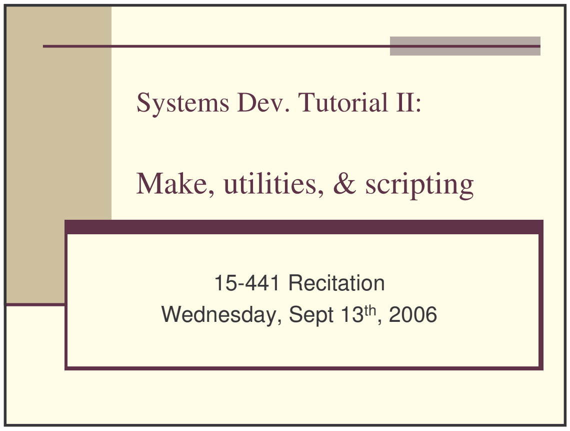
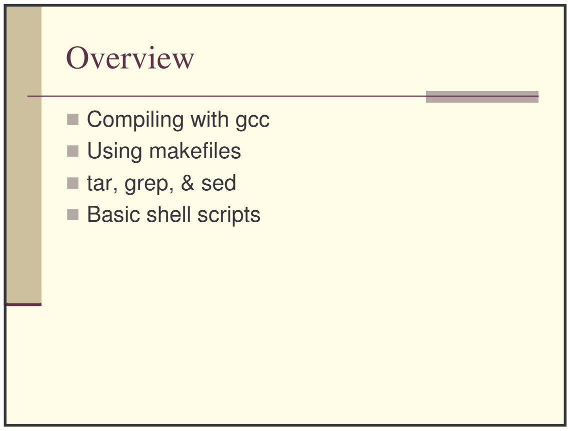


Systems Dev. Tutorial II:

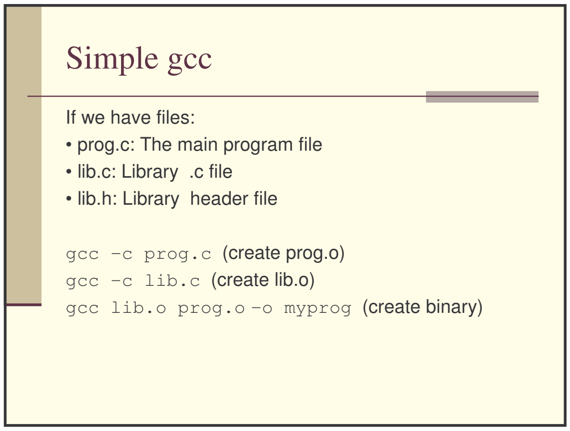
Make, utilities, & scripting

15-441 Recitation
Wednesday, Sept 13th, 2006



Overview

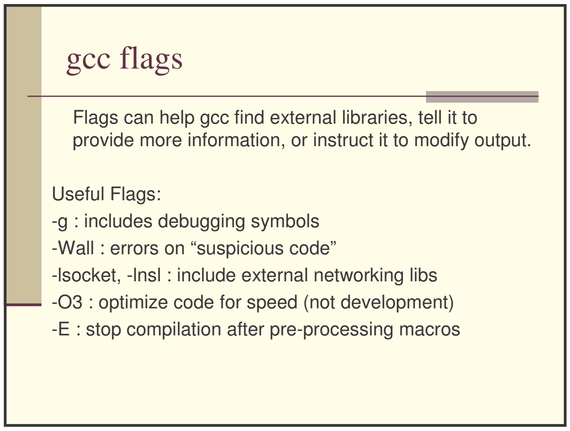
- Compiling with gcc
- Using makefiles
- tar, grep, & sed
- Basic shell scripts



Simple gcc

If we have files:
- prog.c: The main program file
- lib.c: Library .c file
- lib.h: Library header file

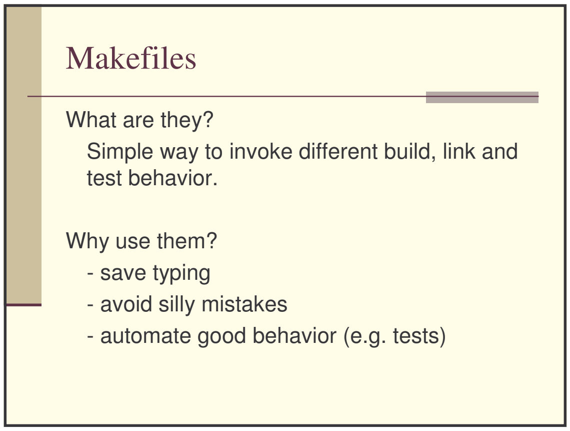
`gcc –c prog.c` (create prog.o)
`gcc –c lib.c` (create lib.o)
`gcc lib.o prog.o –o myprog` (create binary)



gcc flags

Flags can help gcc find external libraries, tell it to provide more information, or instruct it to modify output.

