Amirkabir University of Technology
(Tehran Polytechnic)

Operating Systems

Multiprogramming, Dual-mode and System Calls

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Fall 2021

Part1

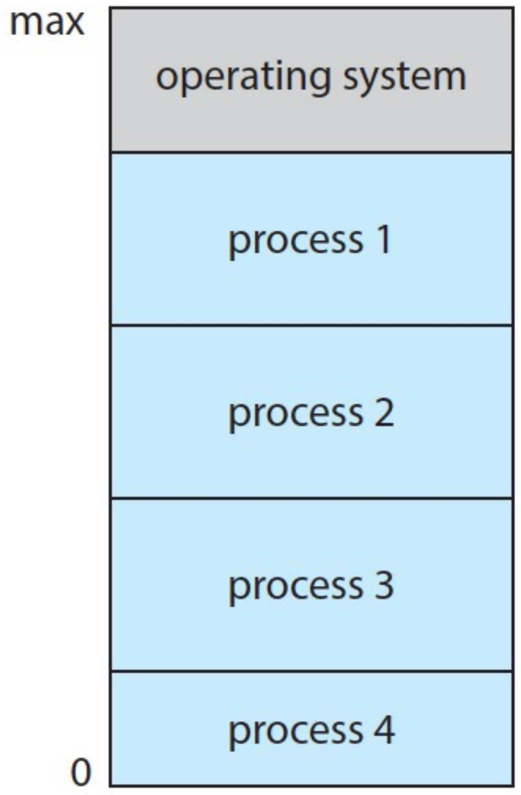
MULTIPROGRAMMING AND DUAL-MODE

Multiprogramming (Batch System) (cont.)

- Multiprogramming organizes multiple jobs (code and data) -->
  - CPU always has one to execute.
- A subset of total jobs

  in system is kept in memory.
- One job selected and run via job scheduling.
- When job has to wait (I/O for example),

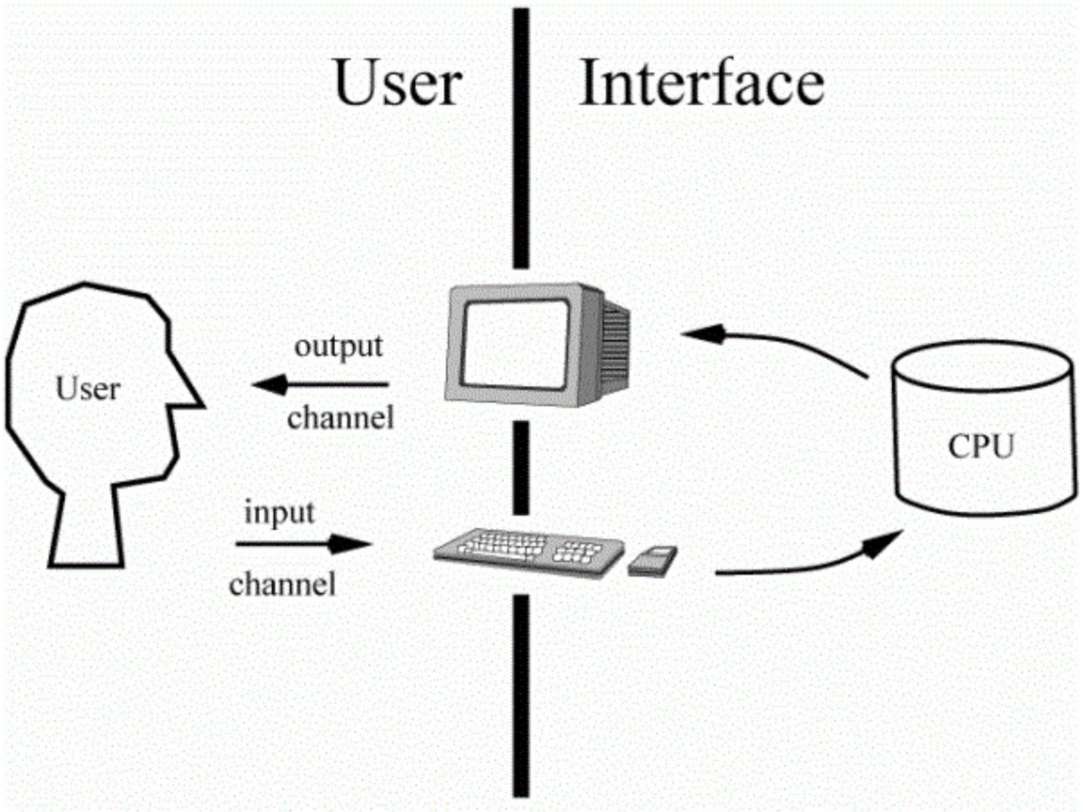
  OS switches to another job.



