Sheets for MIEIC's SOPE

based on teaching material supplied by
A. Tanenbaum for book:
Modern Operating Systems, ed...

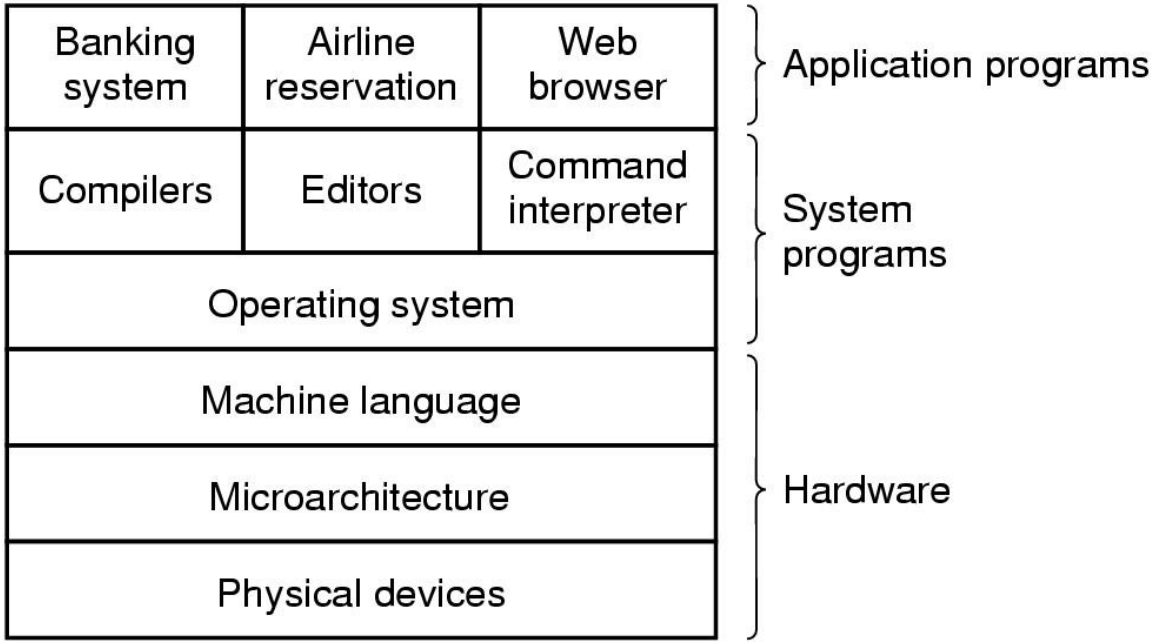
Chap 1:
Introduction to Operating Systems

Chapter 1

Introduction

What is an operating system
Computer hardware review
Operating system concepts
System calls

Introduction



A computer system consists of

- hardware
- system programs
- application programs

What is an Operating System

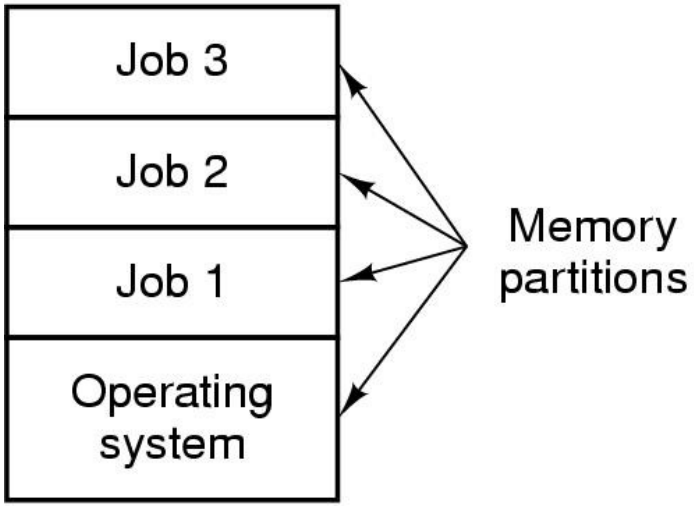
It is an extended machine

- Hides the messy details which must be performed
- Presents user with a virtual machine, easier to use

