Announcement

- **This is your score**
  - **Please check it**

	10	10	10	10	20	20	20	100
학번	Attendance	PA0	PA1	PA2	PA3	PA4	PA5	Sum
2753	9.091	10.000	9.039	9.000	18.000	9.667	15.190	79.986
3065	8.182	10.000	3.013	8.000	18.135	15.382		62.713
0717	9.091	10.000	9.031	9.215	19.940	14.743	10.000	82.020
3814	9.091	9.000	8.114	9.056	18.800	16.476	8.000	78.537
2243	8.182	10.000	6.011	9.199	18.189			51.581
4275	9.091	10.000	9.673	10.000	20.000	19.732	17.840	96.336
2019	3.636	10.000	8.000	9.290				30.927
0197	7.273							7.273
4601	9.091	10.000		8.120				27.211
0963	9.091	10.000	9.041	9.457	19.140	19.117	17.491	93.337
0399	5.455	10.000	2.758	8.112				26.324
1956	9.091	10.000	10.000	10.000	20.000	20.000	20.000	99.091
2759	9.091	10.000	9.776	10.000	20.000	19.347	18.322	96.536
2852	9.091	10.000	9.429	9.463	19.723	18.761	17.943	94.409
2341	9.091	10.000	9.000	9.600	18.880	10.000		66.571

Course Summary

Prof. Jin-Soo Kim(jinsookim@skku.edu)
TA – Sanghoon Han(sanghoon.han@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

About POSIX

- **Flavors of Unix**
  - System V (AT&T->USL->Novell->SCO->Caldera->SCO)
  - BSD (UC Berkeley)
  - SunOS, Solaris (Sun)
  - IRIX (SGI), AIX (IBM), HP-UX (HP), Mac OS X (Apple)
  - **Linux**, FreeBSD, NetBSD, and etc..

- **In old days we didn't have a standard**
  - Different API
  - Different behavior of each API
  - No portability

About POSIX (2)

- **POSIX(Portable Operating System Interface)**
  - POSIX is a standard that describes a single interface to a Unix-like operating system.
  - POSIX is not an implementation – it is a description!
  - Most system vendors are now conforming to POSIX standards (specifically IEEE 1003.1) – Even Microsoft provides a set of POSIX utilities with the Windows NT 4.0 Resource Kit.

About POSIX (3)