Memory layout for a multiprogramming system

Multiprogramming (Batch System)

- Single user/program cannot always keep CPU and I/O devices busy.



Multiprogramming (Batch System) (cont.)

- Single user/program cannot always keep CPU and I/O devices busy.
- Examples

Program	CPU-intensive	Memory-intensive	I/O-intensive
Random Number Generator	?	?	?
Microsoft word	?	?	?
QuickTime Player (a long 4K video)	?	?	?

Multitasking (Timesharing)

- A logical extension of Batch systems
- The CPU ***switches jobs so frequently*** that users can interact with each job while it is running, creating **interactive** computing.
  - Response time should be < 1 second.
  - Each user has at least one program executing in memory ⇨ process.
  - If several jobs ready to run at the same time ⇨ CPU scheduling.
  - If processes don’t fit in memory, swapping moves them in&out to run.
  - Virtual memory allows execution of processes not completely in memory.

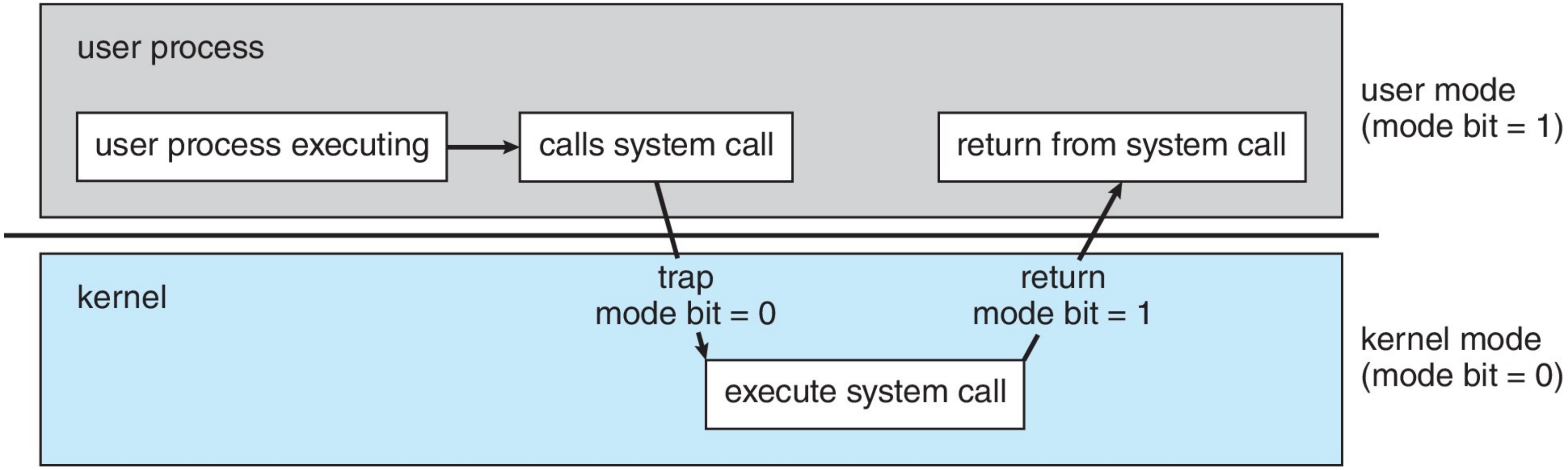
https://www.geeksforgeeks.org/difference-between-job-task-and-process/

Dual-mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components.
  - **User mode** and **kernel mode**
- **Mode bit** provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code.
  - When a user is running ⇨ mode bit is “user”.
  - When kernel code is executing ⇨ mode bit is “kernel”.

Dual-mode Operation (Cont.)

- How do we guarantee that user does not explicitly set the mode bit to “kernel”?
  - System call changes mode to kernel, return from call resets it to user.



Types of Instructions

- Instructions are divided into two categories:
  - The ***non-privileged instruction*** instruction is an instruction that ***any application or user can execute***.
  - The ***privileged instruction*** is an instruction that ***can only be executed in kernel mode***.
- Instructions are divided in this manner because privileged instructions ***could harm the kernel***.

http://web.cs.ucla.edu/classes/winter13/cs111/scribe/4a/

Examples of instructions

Instruction	Type
Reading the status of Processor	?
Set the Timer	?
Sending the final printout of Printer	?
Remove a process from the memory	?

Examples of non-privileged instructions

- Reading the status of Processor
- Reading the System Time
- Sending the final printout of Printer

https://www.geeksforgeeks.org/privileged-and-non-privileged-instructions-in-operating-system/

Examples of privileged instructions

- I/O instructions and halt instructions
- Turn off all Interrupts
- Set the timer
- Context switching
- Clear the memory or remove a process from the memory
- Modify entries in the device-status table

https://www.geeksforgeeks.org/privileged-and-non-privileged-instructions-in-operating-system/

Privileged instructions

If an attempt is made to execute a privileged instruction in user mode

↓

The hardware ***does not execute the instruction*** but rather treats it as ***illegal*** and ***traps*** it to the ***operating system.***

Questions?