It is a resource manager

- Each program gets time with the resource
- Each program gets space on the resource

Operating System as resource manager



Multiprogramming system

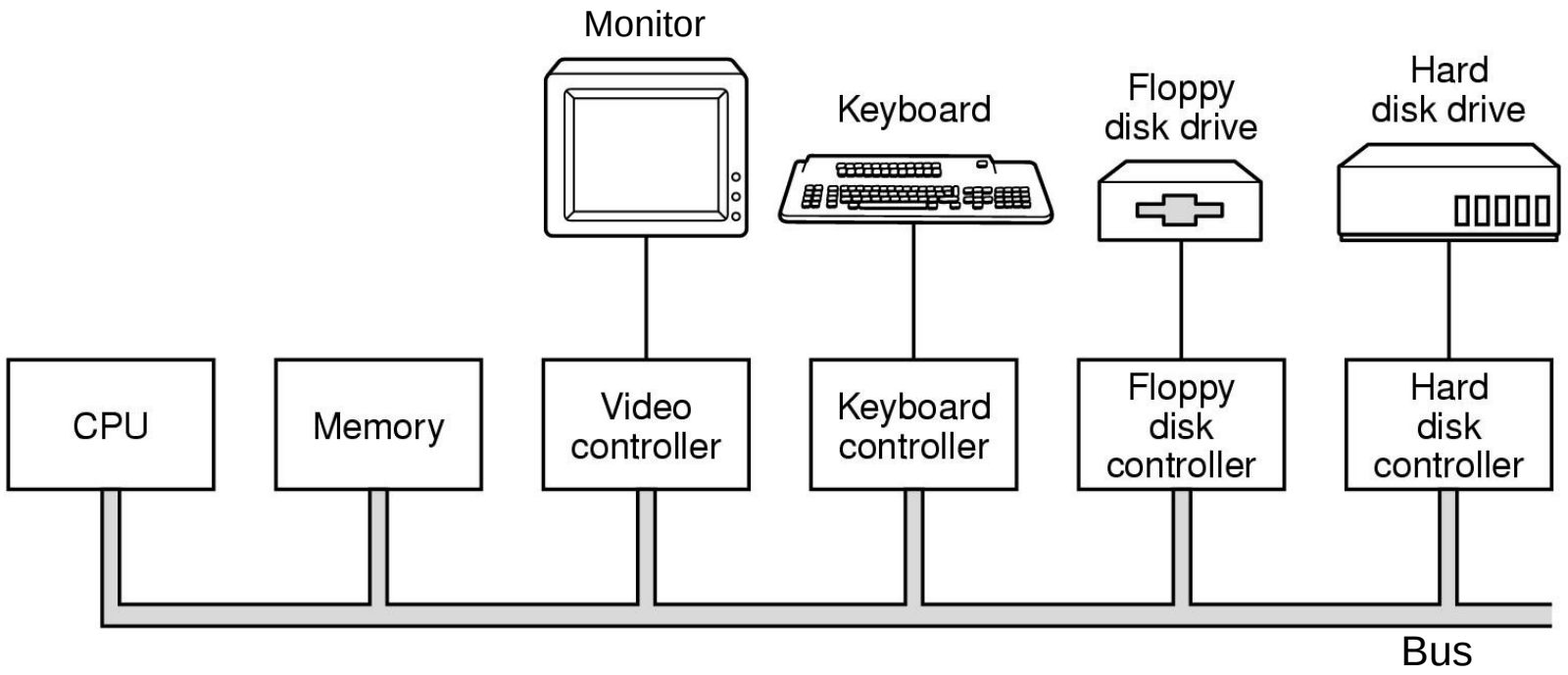
several jobs in memory (e.g. 3 jobs in picture)