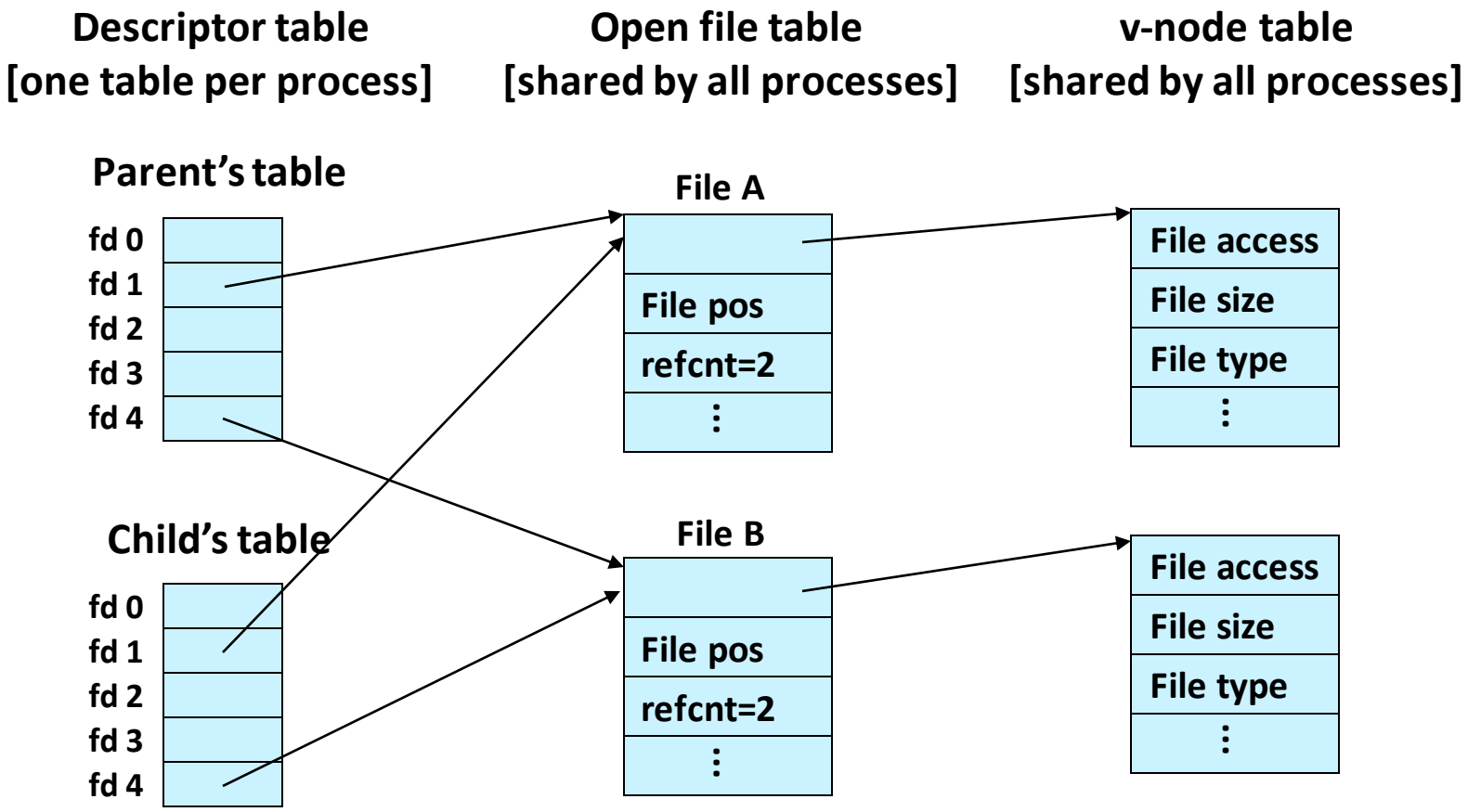
Process Management	fork	Create a new process
	waitpid	Wait for a process to exit
	execve	Load a new binary image
	exit	Terminate execution
	kill	Send a signal
File Management	open	Create a file or open an existing file
	close	Close a file
	read	Read data from a file
	write	Write data to a file
	lseek	Move the file pointer
	stat	Get various file attributes
	chmod	Change the file access permission
File System Management	mkdir	Create a new directory
	rmdir	Remove an empty directory
	link	Make a link to a file
	unlink	Destroy an existing file
	mount	Mount a file system
	umount	Unmount a file system
	chdir	Change the curent working directory
⋮	⋮	⋮

File I/O

- **Everything is a file descriptor**
  - Syscall for File I/O can be used with pipe, socket, …

- **How the Unix kernel represents open files?**
  - Descriptor table
    - 1 table per process
    - Pointer to entry in the "file table"
  - File table
    - Shared by all processes
    - Current file position, mode, reference count, pointer to entry in the "v-node table"
  - v-node table
    - Shared by all processes
    - Information about file itself (size, permission, …)

File I/O (2)