Part2

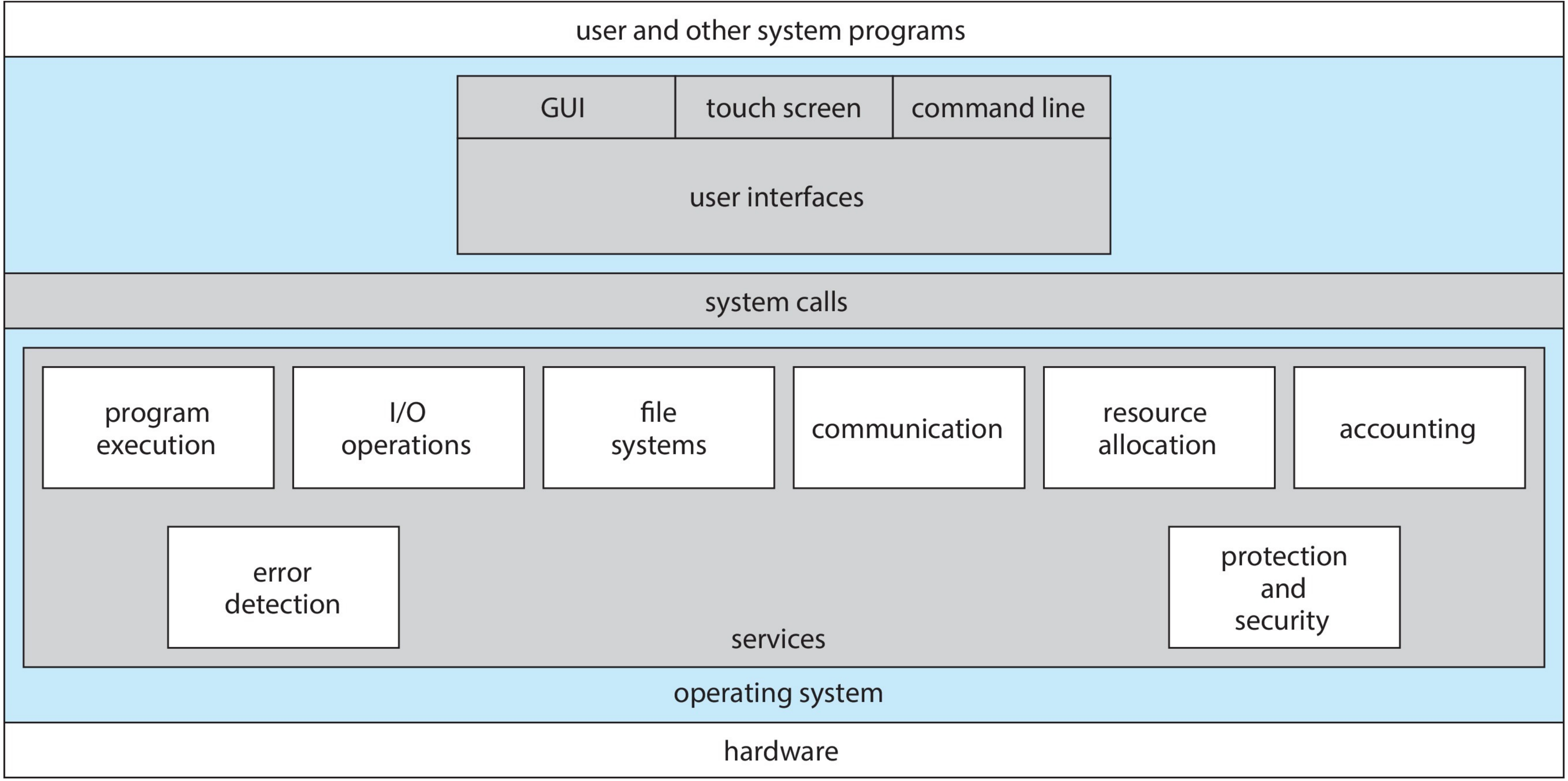
SYSTEM CALLS

System Calls

* راه استفاده از امکانات سیستم عامل

- Programming interface to the services provided by the OS.

