Operating Systems

(System structure)

Chapter 2

These lecture materials are modified from the lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

Spring, 2020

Outline

- Operating System Services
- System Calls
- OS Design & Implementation
- Operating System Structure
- System Boot

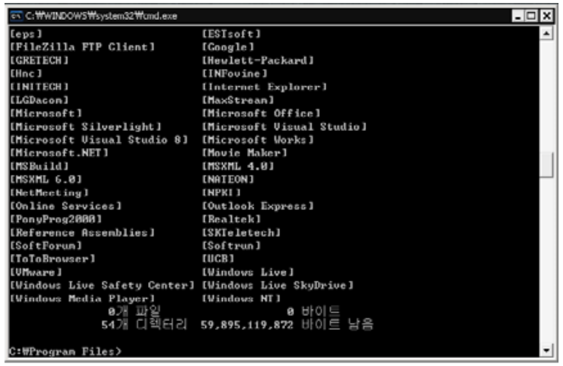
Operating system services

- For users
  - User interface
    - Command-line interface (CLI)
    - Batch interface
    - GUI
  - Program execution
    - Loading and execution
  - I/O operations
    - Providing a means to access a file or an I/O device.

- For users
  - File-system manipulation
    - read and write files and directories,
    - create, delete, search file,
    - list file Information, permission management
  - Communications
    - There are many circumstances in which one process may exchange information with another process
    - Shared memory or message passing
  - Error detection
    - For each type of error, OS should take the appropriate action to ensure consistency

- For efficiency
  - Resource allocation
    - Many types of resources
      - such as CPU cycles, main memory, file storage and I/O devices
    - And many jobs (processes)
      - So, they are fighting for the resources.
  - Accounting
    - To keep track of which users use how much and what kinds of computer resources
  - Protection
    - Ensuring that all access to system resources is controlled
  - Security
    - Outsiders requires user authentication, it extends to defending external I/O devices from invalid access attempts

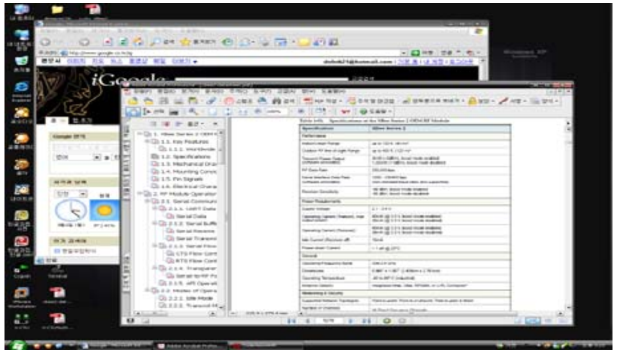
- User interface



C:₩WINDOWS₩system32₩cmd.exe

```
[eps]                        [ESTsoft]
[FileZilla FTP Client]       [Google]
[GRETECH]                    [Hewlett-Packard]
[Hnc]                        [INFovine]
[INITECH]                    [Internet Explorer]
[LGDacom]                    [MaxStream]
[Microsoft]                  [Microsoft Office]
[Microsoft Silverlight]      [Microsoft Visual Studio]
[Microsoft Visual Studio 8]  [Microsoft Works]
[Microsoft.NET]              [Movie Maker]
[MSBuild]                    [MSXML 4.0]
[MSXML 6.0]                  [NATEON]
[NetMeeting]                 [NPKI]
[Online Services]            [Outlook Express]
[PonyProg2000]               [Realtek]
[Reference Assemblies]       [SKTeletech]
[SoftForum]                  [Softrun]
[ToToBrowser]                [UCB]
[VMware]                     [Windows Live]
[Windows Live Safety Center] [Windows Live SkyDrive]
[Windows Media Player]       [Windows NT]
              0개 파일                 0 바이트
             54개 디렉터리  59,895,119,872 바이트 남음

C:₩Program Files>
```



Command line interface (CLI)

