Operating System Principles

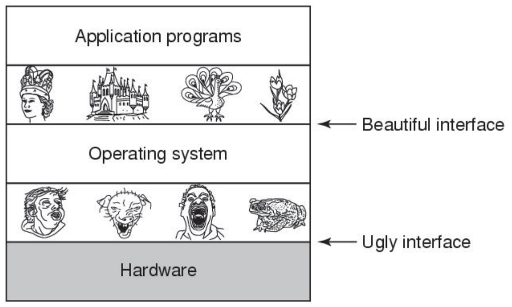
操作系统原理

Introduction

李旭东
leexudong@nankai.edu.cn
—Nankai Univ. SE.

What's an Operating System?

- 1.The Operating System as an Extended Machine



Objectives

- Operating System
- Operating System Functions
- Operating System Characters
- Operating System Structure
- Research on OS

What's an Operating System?

- http://en.wikipedia.org/wiki/Operating_system
- software that manages computer hardware and software resources and provides common services for computer programs
- an essential component of the system software in a computer system
- Application programs usually require an operating system to function

"Operating System"

Basic Services of OS

- Program Creation
- Program Execution
- Access to I/O Devices
- Controlled Access to Files
- System Access
- Error Detection and Response
- Accounting

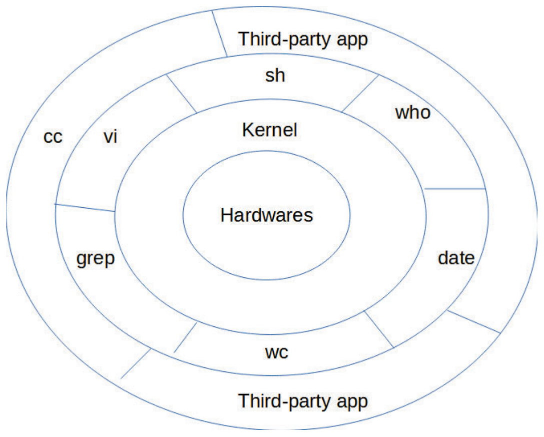
Evolution of An OS

- Maximization of resource utilization
- Hardware upgrades plus new types of hardware
- New Services
- Fixes
- User Experience

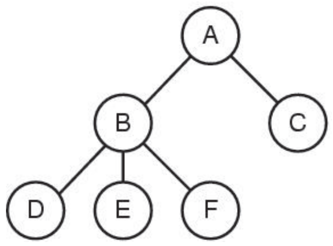
OS Basic Concepts

- Processes
- Address spaces
- Files
- Input/Output
- Protection
- The shell
- System Call

Architecture of UNIX Systems



Processes

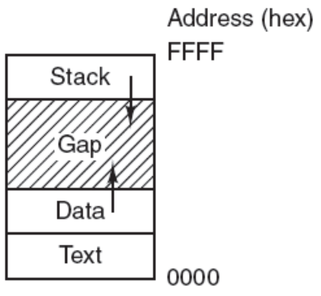


A process tree.
Process A created two child processes, B and C. Process B created three child processes, D, E, and F.

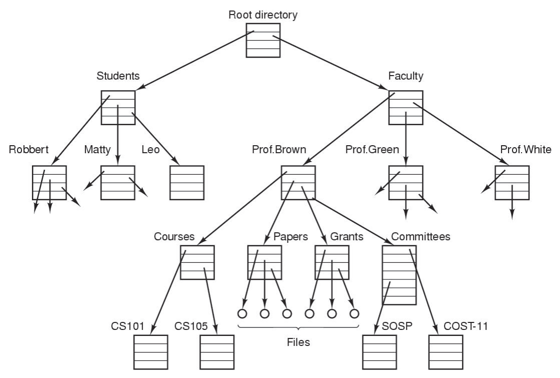
Basic Concepts of OS

Address Spaces

- 8 bits, 16 bits
- 32 bits, 64 bits
- Physical memory
- Virtual Memory



Files



A Simple Shell

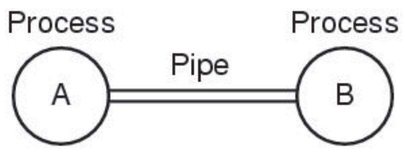
```
#define TRUE 1

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 */
      }
}
```

Files

- Files and Directories
  - Root directory,working directory
  - Path name:/,\
  - File hierarchies are organized as tree
  - File system: root file system
  - Special file
    - block special files 、 character files
  - File descriptor
  - Mount, umount