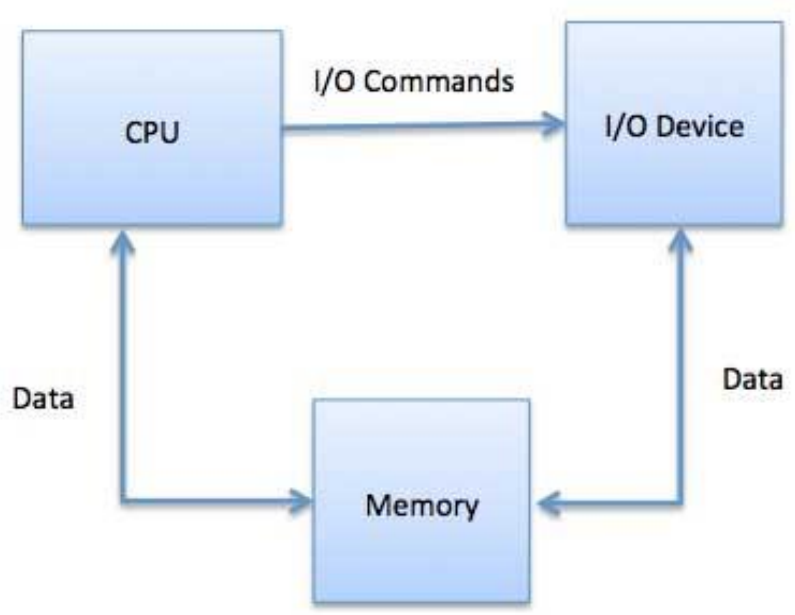

Descriptor table
[one table per process]
Open file table
[shared by all processes]
v-node table
[shared by all processes]
Parent's table
fd 0
fd 1
fd 2
fd 3
fd 4
Child's table
fd 0
fd 1
fd 2
fd 3
fd 4
File A
File pos
refcnt=2
File B
File pos
refcnt=2
File access
File size
File type
File access
File size
File type

File I/O (3)

- **6 System calls**
  - `open()`
  - `close()`
  - `read()`
  - `write()`
  - `lseek()`
  - `stat() / fstat()`

