### Announcement

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

<table>
<thead>
<tr>
<th>학번</th>
<th>Attendance</th>
<th>PA0</th>
<th>PA1</th>
<th>PA2</th>
<th>PA3</th>
<th>PA4</th>
<th>PA5</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2753</td>
<td>9.091</td>
<td>10.000</td>
<td>9.039</td>
<td>9.000</td>
<td>18.000</td>
<td>9.667</td>
<td>15.190</td>
<td>79.986</td>
</tr>
<tr>
<td>3065</td>
<td>8.182</td>
<td>10.000</td>
<td>3.013</td>
<td>8.000</td>
<td>18.135</td>
<td>15.382</td>
<td></td>
<td>62.713</td>
</tr>
<tr>
<td>3814</td>
<td>9.091</td>
<td>9.000</td>
<td>8.114</td>
<td>9.056</td>
<td>18.800</td>
<td>16.476</td>
<td>8.000</td>
<td>78.537</td>
</tr>
<tr>
<td>2243</td>
<td>8.182</td>
<td>10.000</td>
<td>6.011</td>
<td>9.199</td>
<td>18.189</td>
<td></td>
<td></td>
<td>51.581</td>
</tr>
<tr>
<td>4275</td>
<td>9.091</td>
<td>10.000</td>
<td>9.673</td>
<td>10.000</td>
<td>20.000</td>
<td>19.732</td>
<td>17.840</td>
<td>96.336</td>
</tr>
<tr>
<td>2019</td>
<td>3.636</td>
<td>10.000</td>
<td>8.000</td>
<td>9.290</td>
<td></td>
<td></td>
<td></td>
<td>30.927</td>
</tr>
<tr>
<td>0197</td>
<td>7.273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.273</td>
</tr>
<tr>
<td>4601</td>
<td>9.091</td>
<td>10.000</td>
<td></td>
<td>8.120</td>
<td></td>
<td></td>
<td></td>
<td>27.211</td>
</tr>
<tr>
<td>0399</td>
<td>5.455</td>
<td>10.000</td>
<td>2.758</td>
<td>8.112</td>
<td></td>
<td></td>
<td></td>
<td>26.324</td>
</tr>
<tr>
<td>1956</td>
<td>9.091</td>
<td>10.000</td>
<td>10.000</td>
<td>10.000</td>
<td>20.000</td>
<td>20.000</td>
<td>20.000</td>
<td>99.091</td>
</tr>
<tr>
<td>2759</td>
<td>9.091</td>
<td>10.000</td>
<td>9.776</td>
<td>10.000</td>
<td>20.000</td>
<td>19.347</td>
<td>18.322</td>
<td>96.536</td>
</tr>
<tr>
<td>2341</td>
<td>9.091</td>
<td>10.000</td>
<td>9.000</td>
<td>9.600</td>
<td>18.880</td>
<td>10.000</td>
<td></td>
<td>66.571</td>
</tr>
</tbody>
</table>
Course Summary

Prof. Jin-Soo Kim(jinsookim@skku.edu)
TA – Sanghoon Han(sanghoon.han@cs1.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://cs1.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)

<table>
<thead>
<tr>
<th>Process Management</th>
<th>fork</th>
<th>Create a new process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>waitpid</td>
<td>Wait for a process to exit</td>
</tr>
<tr>
<td></td>
<td>execve</td>
<td>Load a new binary image</td>
</tr>
<tr>
<td></td>
<td>exit</td>
<td>Terminate execution</td>
</tr>
<tr>
<td></td>
<td>kill</td>
<td>Send a signal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Management</th>
<th>open</th>
<th>Create a file or open an existing file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>close</td>
<td>Close a file</td>
</tr>
<tr>
<td></td>
<td>read</td>
<td>Read data from a file</td>
</tr>
<tr>
<td></td>
<td>write</td>
<td>Write data to a file</td>
</tr>
<tr>
<td></td>
<td>lseek</td>
<td>Move the file pointer</td>
</tr>
<tr>
<td></td>
<td>stat</td>
<td>Get various file attributes</td>
</tr>
<tr>
<td></td>
<td>chmod</td>
<td>Change the file access permission</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File System Management</th>
<th>mkdir</th>
<th>Create a new directory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rmdir</td>
<td>Remove an empty directory</td>
</tr>
<tr>
<td></td>
<td>link</td>
<td>Make a link to a file</td>
</tr>
<tr>
<td></td>
<td>unlink</td>
<td>Destroy an existing file</td>
</tr>
<tr>
<td></td>
<td>mount</td>
<td>Mount a file system</td>
</tr>
<tr>
<td></td>
<td>umount</td>
<td>Unmount a file system</td>
</tr>
<tr>
<td></td>
<td>chdir</td>
<td>Change the current working directory</td>
</tr>
</tbody>
</table>
### 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 ...

  ![Diagram of computer processes]

  - How our program works with other programs?
    - With context switch
Control flow passes from one process to another via **a context switch**
Process State Transition

- **new**
- **admitted**
- **interrupt**
- **exit**

- **ready**
  - "Ready to use CPU"
  - I/O or event completion
  - scheduler dispatch
  - I/O or event wait

- **running**

- **waiting**

- **terminated**
Process (3)

- Process abstraction
  - Logical control flow
  - Private address space

- Process-related system calls
  - `fork()`
  - `exit()`, `atexit()`
  - `wait()`, `waitpid()`
  - `execl()`, `execle()`, `execv()`, `execve()`, …
Why we need thread?
- Context switch is too heavy

1 address space, many stacks

Thread 1 (main thread)
- Stack 1
- Thread 1 context:
  - Data registers
  - Condition codes
  - SP1
  - PC1

Shared code and data
- Shared libraries
- Run-time heap
- Read/write data
- Read-only code/data
- Kernel context:
  - VM structures
  - Descriptor table
  - Brk pointer

Thread 2 (peer thread)
- Stack 2
- Thread 2 context:
  - Data registers
  - Condition codes
  - SP2
  - PC2
**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

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