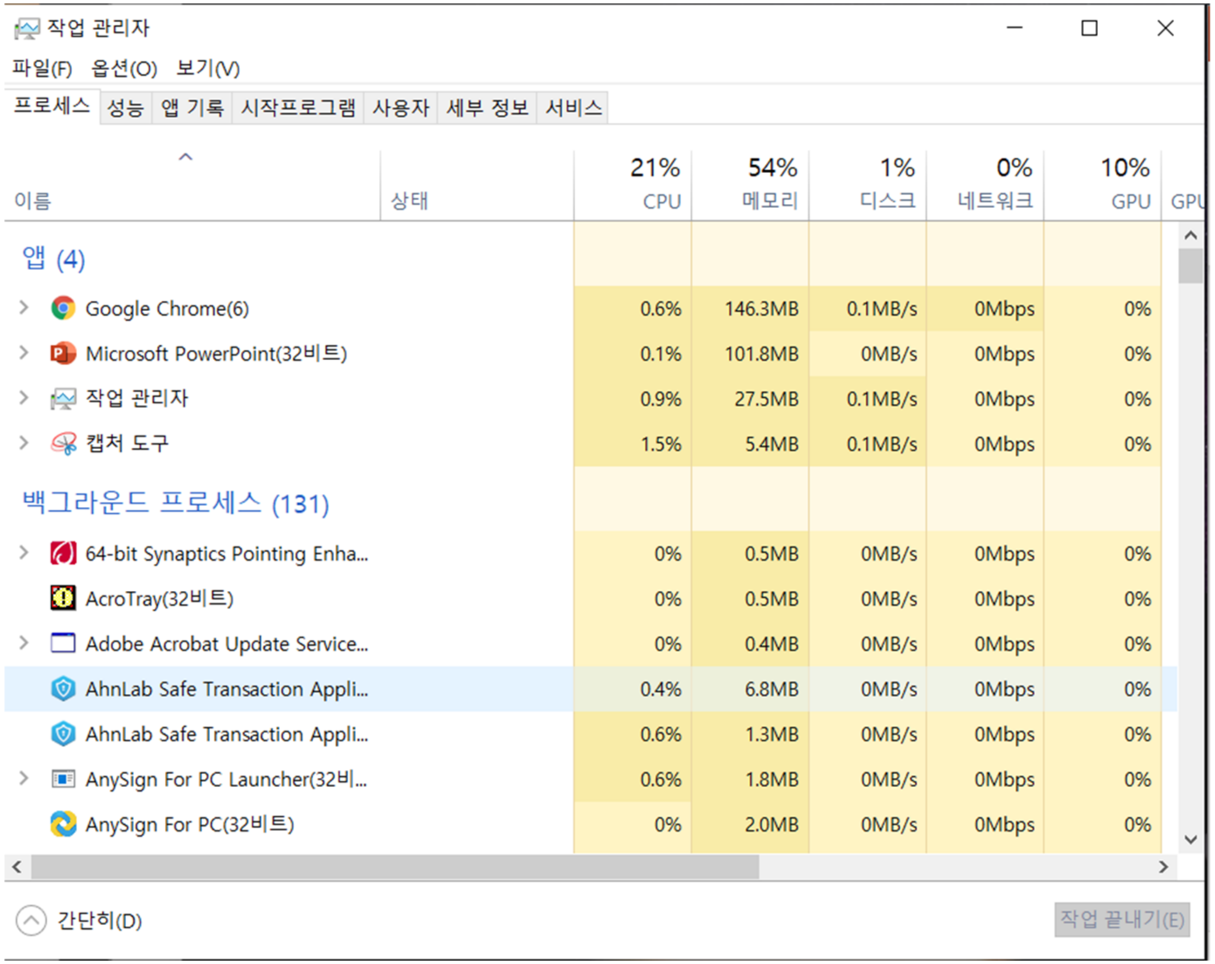
Graphical user interface (GUI)

Batch interface; commands are collected into files, and those files are executed

- User interface
  - Command Line Interface (CLI)
    - Allows users to directly enter commands that are to be performed by the operating system
    - Fetches a command from user and executes it
    - Often called shells
    - Two methods
      - 1. it contains the code to execute the command
      - 2. it searches an execution file and execute the file
        - e.g. rm fils.txt