Time sharing system

more than one user working on-line

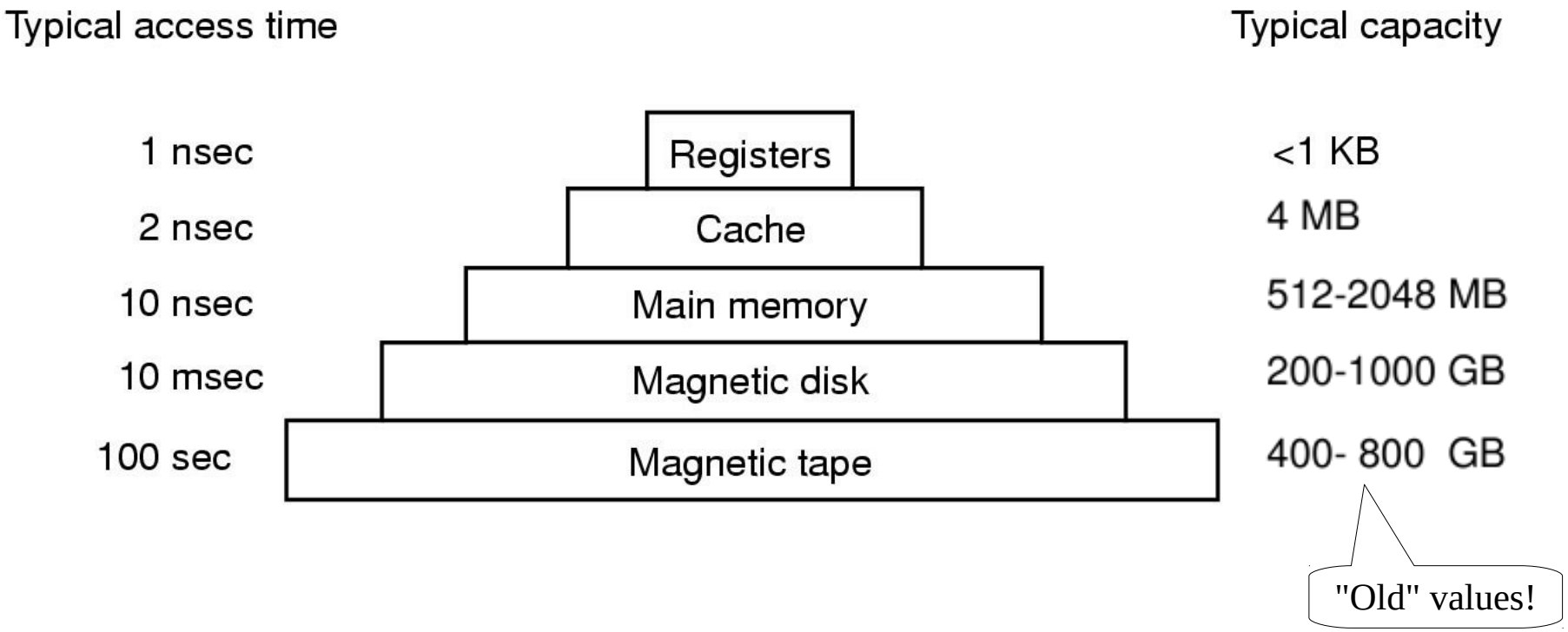
implies *multiprogramming* & *multiuser*