Process

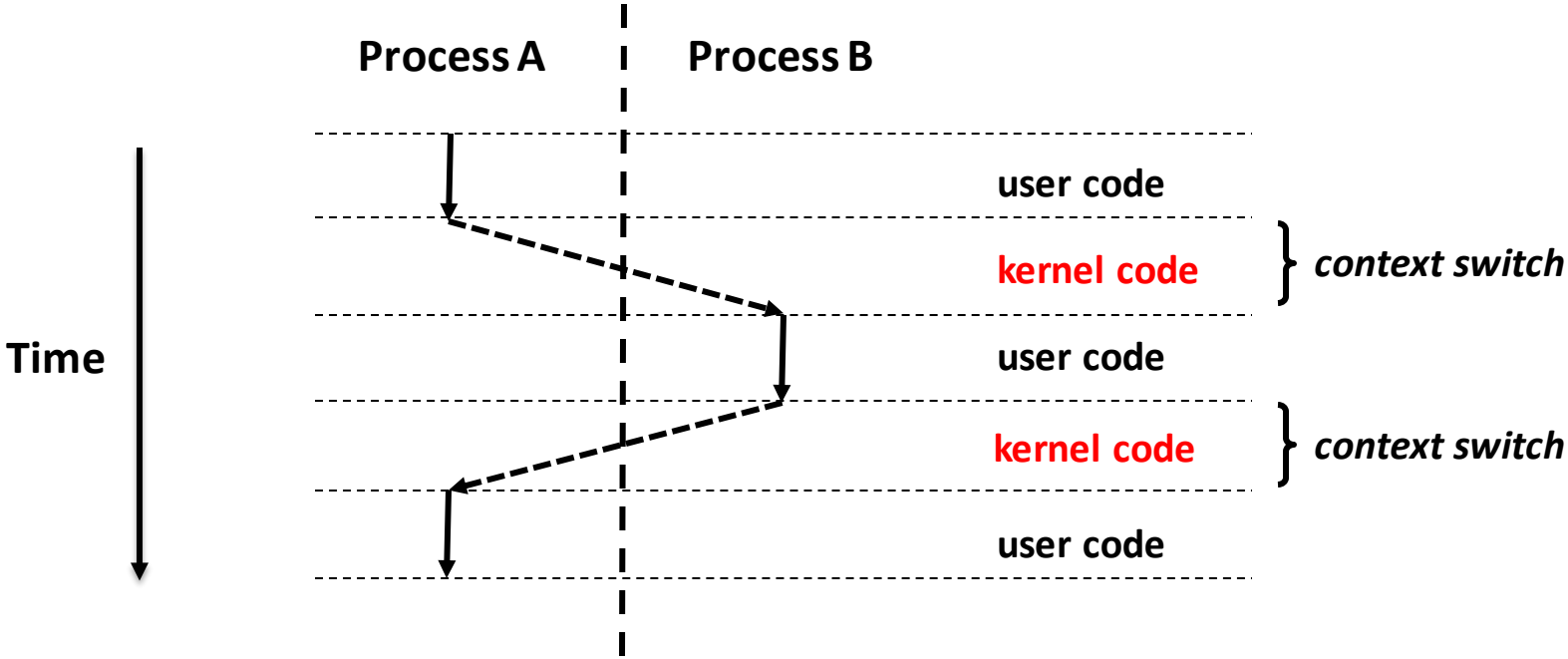
- **Computer is …**



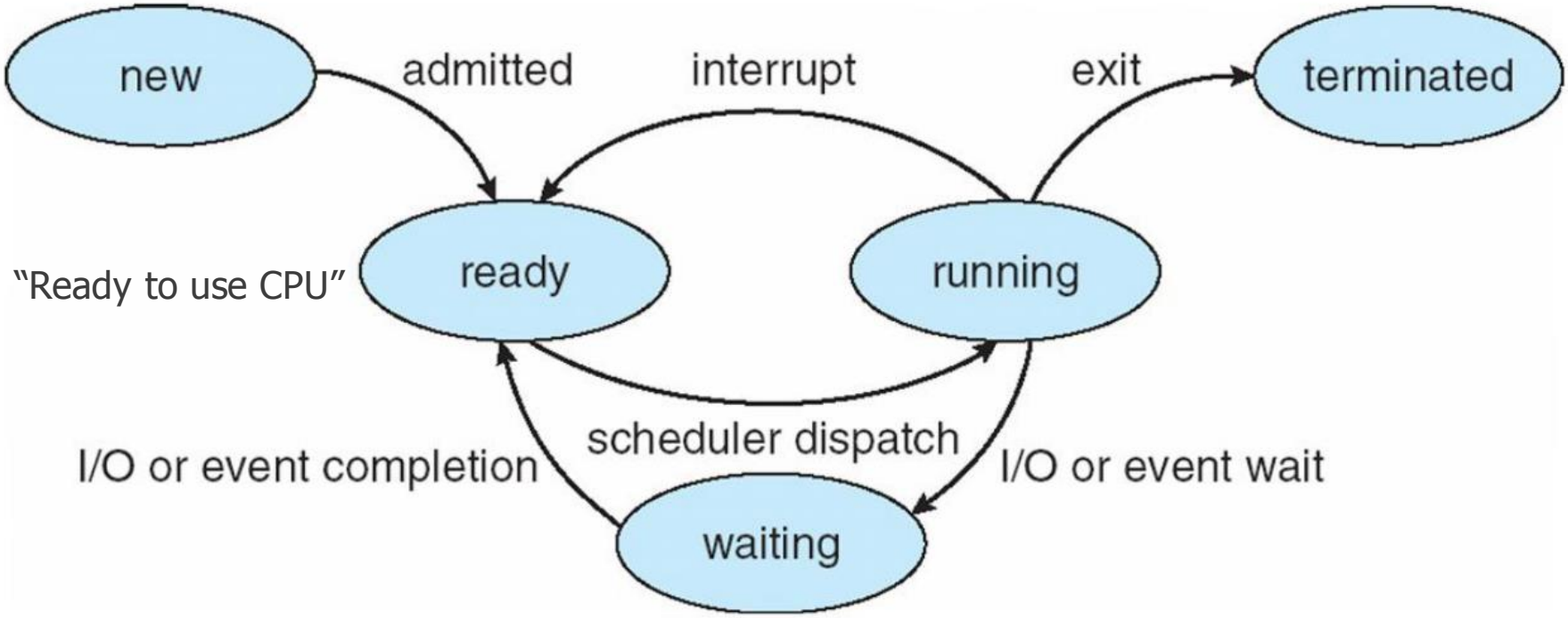
- **How our program works with other programs?**
  - With context switch

Process (2)

- Control flow passes from one process to another via a context switch



Process State Transition



Process (3)