- Graphical user interface (GUI)
  - Windows OS
  - UNIX or LINUX
    - Free software projects for the graphical user interface which runs on top of a computer operating system
      - 1. GNOME
      - 2. KDE

- Accounting

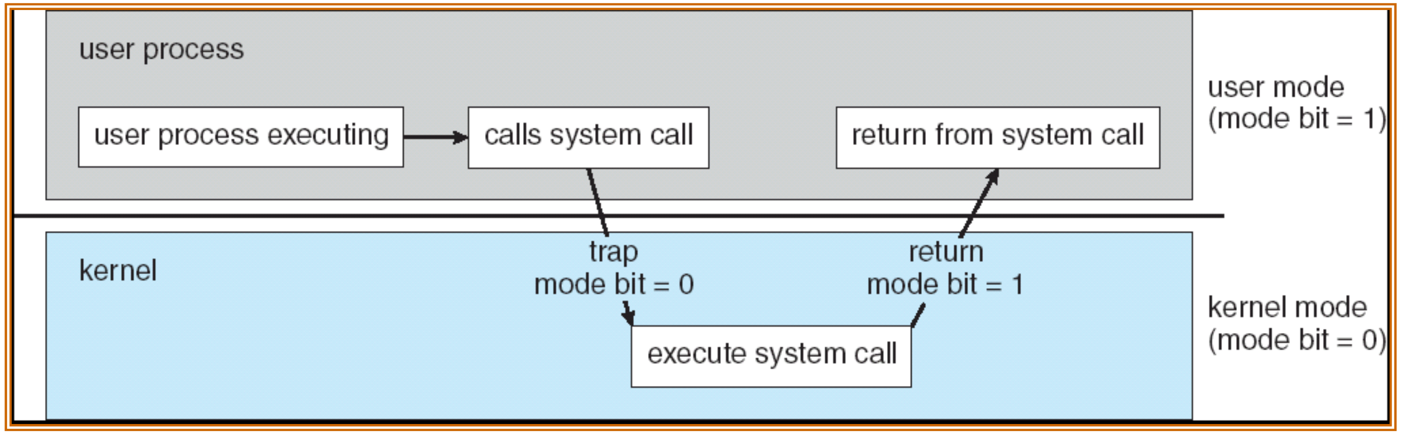


이름	상태	21% CPU	54% 메모리	1% 디스크	0% 네트워크	10% GPU	GPU
앱 (4)							
Google Chrome(6)		0.6%	146.3MB	0.1MB/s	0Mbps	0%	
Microsoft PowerPoint(32비트)		0.1%	101.8MB	0MB/s	0Mbps	0%	
작업 관리자		0.9%	27.5MB	0.1MB/s	0Mbps	0%	
캡처 도구		1.5%	5.4MB	0.1MB/s	0Mbps	0%	
백그라운드 프로세스 (131)							
64-bit Synaptics Pointing Enha...		0%	0.5MB	0MB/s	0Mbps	0%	
AcroTray(32비트)		0%	0.5MB	0MB/s	0Mbps	0%	
Adobe Acrobat Update Service...		0%	0.4MB	0MB/s	0Mbps	0%	
AhnLab Safe Transaction Appli...		0.4%	6.8MB	0MB/s	0Mbps	0%	
AhnLab Safe Transaction Appli...		0.6%	1.3MB	0MB/s	0Mbps	0%	
AnySign For PC Launcher(32비...		0.6%	1.8MB	0MB/s	0Mbps	0%	
AnySign For PC(32비트)		0%	2.0MB	0MB/s	0Mbps	0%	