* یک سری API یا توابع از پیش تعریف شده برای استفاده از قسمت های مختلف کرنل

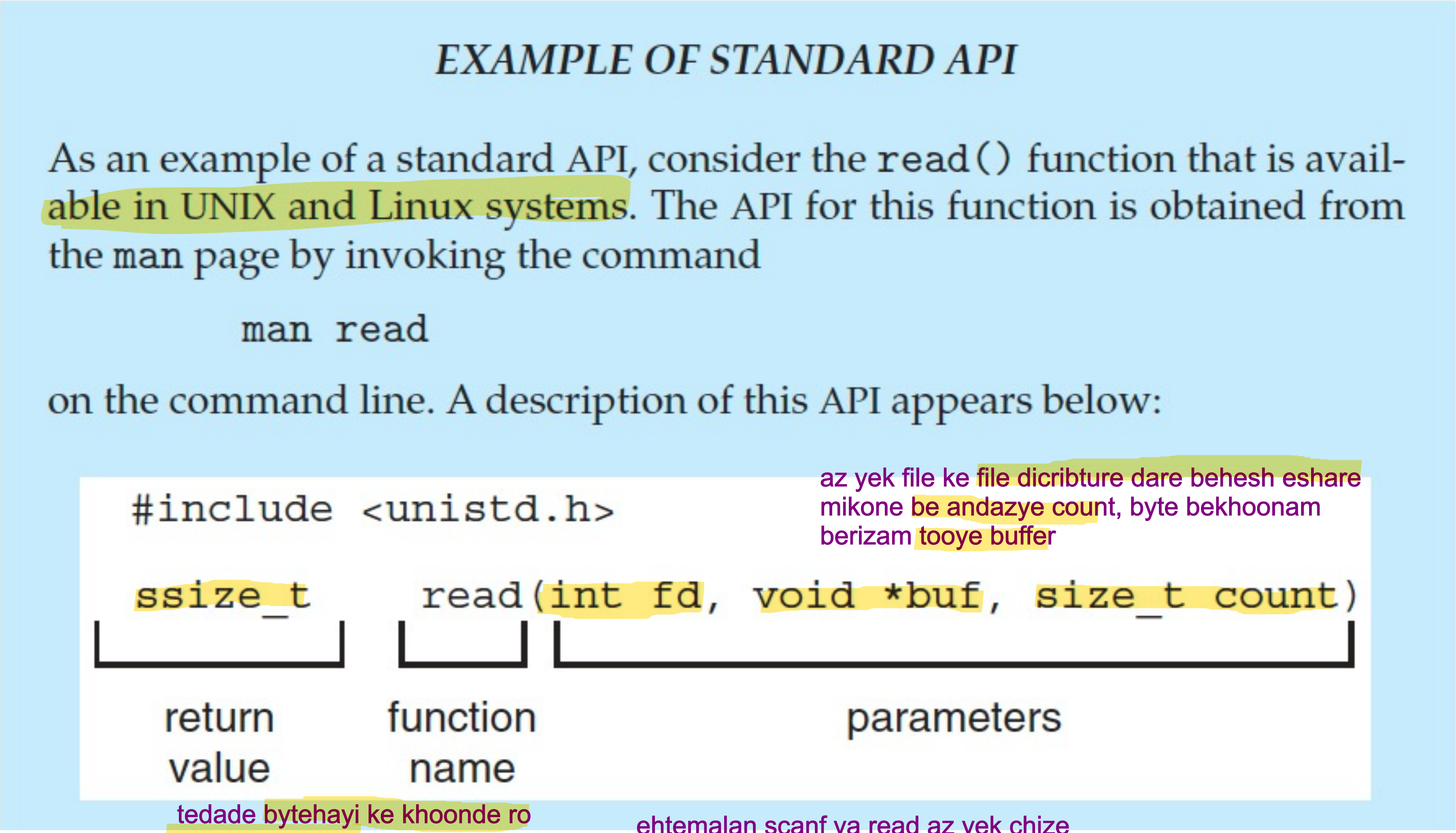


System Calls (cont.)

- Typically written in a high-level language (C or C++ or Assembly).
- Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use.
- Three most common APIs are:
  - Win32 API for Windows (Win API)
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux (***unistd.h***), and Mac OS X)
  - Java API for the Java virtual machine (JVM).

Note that the system-call names used throughout this text are generic.