Input/Output

- I/O Subsystem
- IPC: Pipe

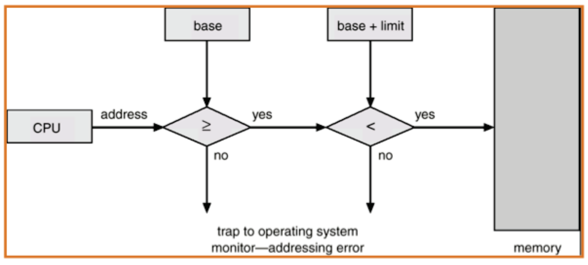


Shell

- shell
  - Command interpreter:shell
  - Prompt
    - >,#,$
  - Execute Commands:
    - #cat file1 file2 file3 l sort >/dev/lp &
  - Environment variables:
    - $#,$*,$?,$HOME,$PATH,$PS1

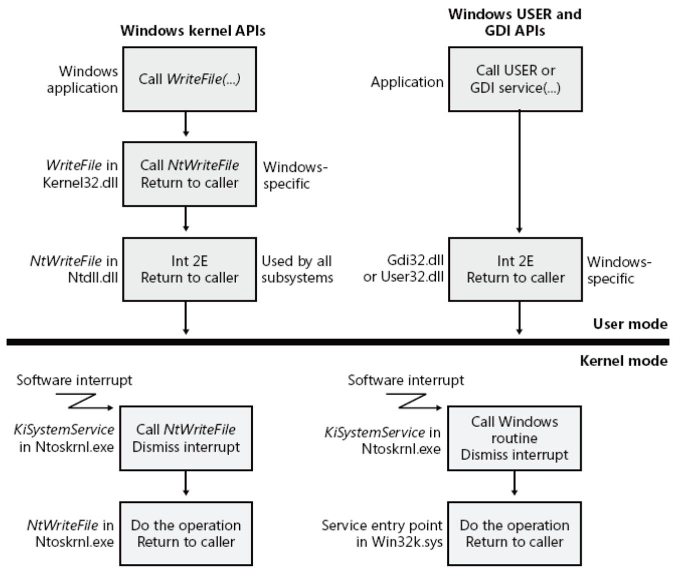
Protection

- Hardware
- Software

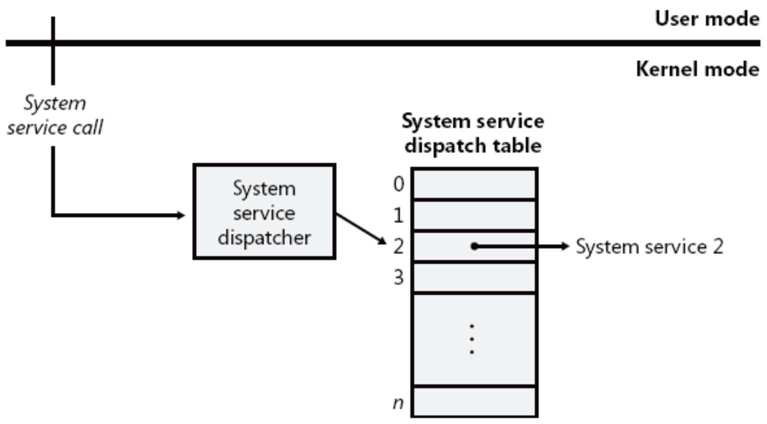


System Call

- System call
  - The interface between user programs and the operating system
  - Executed in kernel mode
  - Computer system running state
    - supervisor mode,kernel mode
    - user mode
- Trap Instruction
  - User mode to kernel mode
- Library Procedure
  - Encapsulates the trap instruction
  - Executed in user mode

