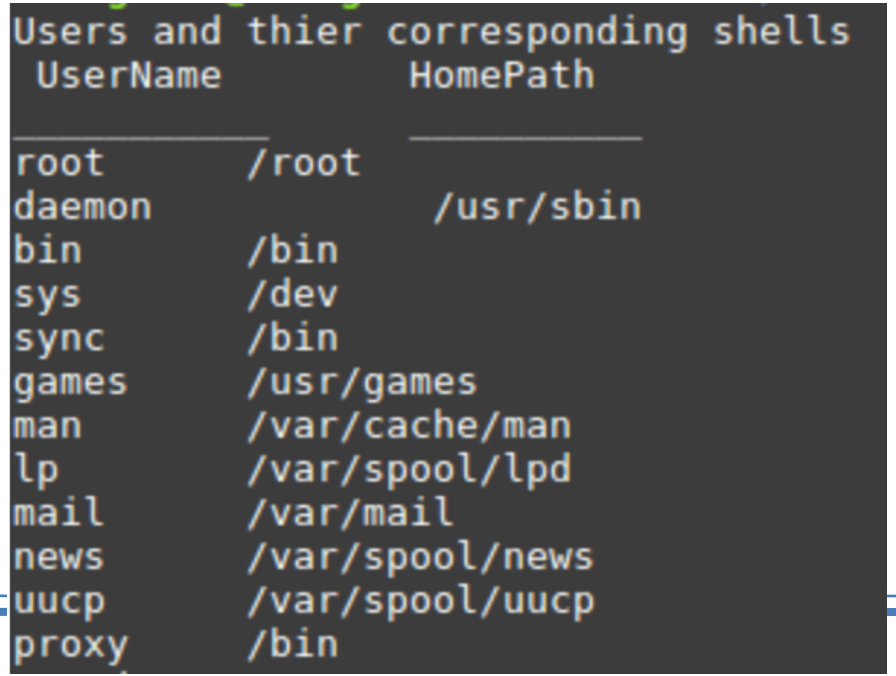
- Sometimes the separator in some files is not space nor tab but something else. You can specify it using –F option:
- awk -F: '{print $0"---> "$1}' /etc/passwd

```
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false---> nobody
root:*:0:0:System Administrator:/var/root:/bin/sh---> root
daemon:*:1:1:System Services:/var/root:/usr/bin/false---> daemon
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico---> _uucp
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false---> _taskgated
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false---> _networkd
_installassistant:*:25:25:Install Assistant:/var/empty:/usr/bin/false---> _installassistant
_lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false---> _lp
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false---> _postfix
_scsd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false---> _scsd
_ces:*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false---> _ces
```

Awk basic usage

- ***echo "Hello Eggert" | awk '{$2=”Guangyu"; print $0}’***
  - => Hello Guangyu
- Create your awk script then run.
- ***awk -f myscript /etc/passwd***

```
BEGIN {
print "Users and thier corresponding home"
print " UserName \t HomePath"
print "_____________ \t ____________"
FS=":"
}
{
print $1 "  \t  " $6
}
END {
print "The end"
```



Modify and Rewrite Software

Week 3

Outline

- **Build from source & Bug Fixing**
- **Compile using makefile**
- **File patching**
- Introduction to Python

How to Install Software

- Windows
  - Microsoft/Windows Installer
- OS X
  - Drag and drop from .dmg mount -> Applications folder
- 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**