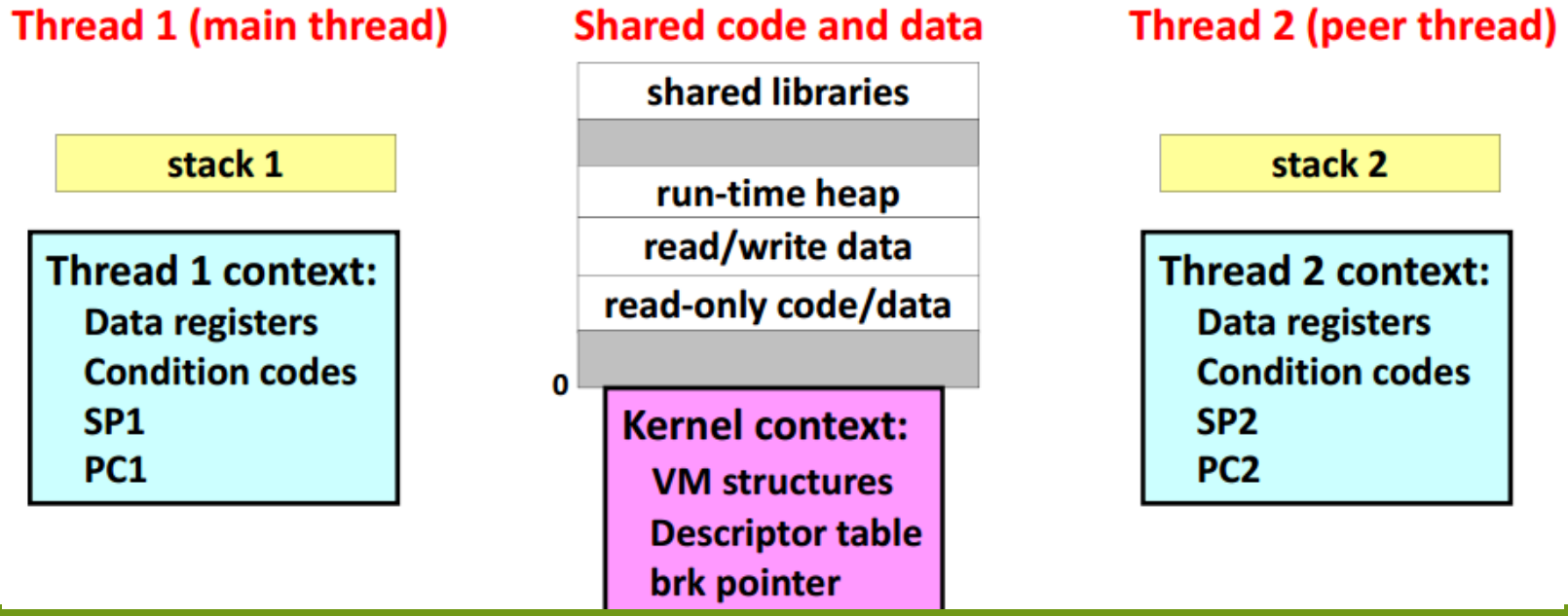
- **Process abstraction**
  - Logical control flow
  - Private address space

- **Process-related system calls**
  - `fork()`
  - `exit()`, `atexit()`
  - `wait()`, `waitpid()`
  - `execl()`, `execle()`, `execv()`, `execve()`, …

Thread

- **Why we need thread?**
  - Context switch is too heavy

- **1 address space, many stacks**



Thread (2)

- **Difference with pthread?**
  - It is fast
  - But it needs mutex, …
- **POSIX Threads Interface**
  - Creating and reaping threads
    - pthread_create()
    - pthread_join()
  - Determining your thread ID
    - pthread_self()
  - Terminating threads
    - pthread_cancel()
    - pthread_exit()

What we learned

Day	Topic	Reading	Assignment
3/7 (T)	Course overview		
3/14 (T)	Introduction to Linux		PA#0
3/21 (T)	File I/O		PA#1
3/28 (T)	Process		
4/4 (T)	Signals		PA#2
4/11 (T)	IPC (Pipes and FIFOs)		
4/18 (T)	Recitation session		
4/25 (T)	Midterm exam week		
5/2 (T)	Sockets		PA#3
5/9 (T)	National holiday		
5/16 (T)	Concurrent programming		
5/23 (T)	Pthreads		PA#4
5/30 (T)	Pthreads (cont'd)		PA#5