CS 444 Operating Systems

Chapter 1 Introduction

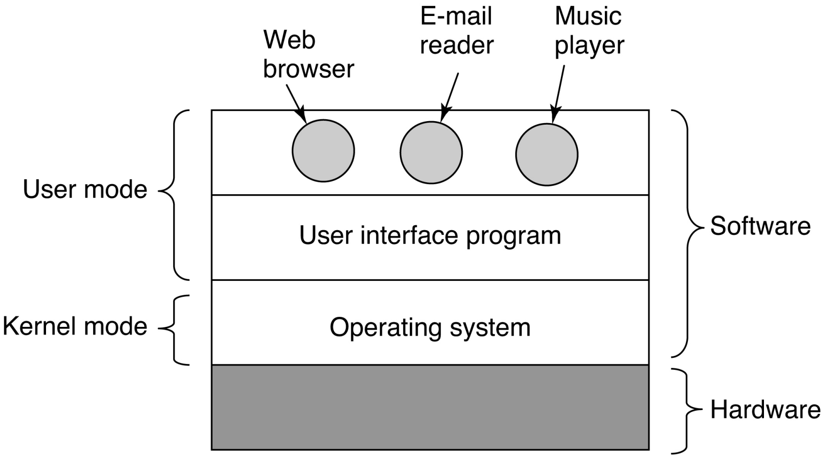
J. Holly DeBlois

September 1, 2024

Components of a Modern Computer

- One or more processors
- Main memory
- Disks
- Printers
- Keyboard
- Mouse
- Display
- Network interfaces
- I/O devices

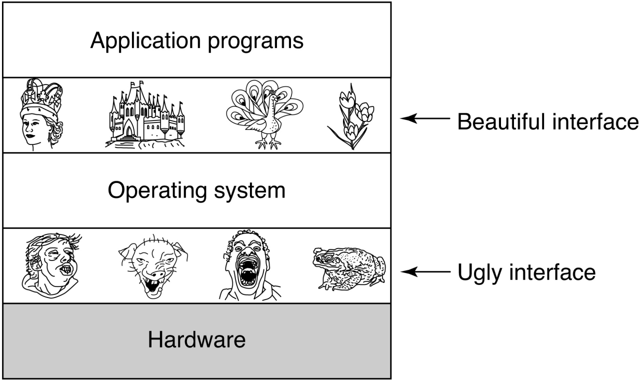
Figure 1-1 Where the operating system fits in



Two Views of Operating Systems

- An extended machine
- A resource manager

The Operating System as an Extended Machine



- SATA disk
- Disk driver

Figure 1-2 Operating systems turn ugly hardware into beautiful abstractions

The Operating System as a Resource Manager

- Top down view
  - Provide abstractions to application programs
- Bottom up view
  - Manage pieces of complex system
- Alternative view
  - Provide orderly, controlled allocation of resources

The Operating System as a Resource Manager

- The first generation (1945–55) vacuum tubes
- The second generation (1955–65) transistors and batch systems (mainframes)
- The third generation (1965–1980) ICs and multiprogramming
    - IBM 360, MULTICS, minicomputers, context switch
- The fourth generation (1980–present) personal computers
    - Microcomputers, CP/M, DOS
- The fifth generation (1990–present) mobile computers

Figure 1-3 An early batch system

- (a) Programmers bring cards to 1401
- (b) 1401 reads batch of jobs onto tape
- (c) Operator carries input tape to 7094
- (d) 7094 does computing
- (e) Operator carries output tape to 1401
- (f) 1401 prints output

