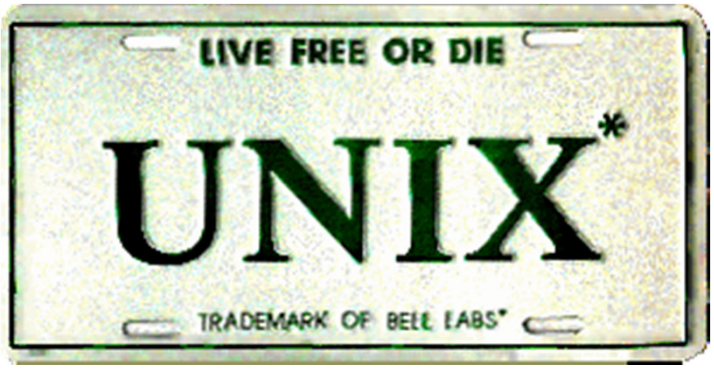


make: File Dependency System

Lecturer: Prof. Andrzej (AJ) Bieszczad
Email: andrzej@csun.edu
Phone: 818-677-4954

“UNIX for Programmers and Users”
Third Edition, Prentice-Hall, GRAHAM GLASS, KING ABLES

Slides partially adapted from Kumoh National University of Technology (Korea) and NYU

make: File Dependency System

- **THE UNIX FILE-DEPENDENCY SYSTEM: MAKE**

  - In order to update the two main program executables "main1" and "main2" manually,

    1. Recompile "reverse.c"
    2. Link "reverse.o" and "main1.o" to produce a new version of "main1".
    3. Link "reverse.o" and "main2.o" to produce a new version of "main2".

  - imagine a system with 1000 object modules and 50 executable programs.
    $\Rightarrow$ a nightmare.

make: File Dependency System

One way to avoid these problems to use the UNIX **make** utility, which allows you to create a *makefile* that contains a list of all file interdependencies for each executable.

Developed at Bell Labs around 1978 by S. Feldman (now at IBM)

make: File Dependency System

- **THE UNIX FILE-DEPENDENCY SYSTEM: MAKE**

  - Once such a file is created,
    to re-create the executable is easy:

    $ **make -f** *makefile*

  - **synopsis of make**

    Utility: **make** [**-f** makefile]

    make is a utility that updates a file based on a series of dependency rules stored in a special-format "make file".

    The **-f** option allows you to specify your own make -filename;
    if none is specified, the name "makefile" is assumed.

make: File Dependency System

- **Make Files**

  - To use **the make utility** to maintain an executable file,
    ⇒ create a make file.

  - contains a list of all of the interdependencies
    that exist between the files that are used to create the executable.

  Example:
    the name of the make file for "main1" , "main1.make"

  - a make file contains make rules of the form

    *target*List: *dependency*List
    *command*List

  *target*List : a list of target files
  *dependency*List : a list of files on which the files in targetList depend.
  *command*List : a list of zero or more commands,
  separated by new lines,

make: File Dependency System

- **Make Files**

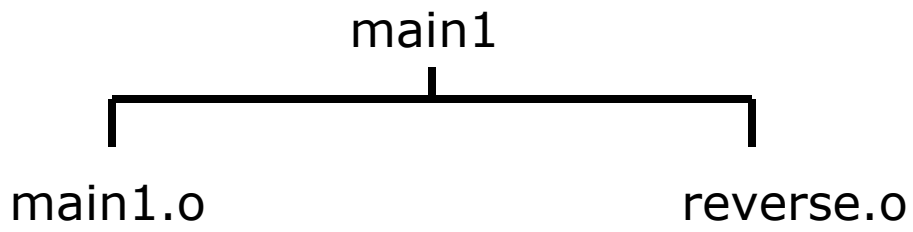
  - For example, "main1.make"

```
main1:  main1.o   reverse.o
        cc   main1.o   reverse.o   -o  main1

main1.o : main1.c reverse.h
        cc  -c  main1.c

reverse.o : reverse.c  reverse.h
        cc  -c  reverse.c
```