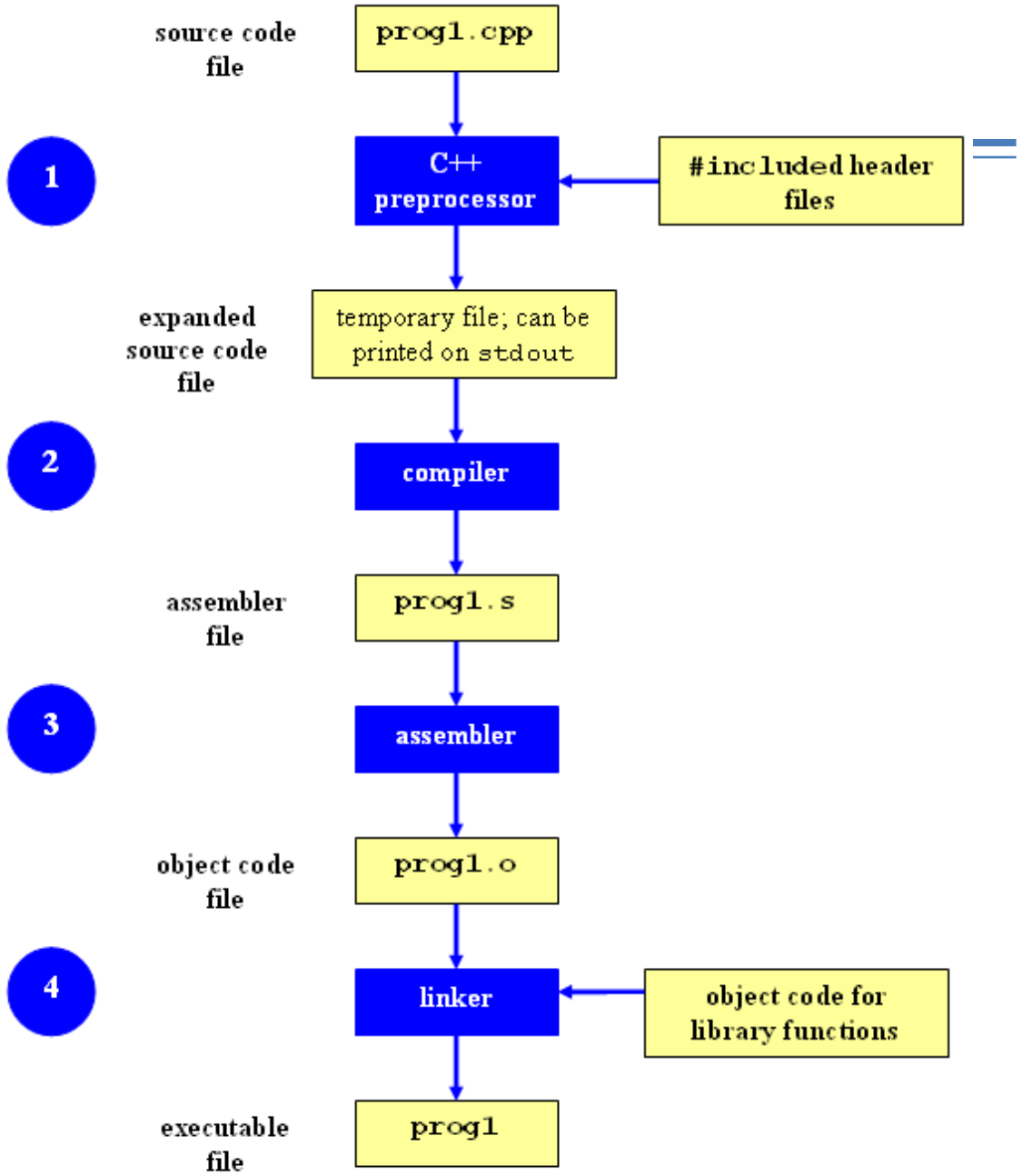
How to decompose files

- 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

Compilation Process

Compilation



Command-Line 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)
    - 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.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