Useful Flags:
-g : includes debugging symbols
-Wall : errors on “suspicious code”
-lsocket, -lnsl : include external networking libs
-O3 : optimize code for speed (not development)
-E : stop compilation after pre-processing macros

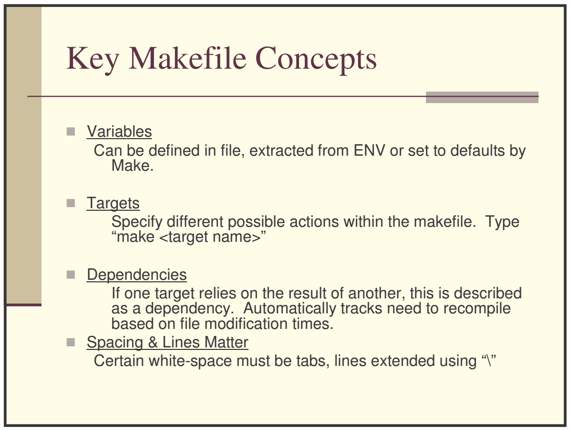


Makefiles

What are they?

Simple way to invoke different build, link and test behavior.

Why use them?
- save typing
- avoid silly mistakes
- automate good behavior (e.g. tests)



Key Makefile Concepts

- Variables
  Can be defined in file, extracted from ENV or set to defaults by Make.
- Targets
  Specify different possible actions within the makefile. Type “make <target name>”
- Dependencies
  If one target relies on the result of another, this is described as a dependency. Automatically tracks need to recompile based on file modification times.
- Spacing & Lines Matter
  Certain white-space must be tabs, lines extended using “\”

Makefile Example

```
# This is a comment

CFLAGS=-Wall -g
LIBS=lib.o \
        lib2.o
HEADERS=lib.h
BINS=prog

all: ${BINS}

prog: prog.o ${LIBS} ${HEADERS}
   ${CC} ${LDFLAGS} prog.o ${LIBS} -o $@

test:
   ./run_tests.sh

clean:
   /bin/rm -rf ${BINS} *.o core *.core
```

3 very useful utilities

- **tar** : create and unpack archives of files
- **grep**: search for a text string or regular expression within a set of files…
- **sed**: powerful search and replace for within a set of files

tar

collect files & directories into a single file, possibly compressed, archive.

examples:

archive: `tar czf my_code.tar.gz my_code/`

unpack: `tar xzf my_code.tar.gz`

grep

search a set of files for lines that contain a certain string, or match a regular expression.

basic: `egrep hello debug.txt`

recursive: `egrep -r hello .`

Advanced (line number, case insenstive):

```
egrep -n -i‘strcpy|strlen’ sock.c
```

sed

“stream editor” useful for powerful search and replace operations, of filtering data files.

examples:

Search/replace:

```
sed -e ‘s/Bush/Andersen/g’ votes.txt > new_votes.txt
```

Filter for some text:

```
sed -e ‘/[^error1]/d’ debug.txt
```

shell scripting

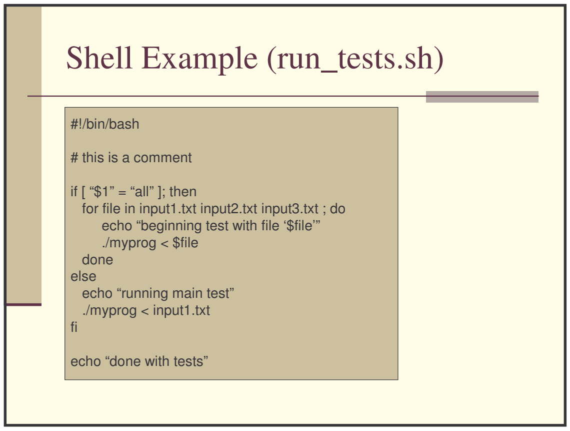
Basic idea: anything you type into the command-line can be automated.

e.g. create script to run all tests.