- **The Order of Make Rules**

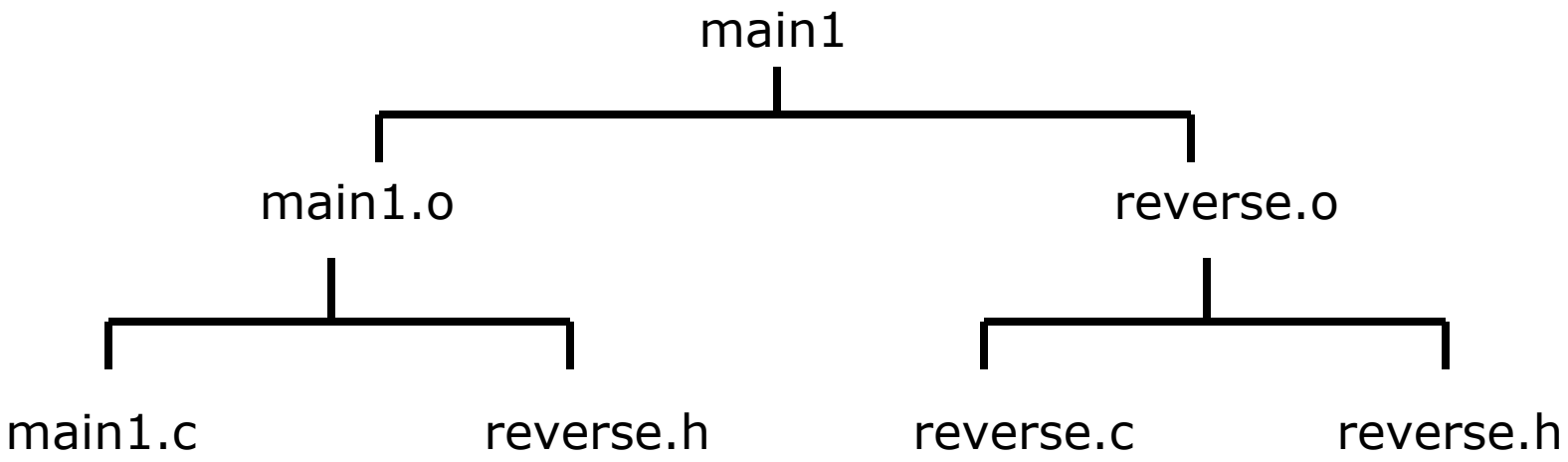
  - The **make** utility creates a "tree" of interdependencies.
    Each target file in the first rule is a root node of a dependency tree,
    Each file in its dependency list is added as a leaf of each root node.

    In example,



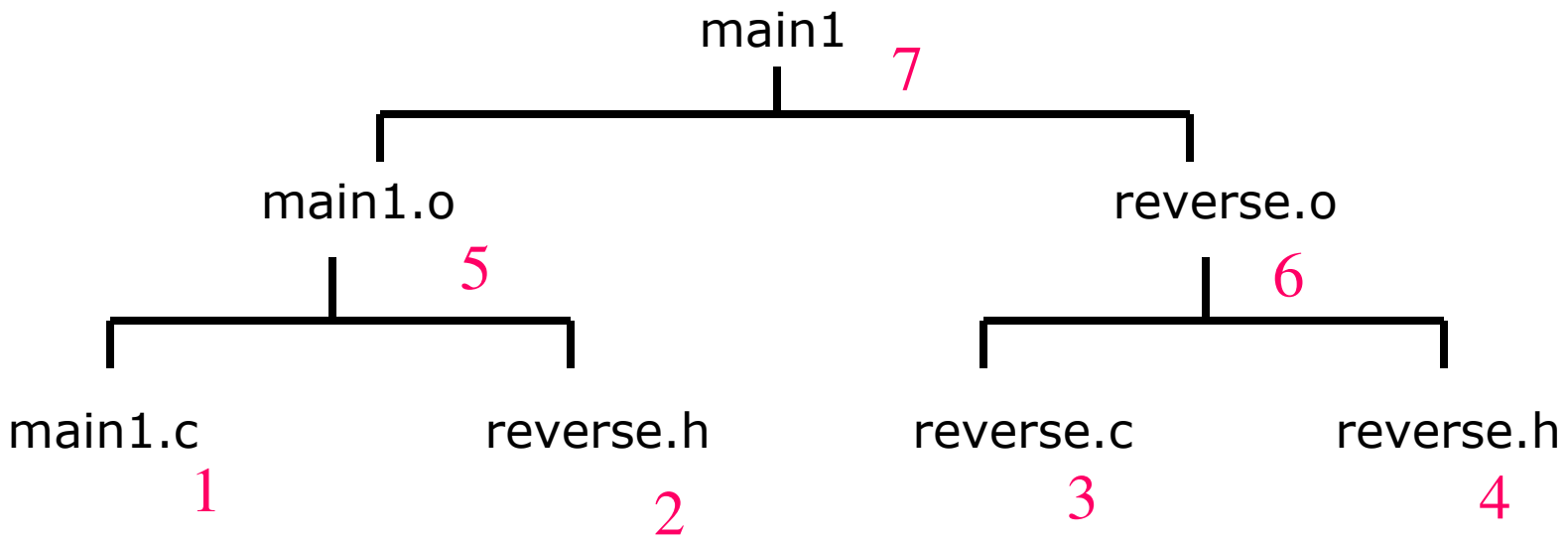
- **The Order of Make Rules**

  - In example, the final tree



make: File Dependency System

- **Make Ordering**



make: File Dependency System

- **Executing a Make**
  - Once a make file has been created, the make process is set into action by using the make utility as follows:

```
make [ -f makeFileName ]
```

  - To re-create the executable file whose dependency information is stored in the file makeFileName.