Computer hardware: main elements



Components of a simple (old!) personal computer

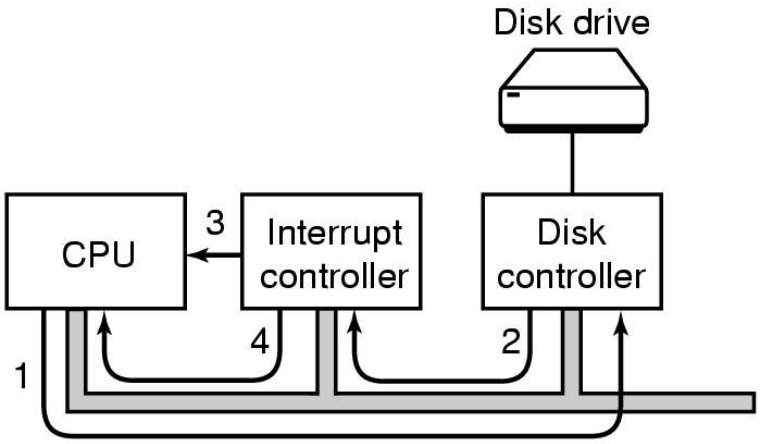
Computer hardware: memory



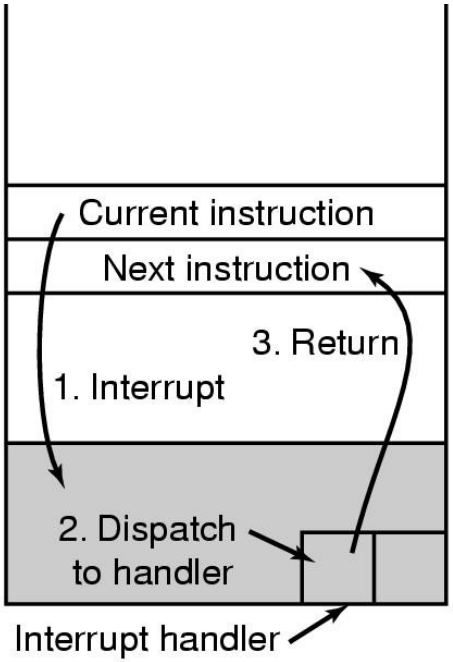
Typical memory hierarchy

numbers shown are rough (obsolete?) approximations

Computer hardware: interrupts



(a)



(b)

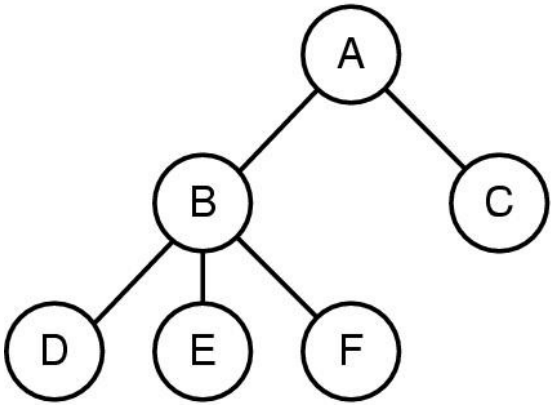
(a) Steps in starting an I/O device and getting interrupt

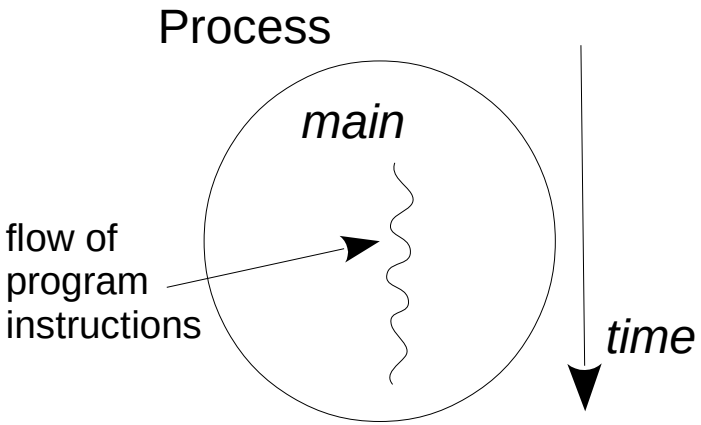
(b) How the CPU is interrupted

Operating System: processes

Process:

program running



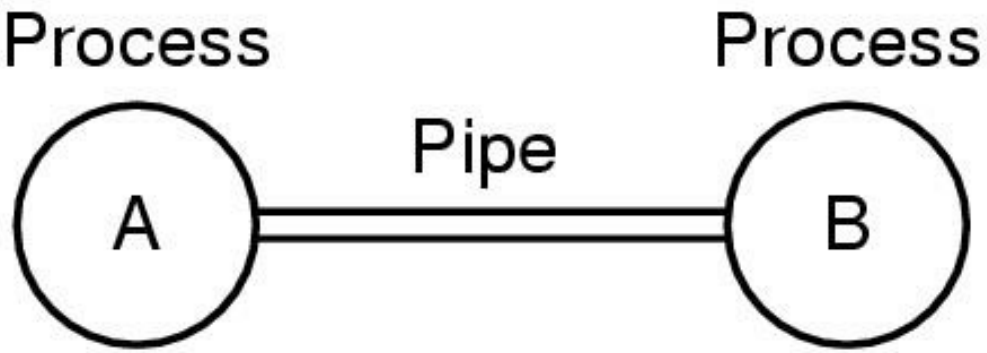


Process tree

A created two child processes, B and C

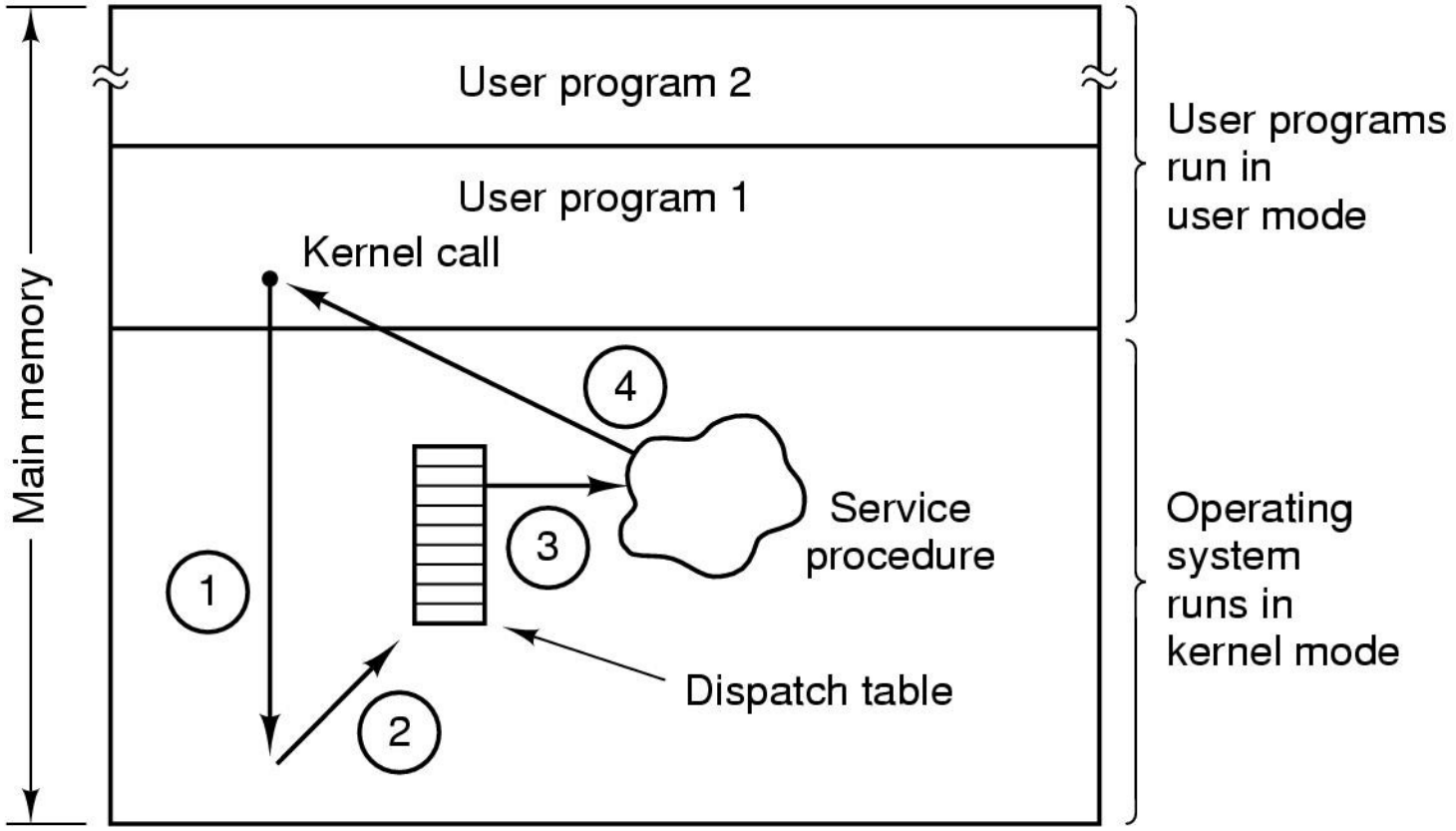
B created three child processes, D, E, and F