Example of Standard API



EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t    read(int fd, void *buf, size_t count)
```

return value

function name

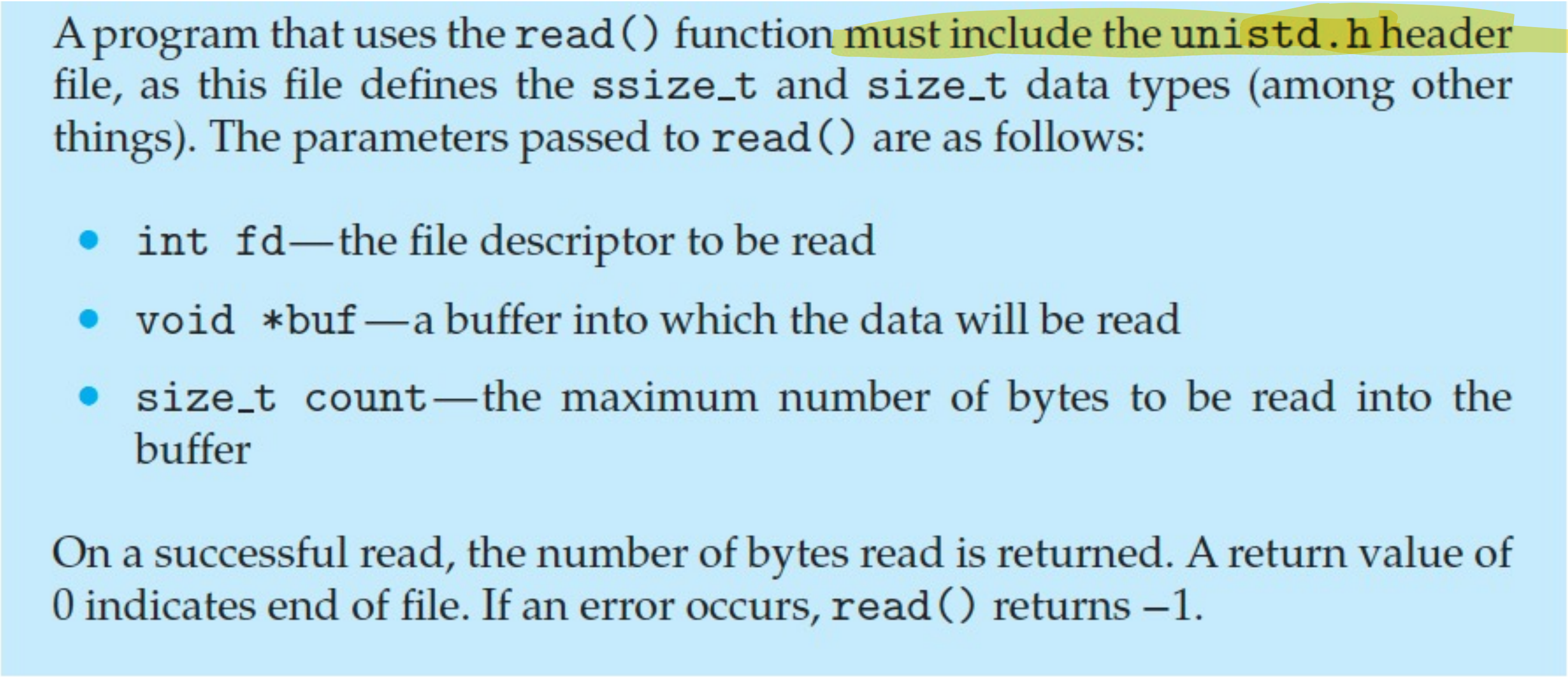
parameters

az yek file ke file dicribture dare behesh eshare mikone be andazye count, byte bekhoonam berizam tooye buffer

tedade bytehayi ke khoonde ro bar migardoone va agar moshkeli pish biyad -1 bar migardoone

ehtemalan scanf va read az yek chize yeksan estefade mikonan

Example of Standard API (Cont.)



A program that uses the read() function must include the unistd.h header file, as this file defines the ssize_t and size_t data types (among other things). The parameters passed to read() are as follows:

- int fd—the file descriptor to be read
- void *buf—a buffer into which the data will be read
- size_t count—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, read() returns −1.

barnamehayi ke masalan ba c neveshte mishavand OS Spesific hastan. yani barnameyi ke dakhele LINUX neveshte shode ro nemitoonim dakhele windows compile o run konim. ya barax masalan age ye file .exe ro bebarim dakhele LINUX va run begirim error mide va mige aslan .exe chiye :|

Screenshot 32 bekhoon

System Call Implementation

dar linux bishtar az 256ta system call nadarim