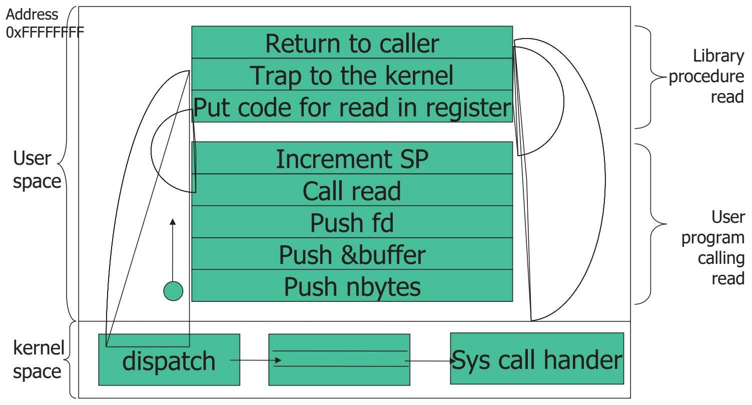

Windows kernel APIs
Windows application
Call WriteFile(...)
WriteFile in Kernel32.dll
Call NtWriteFile Return to caller
Windows-specific
NtWriteFile in Ntdll.dll
Int 2E Return to caller
Used by all subsystems
Windows USER and GDI APIs
Application
Call USER or GDI service(...)
Gdi32.dll or User32.dll
Int 2E Return to caller
Windows-specific
User mode
Kernel mode
Software interrupt
KiSystemService in Ntoskrnl.exe
Call NtWriteFile Dismiss interrupt
NtWriteFile in Ntoskrnl.exe
Do the operation Return to caller
Software interrupt
KiSystemService in Ntoskrnl.exe
Call Windows routine Dismiss interrupt
Service entry point in Win32k.sys
Do the operation Return to caller

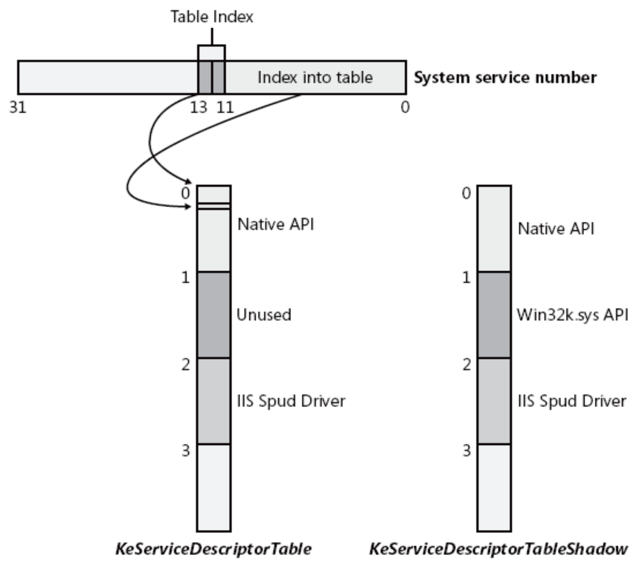
System Call



System call Case: read

- read(fd,buffer,nbytes)





Implementation of trap

- On x86 processors prior to the Pentium II
  - **int 0x2e**
- On x86 Pentium II processors and higher
  - Windows uses the special **sysenter** instruction
- On K6 and higher 32-bit AMD processors
  - Windows uses the special **syscall** instruction
- Case

```
NtWriteFile:
  mov  eax, 0x0E   ;system service number
  mov  ebx,esp     ;point to parameters
  int  0x2E         ;system service trap
  ret  0x2C        ;pop parameters off stack
                   ;    and return to caller
```

System Call

- POSIX API
- Windows Win32 API
- …

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

System Call Cases

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

System Call Types

- Process control
  - Create process 、 Terminate process
  - Get process attributes 、 Set process attributes
- file manipulation
  - Create file,delete file,read,write
  - Get/set file attributes
- device management
  - Request device,release device,read,write
- socket
  - Open connection, accept connection, read msg, write msg, close connection
- information maintenance
  - Getting current date, os version, etc.,

Quiz

- Which of the following several instructions should be executed only in kernel mode?
  - A. mask all interrupts
  - B. read current date
  - C. set current date
  - D. write the image core
  - E. read memory in user address space
  - F. halt

System Call Cases

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

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Ontogeny Recapitulates Phylogeny

- Dawrin, On the Origin of the Species
- The development of an embryo (ontogeny, 胚胎) **repeats** the evolution of the species (phylogeny)
  - Large Memories
  - Protection Hardware
  - Disk
  - Virtual Memory

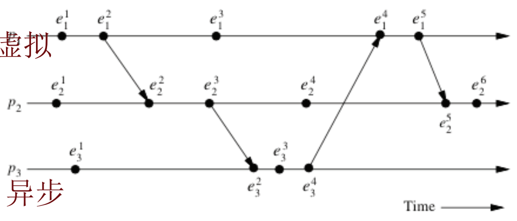
Functions of OS

- Process Management
- Memory Management
- Device Management
- File System Management
- User Interface
  - CLI
  - GUI
  - API
- Job Management