Operating System: process communication



Two processes connected by a pipe

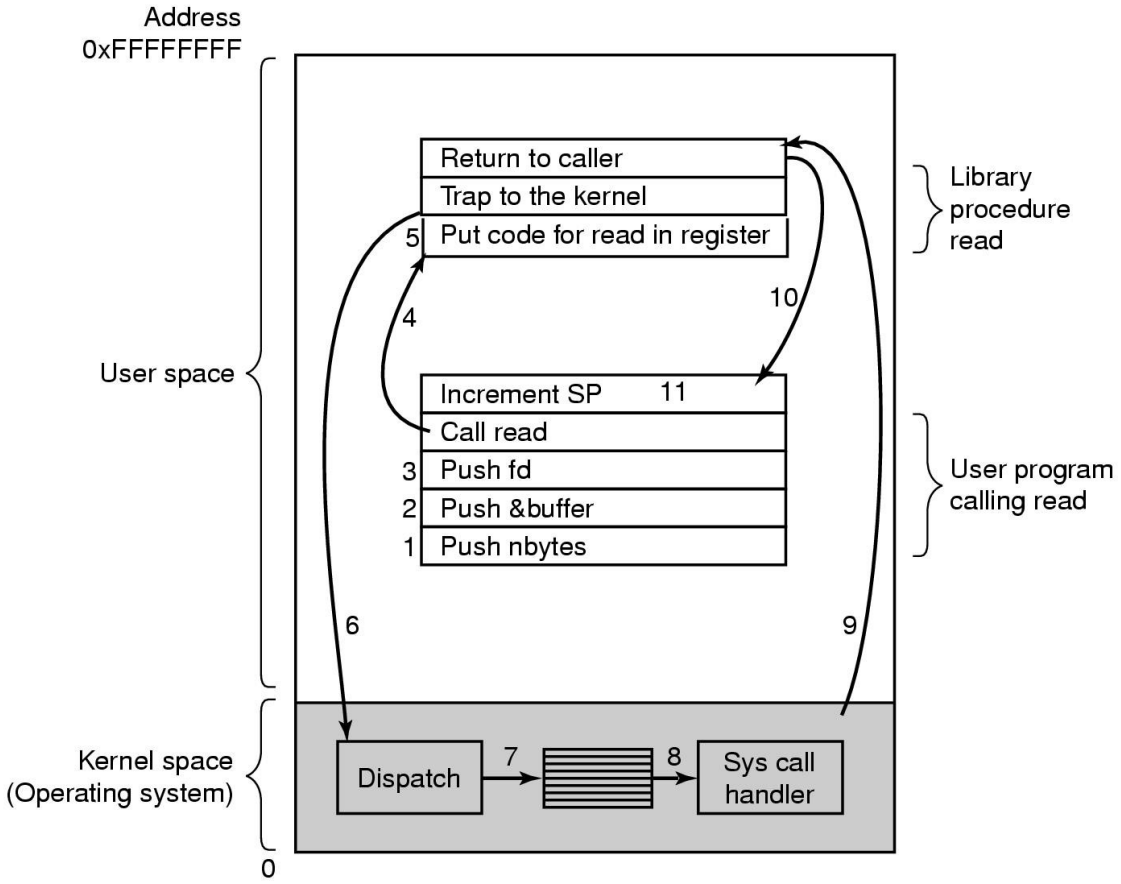
Operating System: system call



How a system call is made

(protection domain perspective)

Steps in Making a System Call



11 steps in making the system call read (fd, buffer, nbytes)

See class exercises!