- Typically, a number is associated with each system call
  - System-call interface maintains a table indexed according to these numbers.

فراخوانی کردن

مورد نظر

- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values

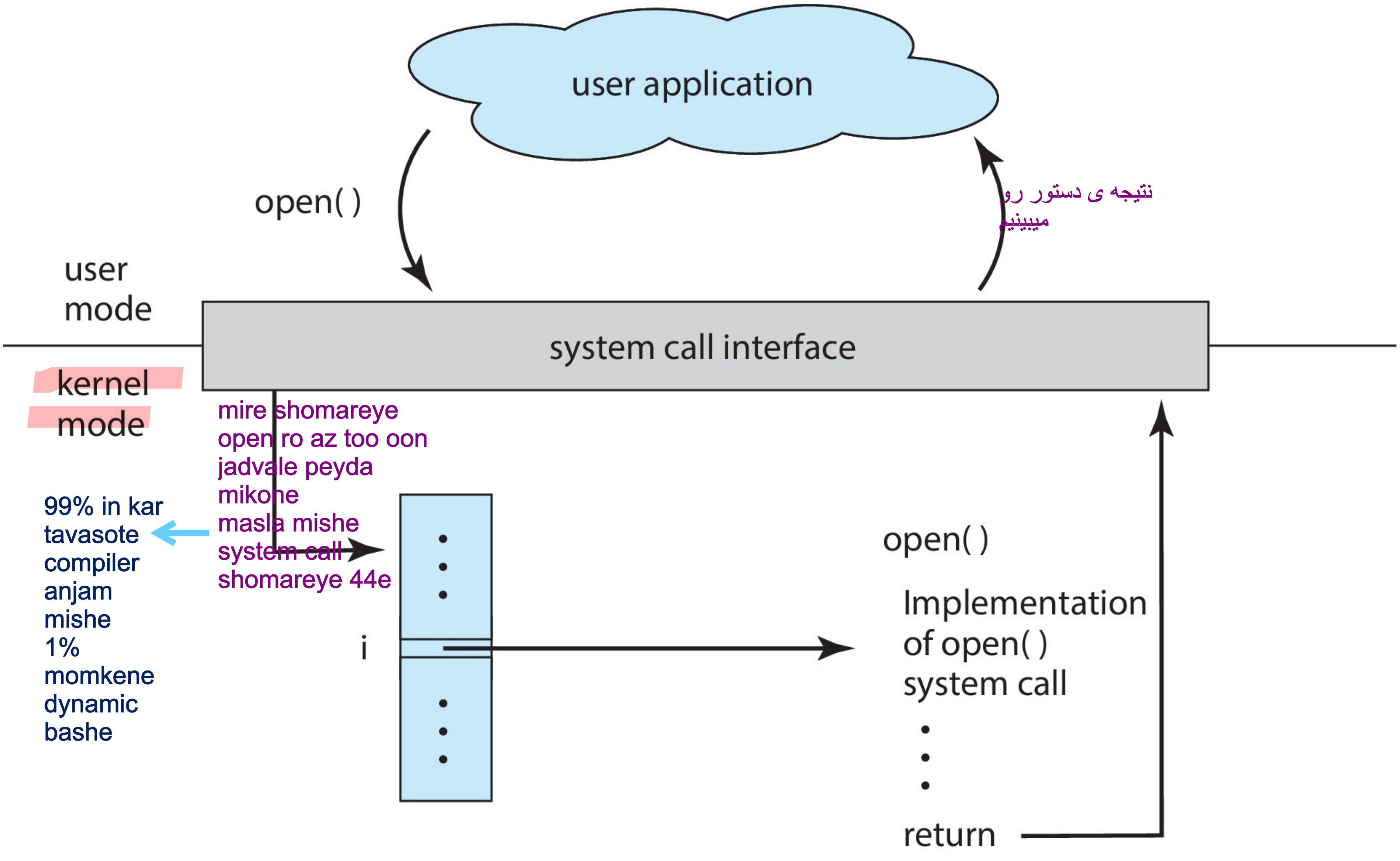
How do we use system calls?

- The caller need know nothing about how the system call is implemented

niyazi nist chizi az dakhelesh bedoonim. hamchenin masalan age ye code darim ke toosh az ye API mese read estefade kardim, piyadesazi zirinesh momkene avaz beshe vali ma niazi nist taghiri too codemoon bedim. kheyli kam pish miyad name yek API avaz beshe

  - Just needs to obey API and understand what OS will do as a result call.
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler).

API – System Call – OS Relationship



نتیجه ی دستور رو
میبینیم

mire shomareye open ro az too oon jadvale peyda mikone masla mishe system call shomareye 44e

99% in kar tavasote compiler anjam mishe 1% momkene dynamic bashe

System calls in assembly programs (demo)

صرفا برای یادگیری

- Put the system call number in the EAX register.
- Store the arguments to the system call in the registers EBX, ECX,...
- Call the relevant interrupt (80h).