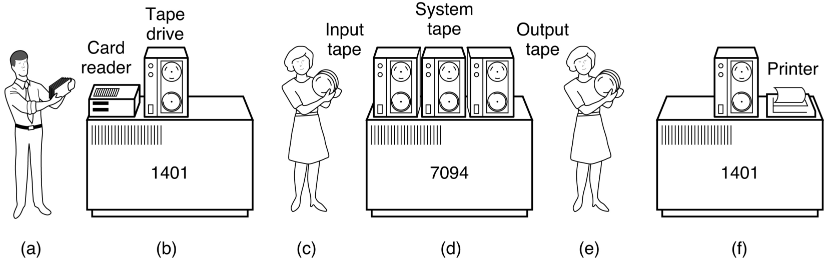
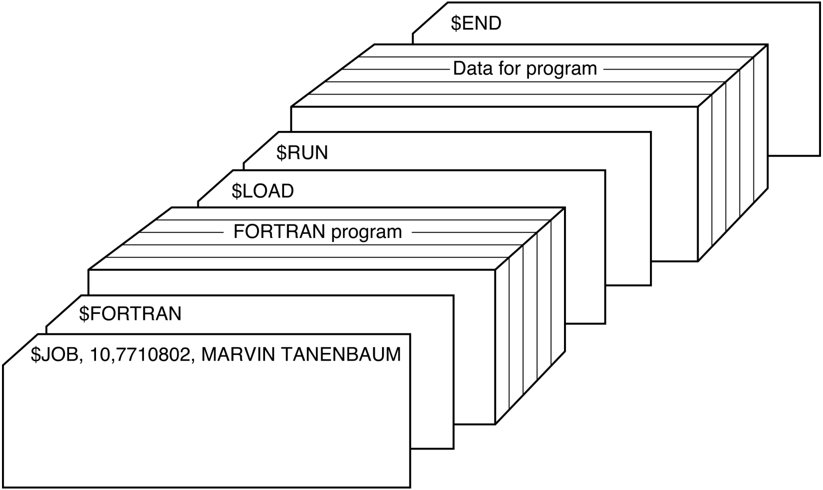
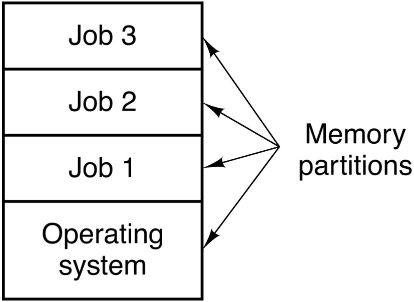


Figure 1-4 Structure of a typical FMS job



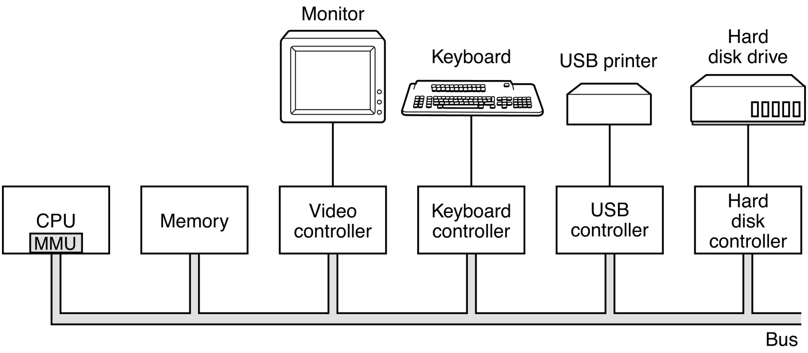
ICs and Multiprogramming

- Figure 1-5 A multiprogramming system with three jobs in memory



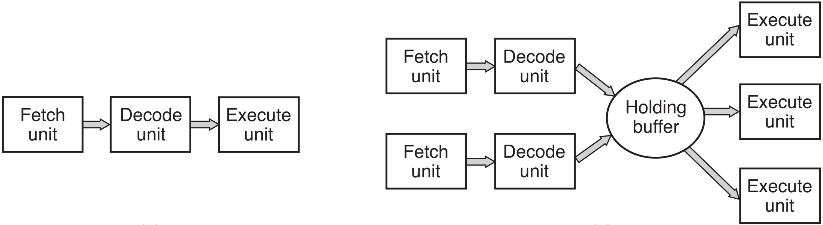
Hardware Overview

- Figure 1-6 Some of the components of a simple personal computer



Processors

- Program counter
- Stack pointer
- Program status word (PSW)
- Superscalar, multithreading (hyperthreading)
- Figure 1-7 (a) A three-stage pipeline (b) A superscalar CPU

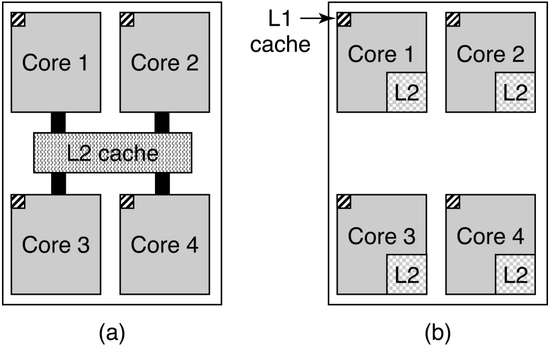


(a)

(b)

