PRAM

PROGRAMMING OPERATING SYSTEMS

Module 1.7

Richard Newman

University of Florida

DEVELOPING OPERATING SYSTEMS

- Operating systems are some of the largest and most complex programs man has ever made
  - Millions to tens of millions of lines of code
- Need ways to
  - Manage complexity – structure
  - Manage cost of modification - builds
- Making changes
  - Patch files
  - Separate compilation and linking
  - Makefiles to manage dependencies

DEVELOPING OPERATING SYSTEMS

- Operating systems used to be written in assembly code
  - Need them to be very fast, efficient
  - Need access to low level hardware interfaces
  - Each new machine had its own instruction set
- Unix changed all that
  - C invented to allow efficient machine code
  - Unix mostly written in C (a little assembly as needed)
  - Port assembly part, then recompile for new architecture to get Unix on new hardware!

SEPARATE COMPILATION

- Compile logical units of sources code independently of each other
- Don't need to compile in other code that is used by a particular unit
- Only need to know what this unit needs to import from somewhere else
- Relocatable object code lists contents (what this unit provides) and what it needs to import
- Linker has to find all imported procedures, vars

MAKEFILES

- Name target file
- Lists dependencies
- Lists actions to make a new version of the target if any of the files on which the target depends have changes (and only if)
- With separate compilation, can reduce compile time dramatically!
- More on this in lab

LARGE PROGRAMMING PROJECTS

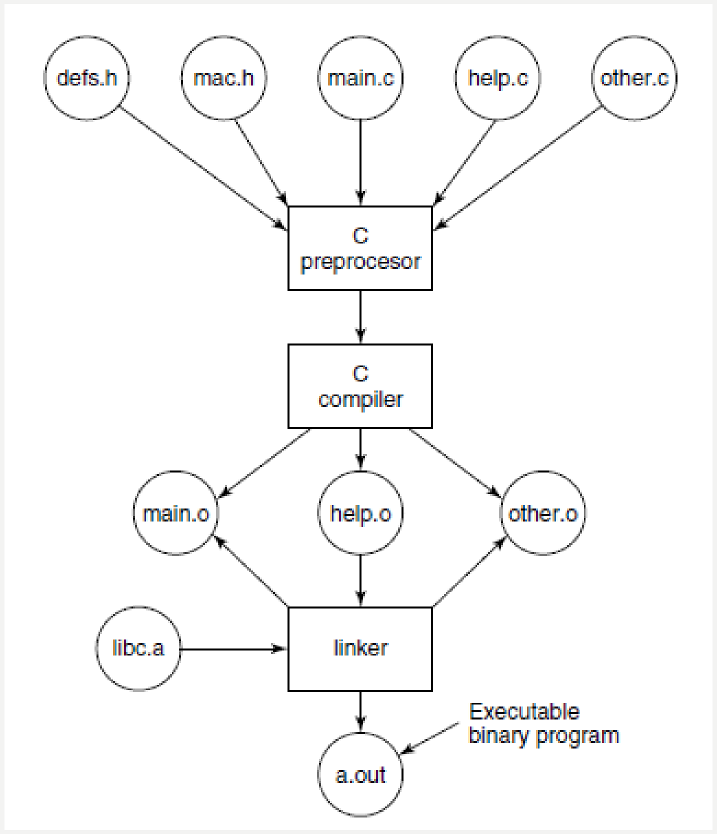

Figure 1-30. The process of compiling C and header files to make an executable.

LIBRARIES

- Combine one or more related procedures into one larger unit that can easily be searched by the linker
- Easy to make custom library
  - Use ar(1) archive utility to collect .o files and provide directory for linker
  - Use ranlib(1) to resolve internal dependencies
- Easy to use one
  - If ar produced libX.a, where X is the library name
  - Use -lX in compile command line
  - Include path with libpath variable or -L flag

LIBRARIES

- C compiler looks for standard C library and for library with API to system automatically
- Other libraries (e.g., math library libm.a) must be explicitly included
- Libraries come with .h header files
- System calls associated with system library by most programmers... BUT ...

… these are really library routines in user space that set up trap to invoke actual system call