- 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
shop : item.o shoppingList.o shop.o
        g++ -g -Wall -o shop item.o shoppingList.o shop.o
item.o : item.cpp item.h
        g++ -g -Wall -c item.cpp
shoppingList.o : shoppingList.cpp shoppingList.h
        g++ -g -Wall -c shoppingList.cpp
shop.o : shop.cpp item.h shoppingList.h
        g++ -g -Wall -c shop.cpp
clean :
        rm -f item.o shoppingList.o shop.o shop
```

Rule

Comments

Targets

Prerequisites

Dependency Line

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)

Task: Fixing a bug

- On a certain computer (not necessarily seasnet), the command ls -l /bin/bash displays:

  $ ls -l /bin/bash

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

- But this is a bug, you want it to display traditional Linux format:

  $ ls -l /bin/bash

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

Steps for fixing bugs

- Outputs the ‘buggy result’
  - ls -l --time-style=long-iso /bin/bash
- Login to Seasnet
- Download coreutils to a temporary directory
  - How to download file (wget)
- Untar\Unzip it
  - How to unzip a file
  - man tar
  - cd to the newly created coreutils folder

The tar command

- Usage of tar
  - tar –cvf <tarfilename.tar> <target directories> # creates tar file.
  - tar –tvf <tarfilename.tar> # list tar file contents
  - tar –xvf <tarfilename.tar> # extracts tar file
  - -z option: generate .gz files
- Tips
  - Always create tarfile in target directory (relative file/directory names)
  - Always list tarfile before extracting (insure relative file names)
  - Always extact tarfile in target directory (relative file/directory names)
- Example
  - tar –tvf a2.tar
  - tar –xzvf filename.tar.gz

Compile using makefile

- Download a utility from the internet to your Linux machine
- There are no binaries, but source code and makefile is available
- Compile and build to install it
- Reading text files(e.g. README) in the program folder gives clues how to install the program

Compile using makefile

- The order of compilation is usually:
  - **./configure**
  - **make**
  - **make install**
- Usage: man make
- View makefile in the programs folder for details
- Configure
  - Setup the path for make and install
  - Should use **absolute path** here
- Demo

Makefile and make

- Function of makefile: Instruct how to compile and link a program
- The make program allows you to use macros, which are similar to variables to codify how to compile a set of source code
  - Macros are assigned as BASH variable:
  - CFLAGS= -O -systype bsd43
  - LIBS = "-lncurses -lm –lsdl”
- Makefile is invoked with make <target_name>

Standard “targets”

- People have come to expect certain targets in Makefiles. You should always browse first, but it's reasonable to expect that the targets all (or just make), install, and clean will be found
  - **make**: compile the default target
  - **make all**: compile everything so that you can do local testing before installing
  - **make install**: install things in the right places. But watch out that things are installed in the right place for your system
  - **make clean**: clean things up. Get rid of the executables, any temporary files, object files, etc.
- Details: see supplement materials [GCC and Make]

Apply a patch

- Read the patch bug report

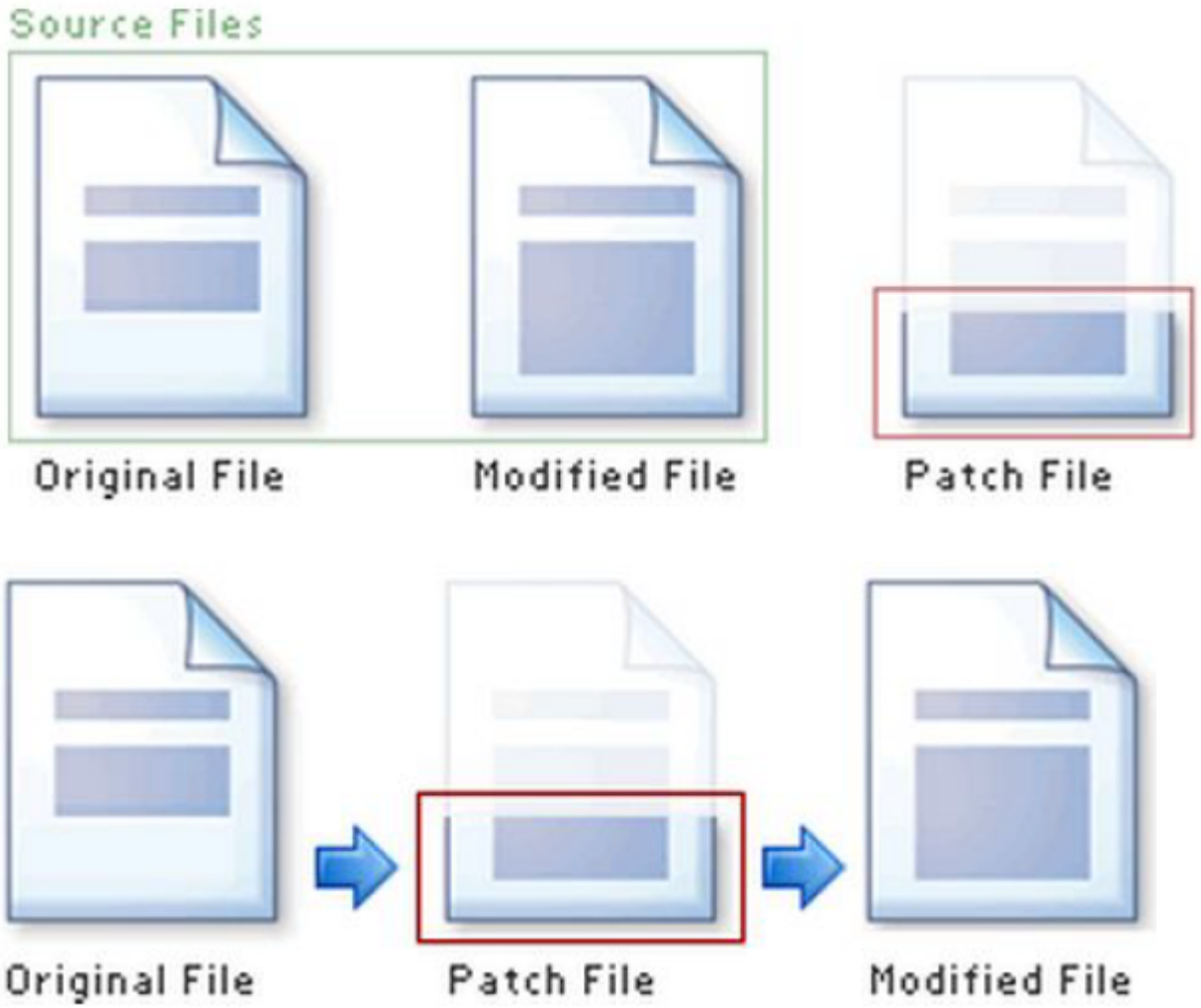
http://lists.gnu.org/archive/html/bug-coreutils/2009-09/msg00410.html

- Understand what part of the code is being fixed

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
- Patch Command
  - Usage: patch [options] [originalfile] [patchfile]
  - **-pnum**: strip the smallest prefix containing num leading slashes from each file name found in the patch file
  - Examples: see supplement materials [Patch command]

Applying a patch



diff Unified Format

- --- path/to/original_file
- +++ path/to/modified_file
- @@ -l,s +l,s @@
  - @@: beginning and end of a hunk
  - l: beginning line number
  - s: number of lines the change hunk applies to for each file
  - A line with:
    - - sign was deleted from the original
    - + sign was added in the new file
    - ‘ ‘ stayed the same

Applying the Patch