Suggestions:

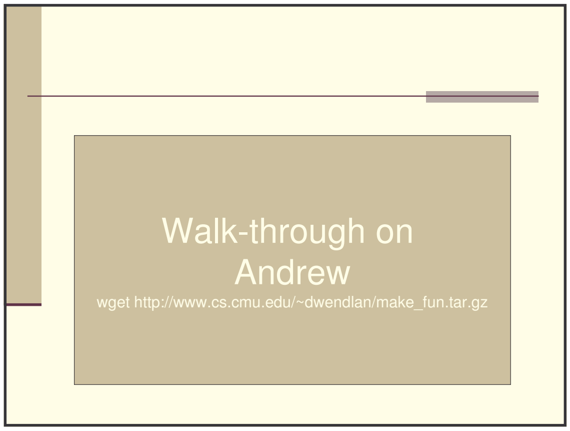
1) Use –x option to debug line-by-line

2) With great power, comes….



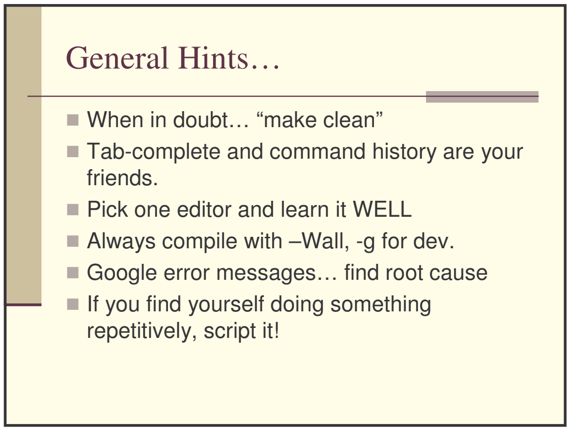
Shell Example (run_tests.sh)

```
#!/bin/bash

# this is a comment

if [ “$1” = “all” ]; then
   for file in input1.txt input2.txt input3.txt ; do
        echo “beginning test with file ‘$file’”
        ./myprog < $file
   done
else
   echo “running main test”
   ./myprog < input1.txt
fi

echo “done with tests”
```



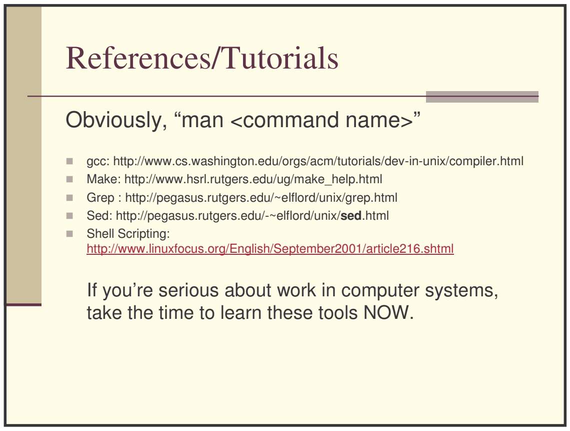
Walk-through on Andrew

wget http://www.cs.cmu.edu/~dwendlan/make_fun.tar.gz



General Hints…

- When in doubt… “make clean”
- Tab-complete and command history are your friends.
- Pick one editor and learn it WELL
- Always compile with –Wall, -g for dev.
- Google error messages… find root cause
- If you find yourself doing something repetitively, script it!

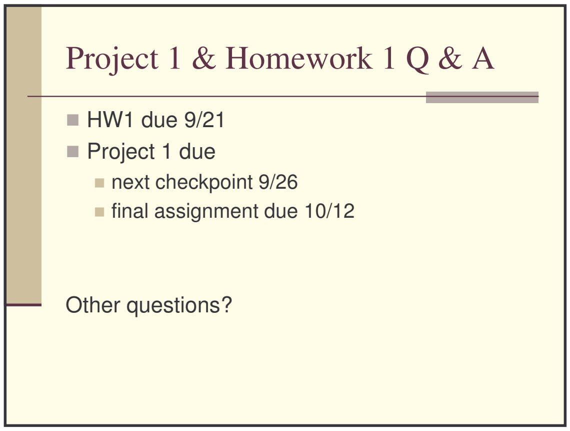


References/Tutorials

Obviously, “man <command name>”

- gcc: http://www.cs.washington.edu/orgs/acm/tutorials/dev-in-unix/compiler.html
- Make: http://www.hsrl.rutgers.edu/ug/make_help.html
- Grep : http://pegasus.rutgers.edu/~elflord/unix/grep.html
- Sed: http://pegasus.rutgers.edu/-~elflord/unix/**sed**.html
- Shell Scripting: http://www.linuxfocus.org/English/September2001/article216.shtml

If you’re serious about work in computer systems, take the time to learn these tools NOW.



Project 1 & Homework 1 Q & A

- HW1 due 9/21
- Project 1 due
  - next checkpoint 9/26
  - final assignment due 10/12

Other questions?