Characters of OS

OS Characters

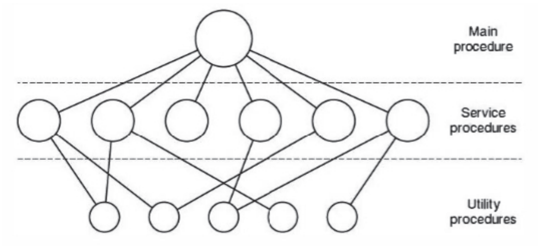
- Concurrency 并发
  - Concurrency: Logical concurrency
  - Parallel: Physical concurrency
- Share 共享
  - CPU, Main Memory, Storage, I/O Devices
  - Space, Time
- Virtualization 虚拟
  - 1 to N
  - N to 1
  - 0 to N
- Asynchronism 异步



OS Runtime Structure

OS Runtime Structure

- Monolithic Systems
  - 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.



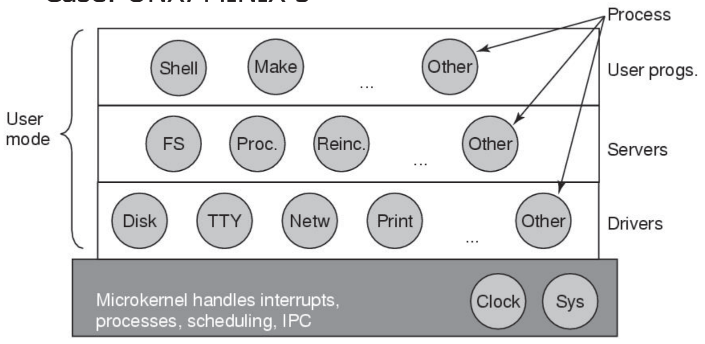
OS Runtime Structure

- Layered Systems
  - Case: THE

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

OS Runtime Structure

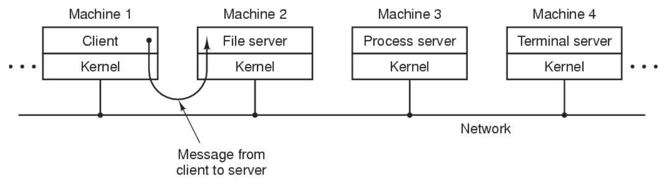
- Microkernels 微内核
  - Case: ONX, MINIX 3



Structure of the MINIX 3 system.

OS Runtime Structure

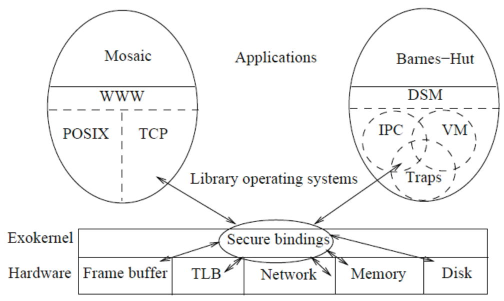
- Client-Server Model
  - Communication between clients and servers is often by message passing



The client-server model over a network.

OS Runtime Structure

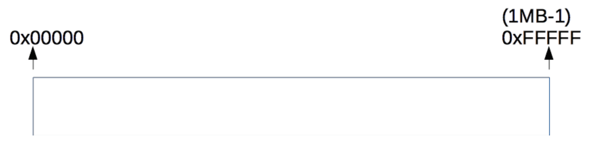
- Exokernels 外核
  - Partitioning the actual machine, rather than cloning the actual machine
  - developed by the MIT Parallel and Distributed Operating Systems group
  - Hardware
  - ExoKernel
  - Library OSes
  - Applications



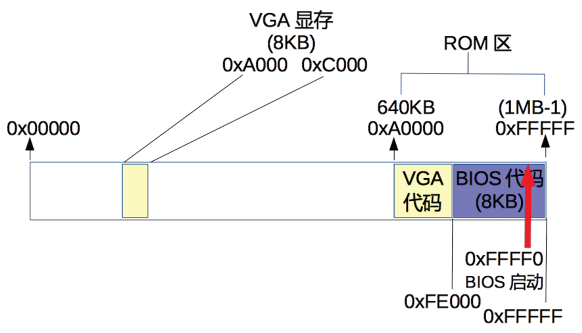
Engler D R , Kaashoek F M , Jr J O . Exokernel: an operating system architecture for application-level resource management[J]. 1995.