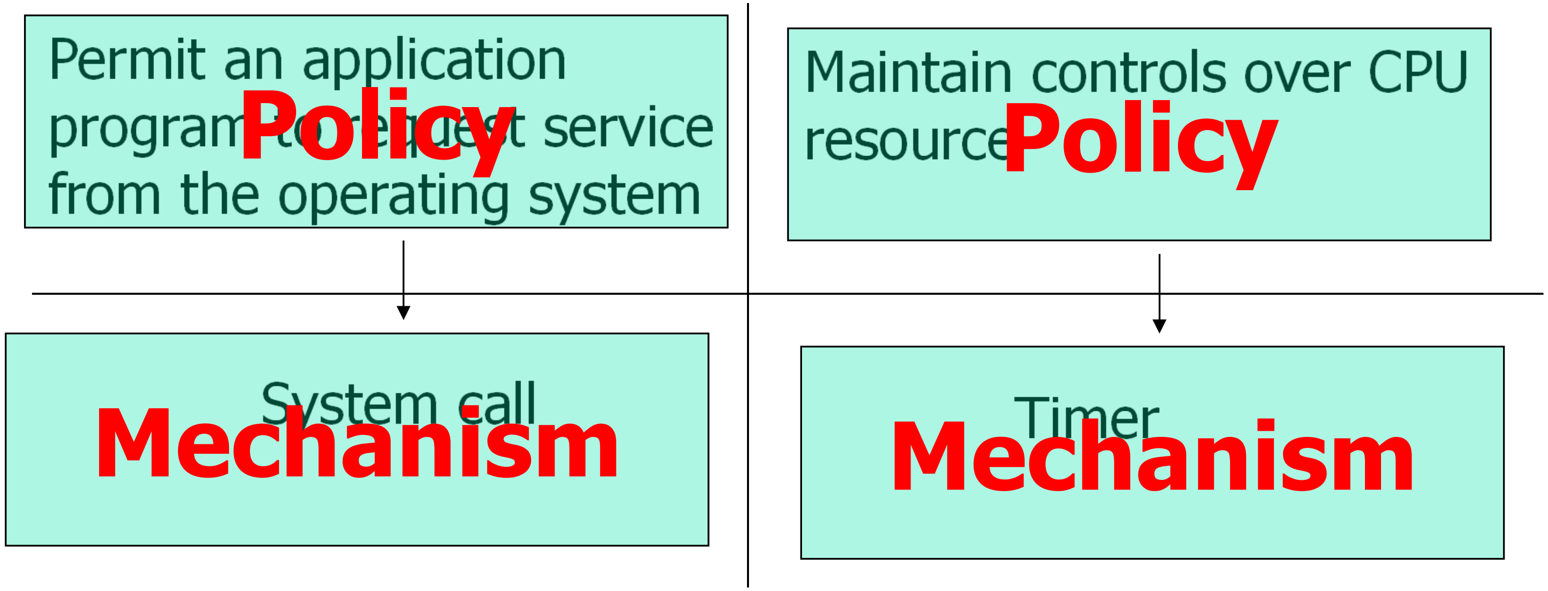
System calls

- Entering a kernel mode
  - 1. Hardware interrupt
  - 2. Trap
    - A mechanism that informs a software event to kernel
      - 1. Exception
        - Divide by zero
        - Illegal machine code
        - Illegal memory access
      - 2. System call

- System call
  - Mechanism used by an application program to request service from the operating system
  - **Typically the library as an intermediary**
    - Through API (Application Program Interface)



- Important design principle
  - **Policy:** What will be done?
  - **Mechanism:** How to do it?