```
diff --git a/src/ls.c b/src/ls.c
    index 1bb6873..4531b94 100644
    --- a/src/ls.c
    +++ b/src/ls.c
    @@ -2014,7 +2014,6 @@ decode_switches (int argc, char **argv)
                  break;

                case long_iso_time_style:
    -           case_long_iso_time_style:
                  long_time_format[0] = long_time_format[1] = "%Y-%m-%d %H:%M";
                  break;

    @@ -2030,13 +2029,8 @@ decode_switches (int argc, char **argv)
                           formats.  If not, fall back on long-iso format.  */
                        int i;
                        for (i = 0; i < 2; i++)
    -                     {
    -                       char const *locale_format =
    -                         dcgettext (NULL, long_time_format[i], LC_TIME);
    -                       if (locale_format == long_time_format[i])
    -                         goto case_long_iso_time_style;
    -                       long_time_format[i] = locale_format;
    -                     }
    +                     long_time_format[i] =
    +                       dcgettext (NULL, long_time_format[i], LC_TIME);
                      }
                  }
              /* Note we leave %5b etc. alone so user widths/flags are honored.  */
```

Additional Resource

- Guide: Building and Installing Software Packages for Linux
  https://www.tldp.org/HOWTO/pdf/Software-Building-HOWTO.pdf