If the "-f" option is omitted,
the default make-file name "makefile" is used.

```
$ make -f main1.make
cc  -c  main1.c
cc  -c  reverse.c
cc  main1.o  reverse.o   -o  main1
$ _
```

make: File Dependency System

- **A second make file, called "main2.make"**

```
main2:   main2.o   reverse.o     palindrome.o
         cc  main2.o  reverse.o  palindrome.o   -o  main2

main2.o:   main2.c   palindrome.h
         cc -c main2.c

reverse.o:   reverse.c  reverse.h
         cc -c reverse.c

palindrome.o: palindrome.c  palindrome.h  reverse.h
         cc -c palindrome.c
```

```
$ make -f main2.make
cc  -c  main2.c
cc  -c  palindrome.c
cc  main2.o  reverse.o   palindrome.o   -o main2
$ _
```

make: File Dependency System

- **Make Rules**

  - Note that several of the rules are of the form

```
xxx.o :   reverse.c       reverse.h
          cc   -c   xxx.c
```

  where xxx varies between rules.

```
main2:          main2.o  reverse.o  palindrome.o
        cc  main2.o  reverse.o  palindrome.o  -o  main2

main2.o:        main2.c  palindrome.h

reverse.o:      reverse.c  reverse.h

palindrome.o:   palindrome.c  palindrome.h  reverse.h
```

make: File Dependency System

- **Make Rules**

  - a sleeker version of "main2.make"

```
main2:          main2.o   reverse.o    palindrome.o
                 cc main2.o  reverse.o  palindrome.o   -o   main2
main2.o:        palindrome.h
reverse.o:      reverse.h
palindrome.o:   palindrome.h   reverse.h
```

This makefile uses one of the build-in compilation rules:

```
.c.o:
        $(CC) $(CFLAGS) $(CPPFLAGS) -c $<
```

$< is a built-in variable whose value is the filename of the dependency (i.e., "main2", "reverse", "palindrome", and so on)

More Advanced Makefile

```
SUPPORT_DIR = /home/whatever
CC=gcc
CFALGS=-g -Wall -DDEBUG
CPPFLAGS = -I $(SUPPORT_DIR)
LDFLAGS  = -L $(SUPPORT_DIR)
LDLIBS   =                                          ---> none here

all:                    main1 main2
main1:                  main1.o   reverse.o
        $(CC) $(CFALGS) $(CPPFLAGS) main1.o   reverse.o   -o  main1  $(LDLIBS)
main1.o : main1.c reverse.h
        $(CC) $(CFALGS) $(CPPFLAGS) -c  main1.c
reverse.o :             reverse.c  reverse.h
        $(CC) $(CFALGS) $(CPPFLAGS) -c  reverse.c
main2:                  main2.o reverse.o palindrome.o
        $(CC) $(CFALGS) $(CPPFLAGS) main2.o reverse.o palindrome.o -o main2
main2.o:                palindrome.h
        reverse.o:      reverse.h
palindrome.o:           palindrome.h   reverse.h
lint :
        lint $(CPPFLAGS) *.c
clean:
        rm palindrome reverse *.o
```

make: File Dependency System

- **Touch**

- To confirm that the new version of the make file worked, we executed make and obtained the following output:

```
$ make  -f  main2.make
'main2' is up to date.
$ -
```

- make uses the last modification time of all of the named files
- compiles those that have time stamps less or equal to the current system time.

Utility : **touch** -c { fileName }+

touch updates the last modification and access times of the named files to the current time.

make: File Dependency System

- **Touch**

  - we touched the file "reverse.h",
    which subsequently caused the recompilation of several source files:

```
$ touch reverse.h
$ make -f main2.make
/bin/cc  -c  -O reverse.c
/bin/cc  -c  -O palindrome.c
cc  main2.o  reverse.o  palindrome.o   -o  main2
$ _
```

make: File Dependency System

- **Touch**

  - To recompile the suite of programs,
    we can use the touch utility to force recompilation of
    all of the source files:

```
$ touch *.c                    ---> force make to recompile everything.
$ make -f main2.make
/bin/cc  -c  -p  main2.c
/bin/cc  -c  -p  palindrome.c
/bin/cc  -c  -p  reverse.c
cc  -p  main2.o  reverse.o   palindrome.o   -o main2
$ _
```

Better Make Tools

- gmake
  - –GNU make, from free software foundation
  - –Support for parallel building
  - –More flexibility with macros
- nmake
  - –From AT&T and Bell Labs
  - –Uses state files instead of time stamps
  - –Dynamic dependency checking
    - Steps can alter dependency graph
    - Includes rules for C header files, several others