Permit an application program to request service from the operating system

Policy

Maintain controls over CPU resource

Policy

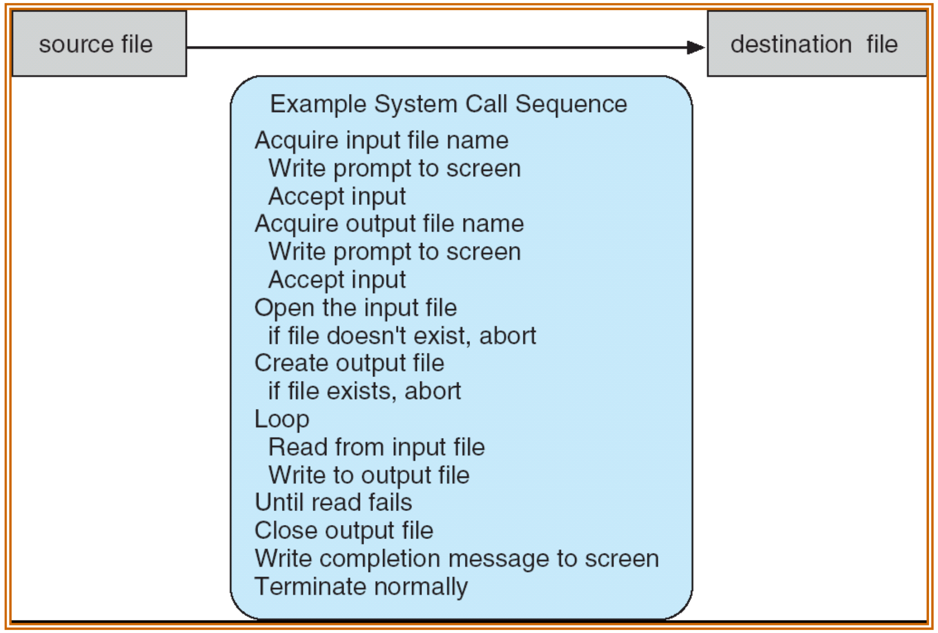
System call

Mechanism

Timer

Mechanism

- Example of System Calls
  - System call sequences to copy the contents of one file to another file