int 80h → in hamoon dastooras ke mode bit ro az 1 be 0 taghir mide

system call yani interrupt narmafzari shomareye 80

yek interrupt narmafzari ya trap

- The result is usually returned in the EAX register.

Let's see it in practice ☺

Screenshot 31 ro bekhoon

https://www.tutorialspoint.com/assembly_programming/assembly_system_calls.htm

System Call Parameter Passing

- **Parameter Passing**
  - Register
  - Register pointer to mem. Block
  - Stack (Push, Pop)
- Often, more information is required than simply identity of desired system call.
- Exact type and amount of information vary according to OS and call.

System Call Parameter Passing--Methods

- **Simplest**: pass the parameters in registers.
  - In some cases, may be more parameters than registers. مشکل رجیستر ها

روی تعداد پارامتر ها و روی طول پارامتر ها محدودیت داریم

!مهم! تستی

- **Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register.**
  - This approach taken by Linux and Solaris.

- **Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system.**

jayi az code kernel ke be vaseteye system call seda zade mishe

param 3
param 2
param 1

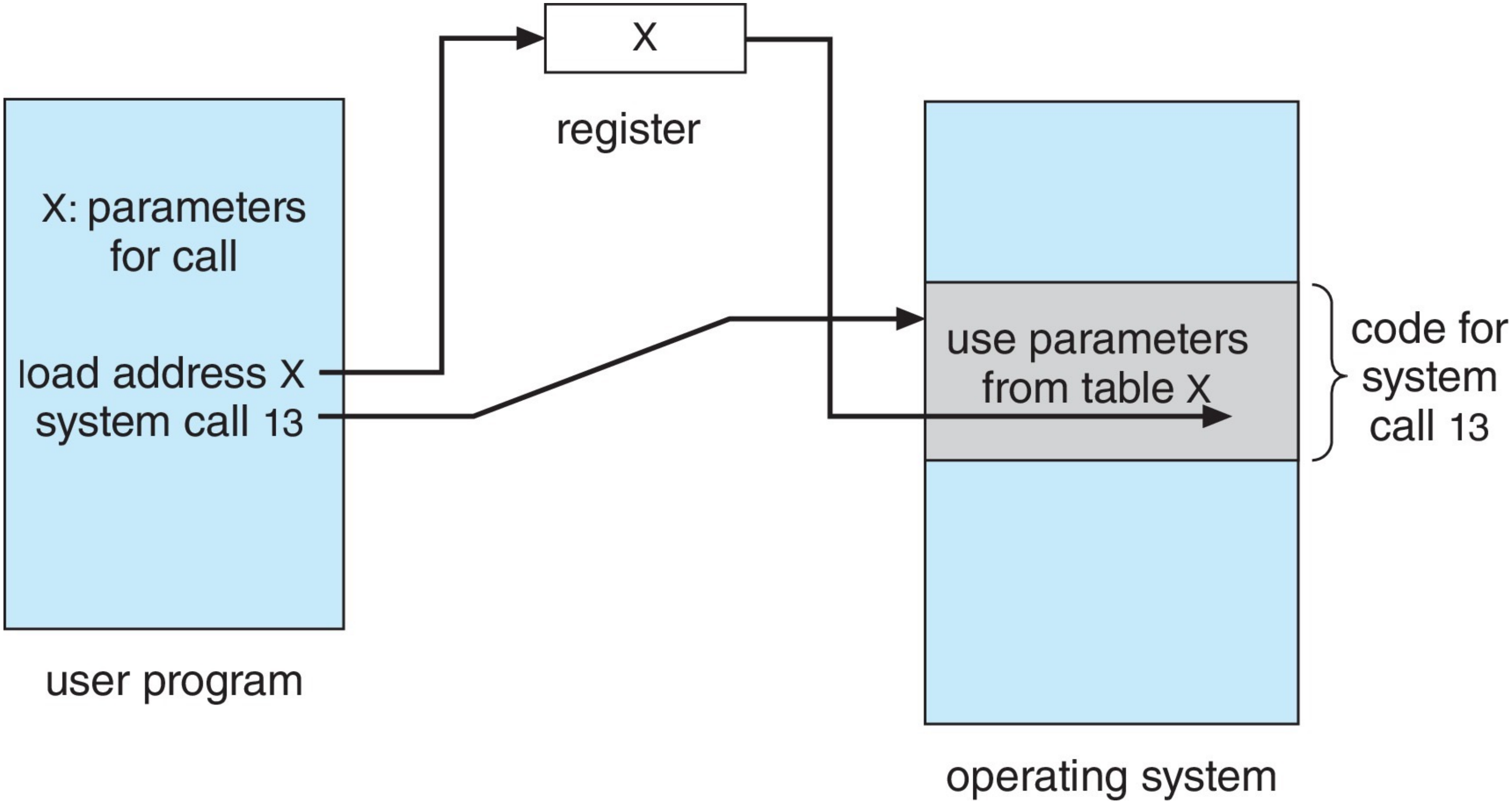
yek stack dakhele ram hast. har thread stack khodesh ro dare.

- Block and stack methods do not limit the number or length of parameters being passed.