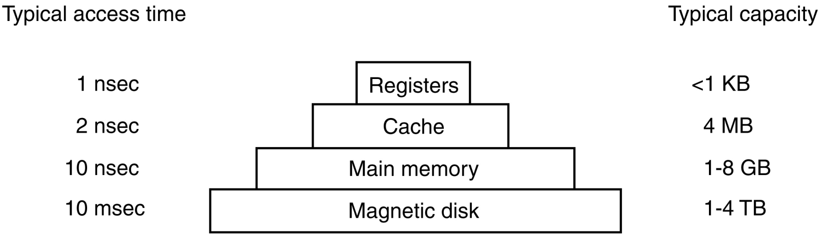
Memory

- Figure 1-8
- (a) A quad-core chip with a shared L2 cache, Intel
- (b) A quad-core chip with separate L2 caches, AMD



Memory

- Figure 1-9
- A typical memory hierarchy
- The numbers are very rough approximations



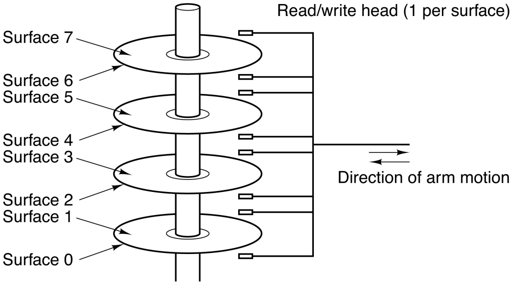
Typical access time		Typical capacity
1 nsec	Registers	<1 KB
2 nsec	Cache	4 MB
10 nsec	Main memory	1-8 GB
10 msec	Magnetic disk	1-4 TB

Caching System Issues

1. When to put a new item into the cache
2. Which cache line to put the new item in
3. Which item to remove from the cache when a slot is needed
4. Where to put a newly evicted item in the larger memory

Disks

- Figure 1-10 Structure of a disk drive



- Tracks, cylinders, sectors (historically 512B, nowadays 4KB)
- Linux commands to see sector and block sizes
- `fdisk -l /dev/sda`
- `blockdev --getbsz /dev/sda`

Device Drivers

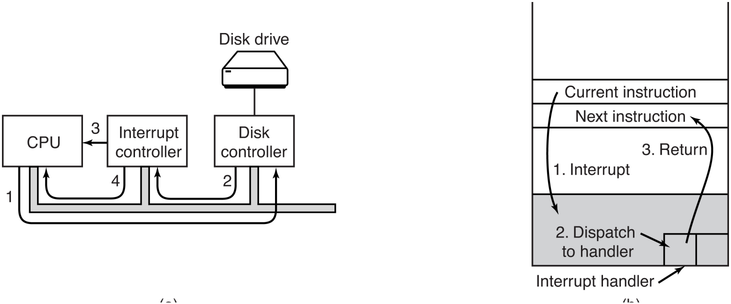
- Three ways to incorporate device drivers into the OS kernel
- Relink the kernel with new drivers, and then reboot
- Make a note in the OS file, reboot and load the drivers at boot up time (Windows)
- Dynamically loaded drivers (USB, IEEE 1394)

I/O Mechanisms

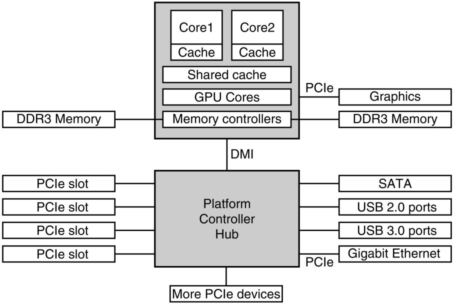
- Three ways to perform I/O
- Busy waiting
- Interrupt
- Direct memory access (DMA)

Interrupt I/O

- Figure 1-11
- (a) The steps in starting an I/O device and getting an interrupt
- (b) Interrupt processing involves taking the interrupt, running the interrupt handler, and returning to the user program
    - Save program counter, PSW
    - Jump to an index of the interrupt vector



Buses



- Figure 1-12 The structure of a large x86 system
- PCI
    - Shared bus architecture
    - Parallel bus architecture
- PCIe
    - Point-to-point connection
    - Serial bus architecture

Boot up the Computer

- BIOS
  - Stored in flash RAM — “flash the BIOS”
- Check a number of things
  - RAM
  - Basic devices
  - Scanning the buses
- Follow a prescribed list of devices to find an OS
  - CD, USB, hard drive