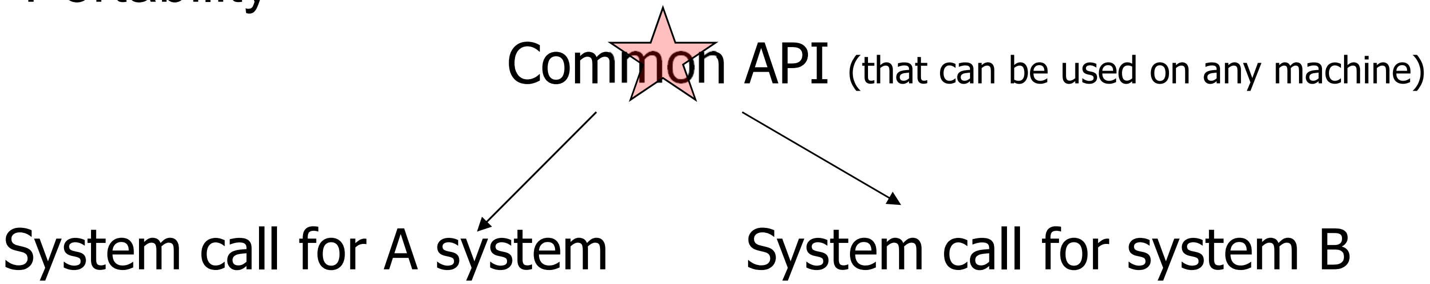


```
Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

Too many system calls !

- API
  - Win32 API
  - POSIX (Portable Operating System Interface)
  - Examples
    - Win32 API: CreateProcess()
      - System call: NTCreateProcess()
    - POSIX API: malloc()
      - System call: sbrk()

- Why use API instead of system call directly
  - Portability



Common API (that can be used on any machine)

System call for A system

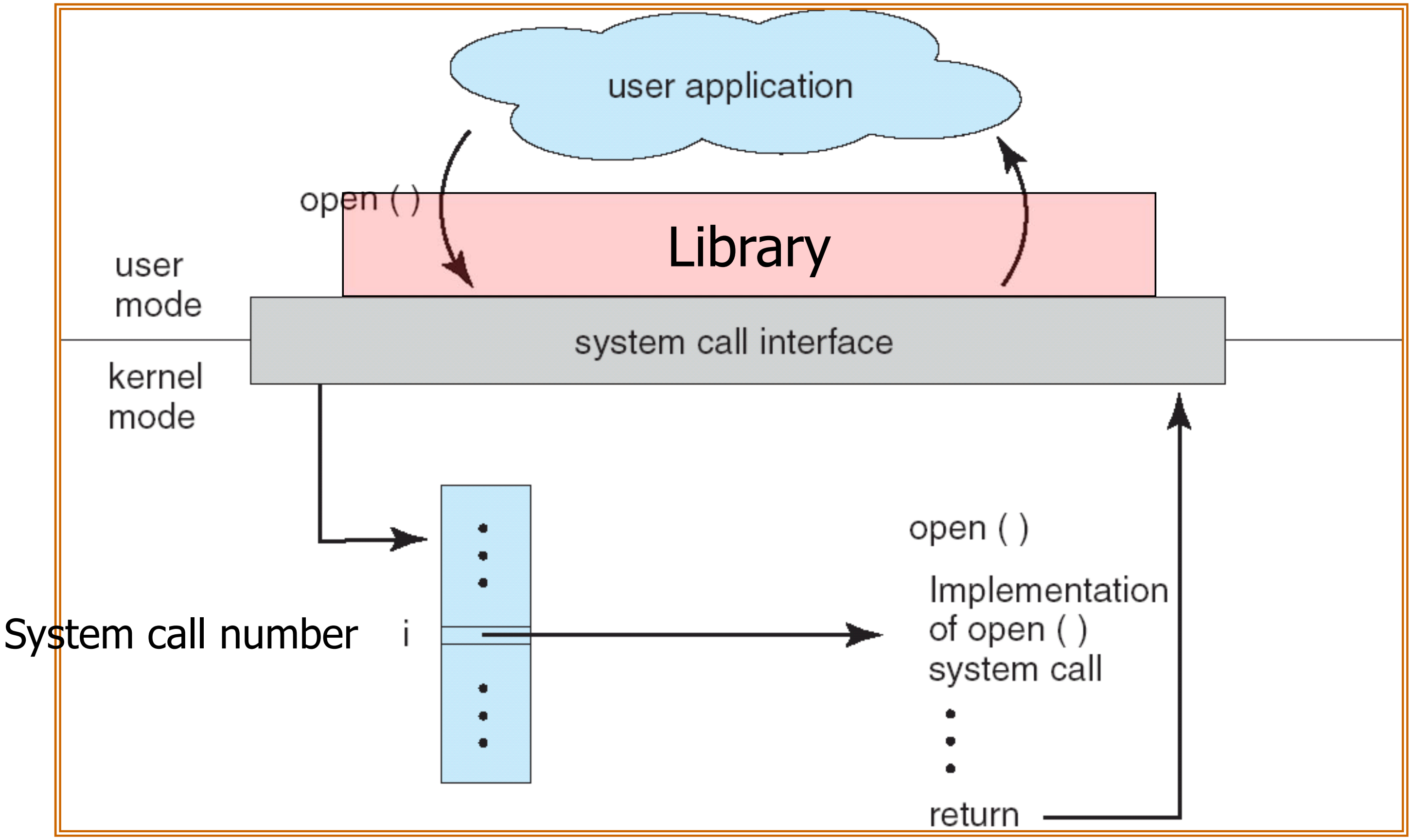
System call for system B

  - Ease of programming
    - An application programmer does not need to know the details of system calls

- System call interface
  - Serves as links to system calls
    - 1. A number is associated with each system call