inja os midoone age param 1 ro bekhad bayad 2 , 3 ro pop kone vali age ram bashe mostaghim mitoone bere param 1 ro bardare

Parameter Passing via Table

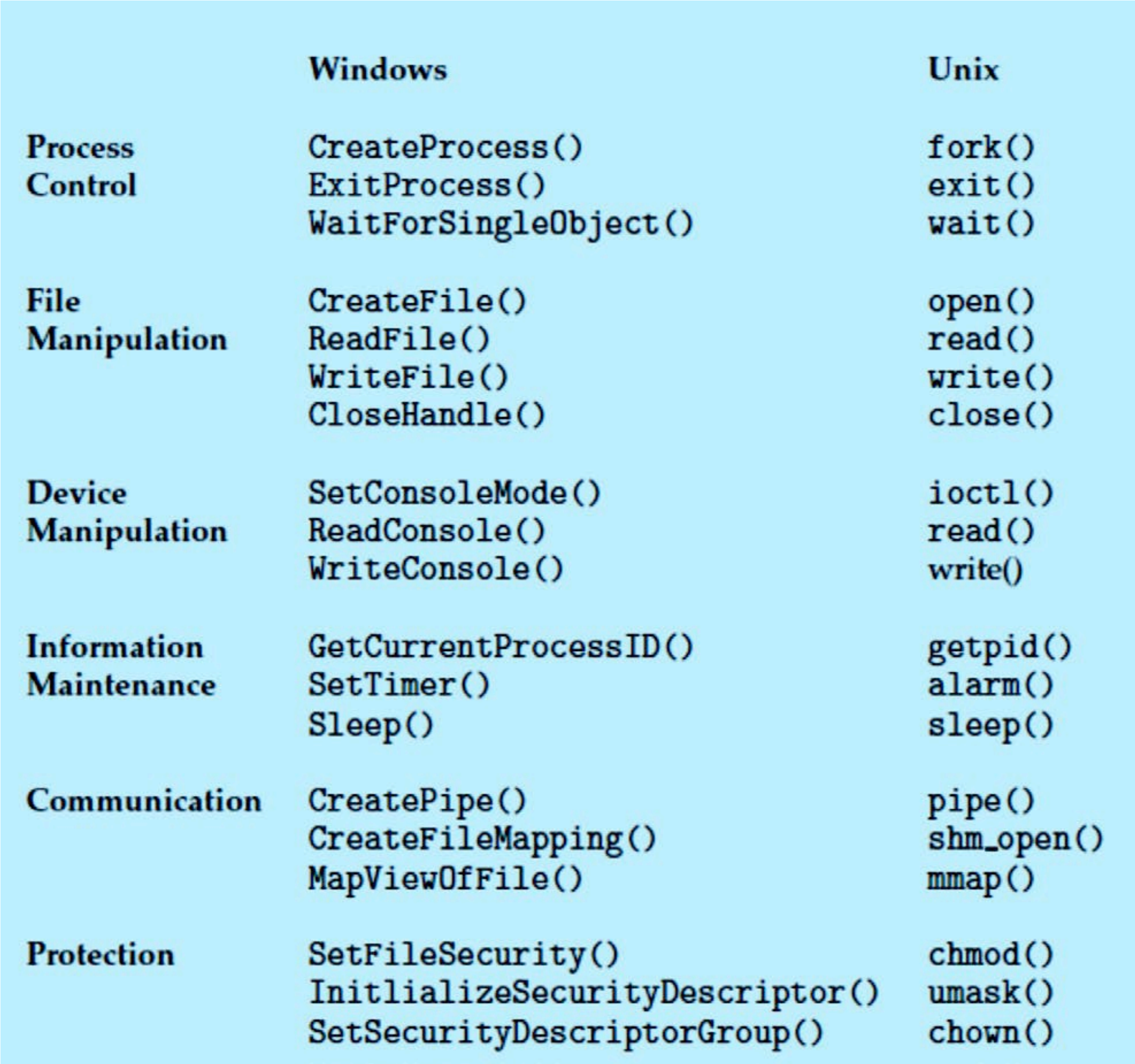
توضیح نداد



Types of System Calls

- **Process control**
  - Create process, terminate process
  - …
- **File management**
  - create file, delete file
  - …
- **Device management**
  - request device, release device
  - …
- **Please study the reference book for more details**

Types of System Calls (Cont.)



	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shm_open()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

Why Applications are Operating System Specific

APPs are OS specific!

- Apps compiled on one system usually not executable on other OSs.

- Each OS provides its own unique system calls
  - Own file formats, etc.

- Apps can be multi-operating system
  - Written in interpreted language like Python, Ruby, and interpreter available on multiple OSs.
  - App written in language that includes a VM containing the running app (like Java). Like java virtual machine

    Virtual Machine
  - Use standard language (like C), compile separately on each operating system to run on each.

Questions?