- After the OS is up
- Query BIOS for new devices
- Load drivers into kernel
- Initialize tables
- Start the first process
- Start background processes
- Start the console or a GUI

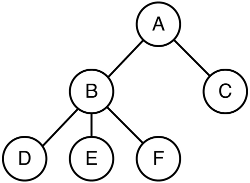
The Operating System Zoo

- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Handheld computer operating systems
- Embedded operating systems
- Sensor node operating systems
- Real-time operating systems
- Smart card operating systems

Processes

- Key concept in all operating systems
- Definition: a program in execution
- Process is associated with an address space
- Also associated with a set of resources
- A process can be thought of as a container — holds all information needed to run a program

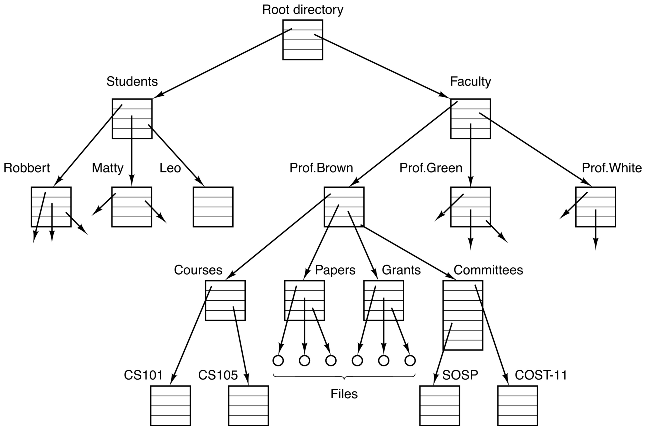
Process Tree



- Figure 1-13
  - A created two child processes — B, C
  - B created three child processes — D, E, F
- OS keeps a process table
- Core image: the address space of a process
- A user has a user ID
- Users are put into (overlapping) groups
- A group has a group ID
- One special UID
  - Unix: root
  - Windows: administrator

File Systems

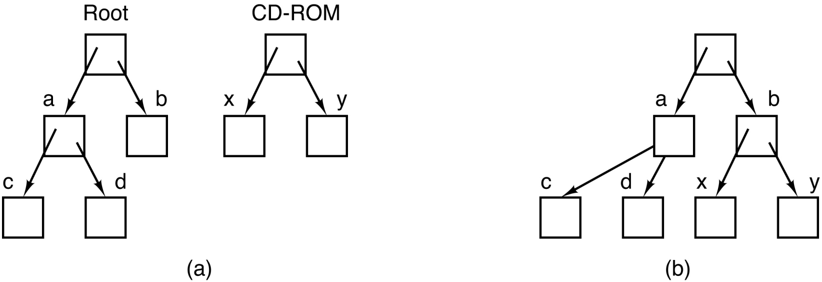
- Figure 1-14, a file system for a university department



- Root directory, working directory, path
- Protection, rwx bits

Mount an External Media

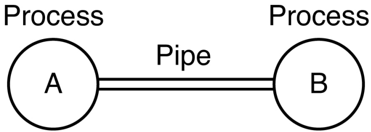
- Figure 1-15
  - (a) Before mounting, the files on the CD-ROM are not accessible
  - Mount point is directory b
  - (b) After mounting, they are part of the file hierarchy



- Linux commands: `mount`, `umount`

Pipe

- Figure 1-16, two processes connected by a pipe



- A pipe is a pair of files created by the parent process for two child processes
- The output of process A is sent to the first file, which is piped into the second file, which is sent to the input of process B
- Linux command: `df -h | grep home`

Ontogeny Recapitulates Phylogeny

- Each new "species" of computer goes through same development as "ancestors"
- Consequence of impermanence
  - Text often looks at "obsolete" concepts
  - Changes in technology may bring them back
- Happens with large memory, protection hardware, disks, virtual memory

Instruction Set Evolution

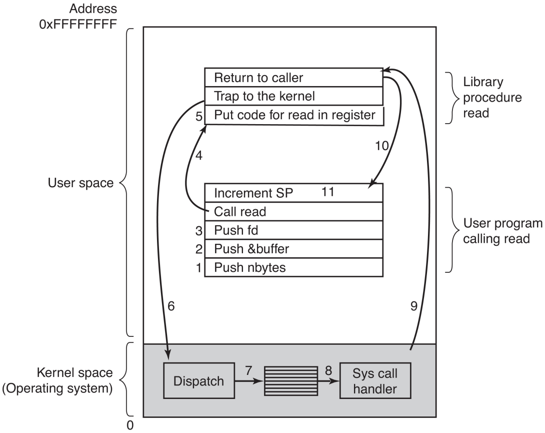
- Hardwired instruction sets
- Microprogramming (IBM 360), interpreted execution
- RISC
- Java applets