Some System Calls

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

Miscellaneous

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

Annex: syscall example: read (fd, buffer, nbytes)

- syscall e.g. : read (fd, buffer, nbytes)
  - main () { int n = read (0x20, 0x30, 0x40); }
  - gcc -static ...
  - objdump -d ... > dumpfile [big...]
  - dumpfile : <main> ... <__libc_read>
  - /usr/include/x86_64-linux-gnu/asm/unistd_64.h

Annex: read syscall (2)

- x86_64

```
0000000000400b6d <main>:
  400b6d:  55                 push  %rbp
  400b6e:  48 89 e5           mov   %rsp,%rbp
  400b71:  48 83 ec 10        sub   $0x10,%rsp
  400b75:  ba 40 00 00 00     mov   $0x40,%edx
  400b7a:  be 30 00 00 00     mov   $0x30,%esi
  400b7f:  bf 20 00 00 00     mov   $0x20,%edi
  400b84:  b8 00 00 00 00     mov   $0x0,%eax
  400b89:  e8 22 7f 04 00     callq 448ab0 <__libc_read>
  400b8e:  89 45 fc           mov   %eax,-0x4(%rbp)
  400b91:  b8 00 00 00 00     mov   $0x0,%eax
  400b96:  c9                 leaveq
  400b97:  c3                 retq
...
```

```
0000000000448ab0 <__libc_read>:
...
  448aba: 31 c0     xor %eax,%eax
  448abc: 0f 05     syscall
...
```

```
#ifndef _ASM_X86_UNISTD_64_H
#define _ASM_X86_UNISTD_64_H 1

#define __NR_read  0
#define __NR_write 1
...
```

Annex: read syscall (3)

- i386

```
080481d0 <main>:
 80481d0: 55                push   %ebp
 80481d1: 89 e5             mov    %esp,%ebp
 80481d3: 83 ec 18          sub    $0x18,%esp
 80481d6: 83 c4 fc          add    $0xfffffffc,%esp
 80481d9: 6a 40             push   $0x40
 80481db: 6a 30             push   $0x30
 80481dd: 6a 20             push   $0x20
 80481df: e8 6c 4b 00 00    call   804cd50 <__libc_read>
 80481e4: 83 c4 10          add    $0x10,%esp
 80481e7: 89 c0             mov    %eax,%eax
 80481e9: 89 45 fc          mov    %eax,0xfffffffc(%ebp)
 80481ec: 89 ec             mov    %ebp,%esp
 80481ee: 5d                pop    %ebp
 80481ef: c3                ret
...
```

```
0804cd50 <__libc_read>:
 804cd50: 53                   push %ebx
 804cd51: 8b 54 24 10          mov  0x10(%esp,1),%edx
 804cd55: 8b 4c 24 0c          mov  0xc(%esp,1),%ecx
 804cd59: 8b 5c 24 08          mov  0x8(%esp,1),%ebx
 804cd5d: b8 03 00 00 00       mov  $0x3,%eax
 804cd62: cd 80                int  $0x80
 804cd64: 5b                   pop  %ebx
 804cd65: 3d 01 f0 ff ff       cmp  $0xfffff001,%eax
 804cd6a: 0f 83 a0 03 00 00    jae  804d110
                          <__syscall_error>
 804cd70: c3                   ret
```

```
#ifndef _ASM_I386_UNISTD_H_
#define _ASM_I386_UNISTD_H_

#define __NR_exit  1
#define __NR_fork  2
#define __NR_read  3
#define __NR_write 4
...
```