Booting The Computer

Memory Layout 1



Memory Layout 2



Booting A Computer

- an IBM-compatible personal computer's x86 CPU executes
  - Power On, real mode
    - the instruction located at reset vector (the physical memory address FFFF0h on 16-bit x86 processors and FFFFFFF0h on 32-bit and 64-bit x86 processors, i.e. BIOS entry point
  - BIOS: POST, power-on self-test
  - BIOS: goes through a pre-configured list of non-volatile storage devices ("boot device sequence") until it finds one that is bootable
  - BIOS:load the bootstrap (i.e. MBR, Master Boot Record) from bootable storage device
  - MBR:load the OS Kernel
  - OS Kernel: OS services and shell

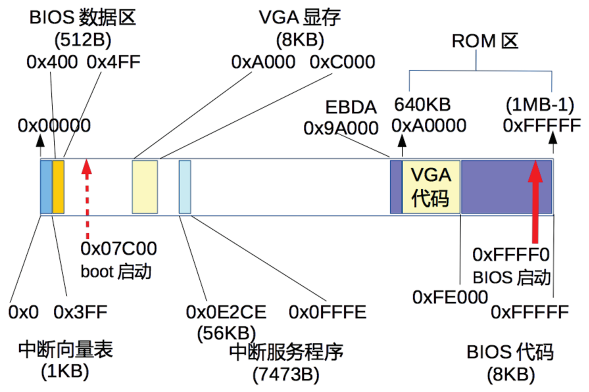
Booting The Computer

- Harddisk Partition table

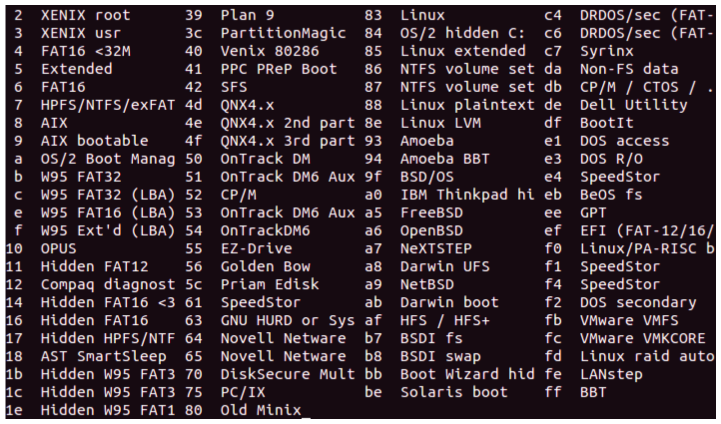
	00h	01h	02h03h		04h	05h	06h07h		08h~0Bh	0Ch~0Fh
01BEh	BI	Hs	Ss	Cs	SI	He	Se	Ce	HS	N
01CEh	BI	Hs	Ss	Cs	SI	He	Se	Ce	HS	N
01DEh	BI	Hs	Ss	Cs	SI	He	Se	Ce	HS	N
01EEh	BI	Hs	Ss	Cs	SI	He	Se	Ce	HS	N

SI:
00h undefined,01h Dos (12bit),02h XENIX,
04hDos (16bits),
05h extended partition,
06h Dos (32bits)

Memory Layout 3



File System Types



```
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix
```

Booting The Computer

- bootstrap

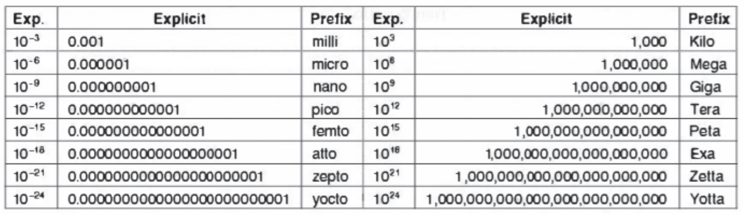
0x000~0x002 <A jump instruction to 0xttt>

0x003~… Disk parameters(used by BIOS)

0xttt~0x1fd Bootstrap program

0x1ff~0x1fe 0xaa55

Metric Unit



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}	0.0000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.0000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.0000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Research On OS

- Computer Science
- Internet
- GUI: Doug Engelbart
- Hot topics
  - Security, engergy, recovery, virtualization, fs, multicore,...
- ACM
  - www.acm.org
  - sigops
- IEEE Computer Society
  - www.computer.org
- USENIX
  - www.usenix.org

OSDI 20

SOSP '19

Summary

- Operating System
- Operating System Functions
- Operating System Characters
- Operating System Structure
- Research on OS

Q&A?