System Calls

- A user program runs in the user mode
- When it needs services from the OS, it makes “system calls” to get them
- A system call is implemented by a TRAP instruction to enter the kernel mode to run the kernel code of the OS

Steps of a System Call

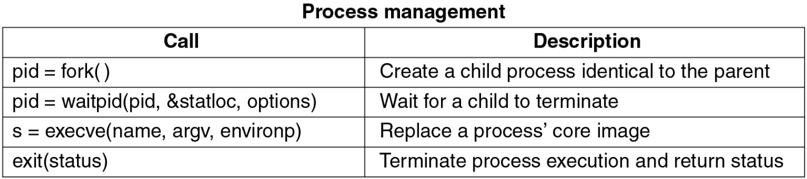
- Figure 1-17, the 11 steps in making the system call `read(fd, buffer, nbytes)`



- The return address is pushed into the stack by the CALL instruction

System Calls for Process Management

- Some of the major POSIX system calls



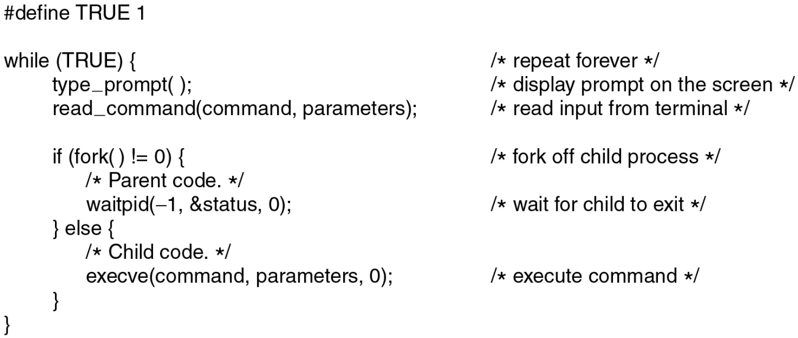
Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

- The return code s is 1 if an error has occurred
- The return code pid is a process id
- See the textbook for other categories of system calls

An Example of System Calls

- Figure 1-19, a stripped-down shell



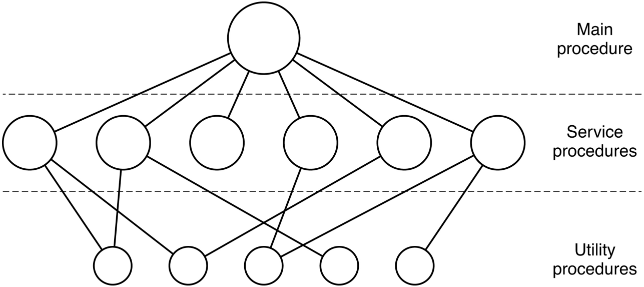
```
#define TRUE 1

while (TRUE) {                                   /* repeat forever */
     type_prompt( );                              /* display prompt on the screen */
     read_command(command, parameters);          /* read input from terminal */

     if (fork( ) != 0) {                          /* fork off child process */
         /* Parent code. */
         waitpid(−1, &status, 0);                 /* wait for child to exit */
     } else {
         /* Child code. */
         execve(command, parameters, 0);         /* execute command */
     }
}
```

Basic Structure of OS

- A main program that invokes the requested service procedure
- A set of service procedures that carry out the system calls
- A set of utility procedures that help the service procedures

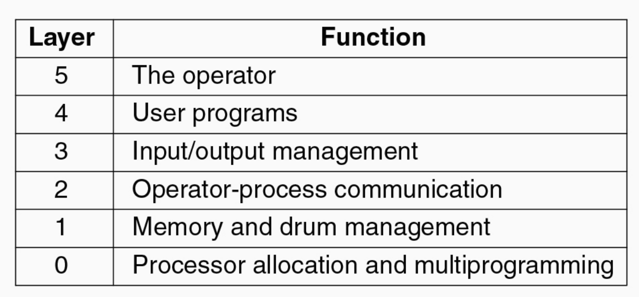
Monolithic Systems

- Figure 1-24, a simple structuring model for a monolithic system



Layered Systems

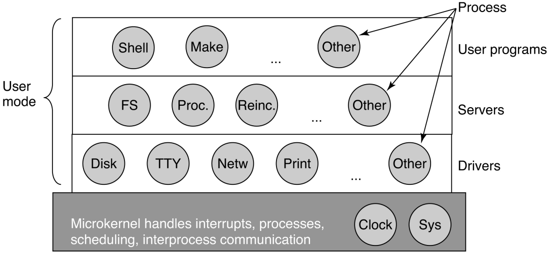
- Figure 1-25, structure of the THE operating system



Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Microkernels

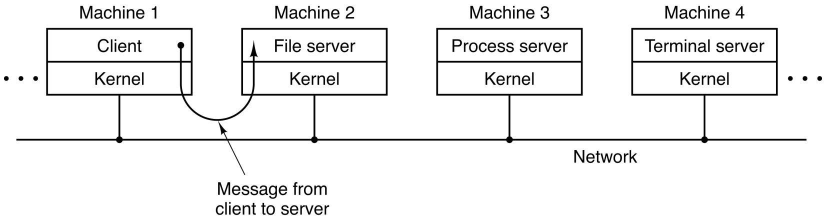
- Figure 1-26, simplified structure of the MINIX 3 system



- Minix 3 kernel has 5,000 lines of code
- Known statistics: 2 to 10 bugs per 1,000 lines of code

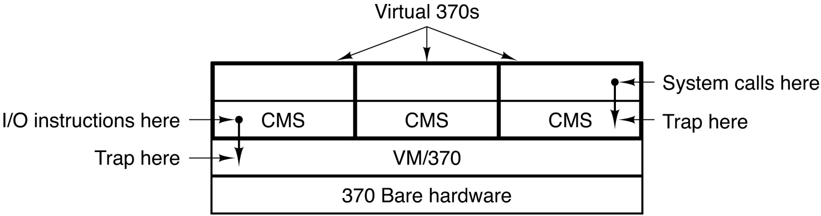
Client-Server Model

- Figure 1-27, the client-server model over a network



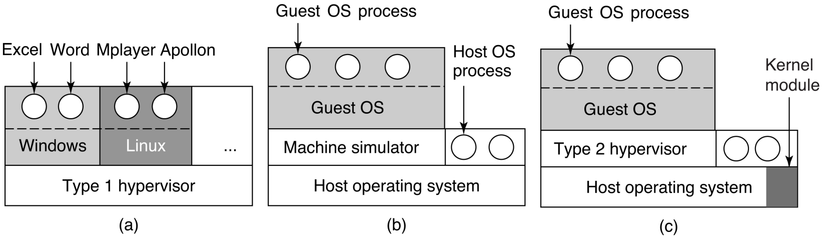
Virtual Machines

- Figure 1-28, the structure of VM/370 with CMS



Virtual Machines Rediscovered - see Popek and Goldberg, 1974, p71

- Figure 1-29
- (a) A type 1 hypervisor
- (b) A pure type 2 hypervisor
- (c) A practical type 2 hypervisor



- VMWare
- VirtualBox

The World of C

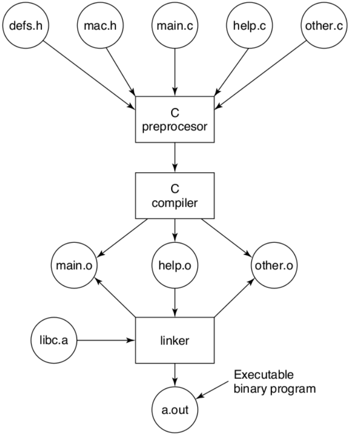
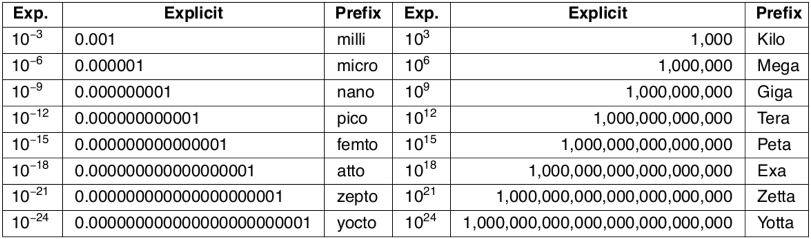


Figure 1-30. The process of compiling C and header files to make an executable.

Metric Units - Memory is different!

- units for measuring memory sizes have slightly different meanings, p80



Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Figure 1-31. The principal metric prefixes.