    - 2. A table indexed according to these numbers
    - 3. Invokes intended system call in OS kernel and returns status of the system call and any return values

sys_call table
1. sys_exit()
2. sys_fork()
…….

System call interface



user application

open ( )

Library

user mode

system call interface

kernel mode

System call number

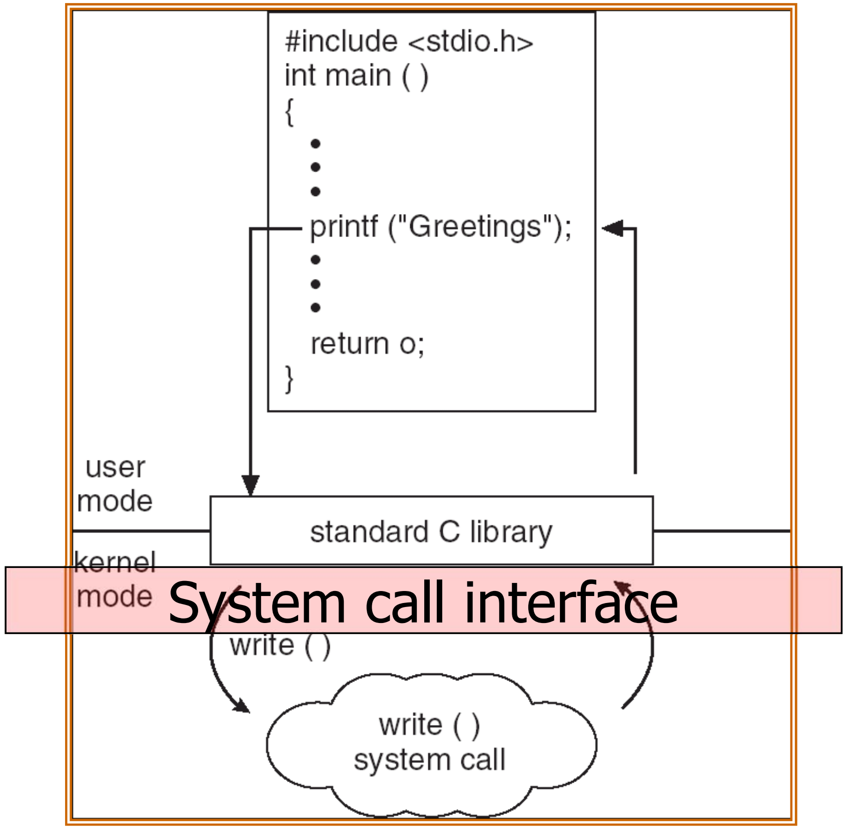
i

open ( )

Implementation of open ( ) system call

return

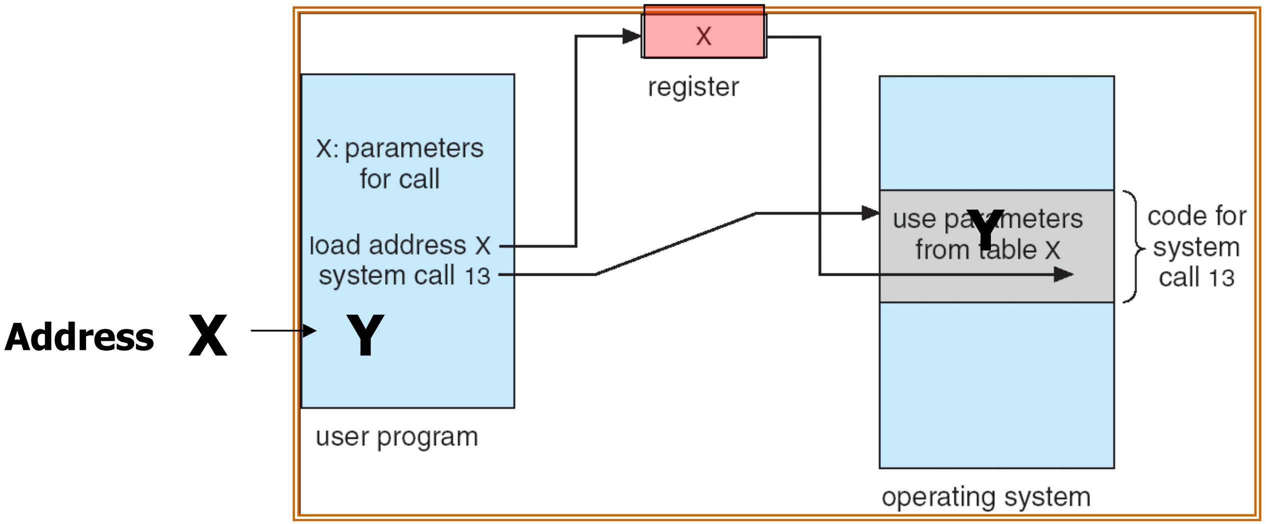
- Another example



- Passing parameters in system calls
  - Simplest: pass the parameters in *registers*
    - In some cases, there may be more parameters than registers
  - Parameters stored in a *block,* or table, in memory, and address of block passed as a parameter in a register

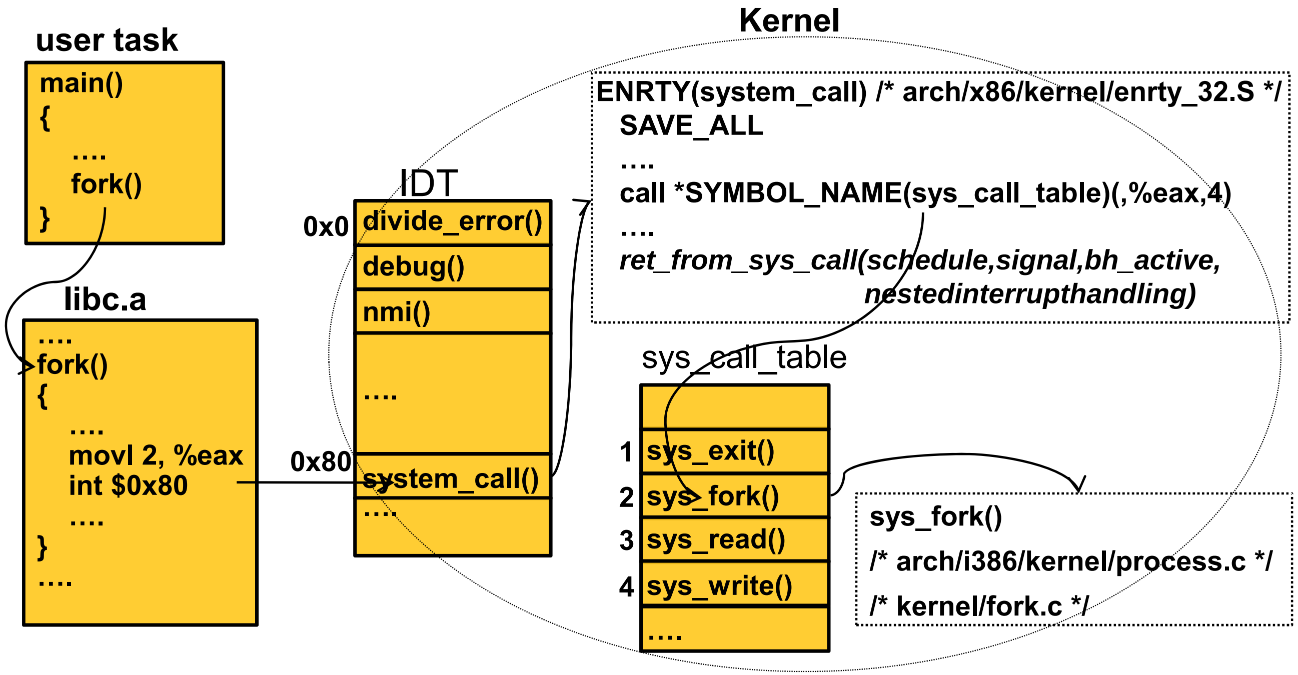
The second method;

Address: X, Arguments: Y



This method is used in Linux or Solaris

<Linux system call example>



user task

```
main()
{
    ....
    fork()
}
```

libc.a

```
....
fork()
{
    ....
    movl 2, %eax
    int $0x80
    ....
}
....
```

IDT

0x0	divide_error()
	debug()
	nmi()
	....
0x80	system_call()
	....

Kernel

```
ENRTY(system_call) /* arch/x86/kernel/enrty_32.S */
  SAVE_ALL
  ....
  call *SYMBOL_NAME(sys_call_table)(,%eax,4)
  ....
  ret_from_sys_call(schedule,signal,bh_active,
                     nestedinterrupthandling)
```

sys_call_table

1	sys_exit()
2	sys_fork()
3	sys_read()
4	sys_write()
	....

```
sys_fork()
/* arch/i386/kernel/process.c */
/* kernel/fork.c */
```