CS 537: Intro to OS

What is an OS?

User

Apps

OS

Hardware

CPU

RAM

Disk

How does an app program run?

hello.c --> gcc --> hello (exec) --> ./hello

Disk

Disk

Memory

hello

code

Data
Memory layout of a C program

- code
- data
- heap
- stack

1. fetch
2. decode
3. execute

CPU

mov 0x1000, %eax

understand what it means.
\*p = atoi (argv[i]);
hello threads - can access the same memory location.

1. load
2. mov
3. store

P1

P2

add
What is a process?

Running program.

Processes

CPU

Memory

- code
- data
- heap
- stack

Disk

Loading

Machine State

Memory (Address Space)

Registers

- PC
- SP
- BP

Files
Process States

Running

Ready

blocked

I/O

I/O complete

P1

0

Run

1

Run

I/O Complete

2

I/O: Blocked

3

...

P2

Run

Ready

Run

Ready

Run

...
PCB - Process Control Block

```c
struct proc {
  context
  page tables
  files
};
```

```
PCB A
PCB B
```

```
XV6
```

```
    0
    A B ...
```
Memory layout of a process.

Kernel Stack

User stack.

OS Kernel

User part of addresses & stack.

stack

heap

data

code

stack

heap

data

code

Kernel Stack

User stack.
Limited Direct Execution.

Problem #1: Restricted Ops

Soln #1: H/W - Processor Modes.

CPU

User

Kernel

System Calls

App

write( ...)  

libc

%eax 4

int 0x80

TRAP \rightarrow Trap handler.

\leftarrow Return From Trap.
Example: Linux "write" system call

C code:

```c
...  
printf("Hello world\n");  
...  
libc:

%eax = sys_write;
int 0x80
```

```c
system_call()
  sc = sys_call_table[%eax]
}
```

```c
sys_write(...){
  //handle write
}
```
Booting

ON \rightarrow BIOS \rightarrow \text{Run boot loader} \rightarrow \text{starts Xv6 kernel.}

[prepare h/w] + start os

Files: bootasm.s, bootmain.c, & entry.s
XV6

Kernel starts...

first user process

init

fork

sh

pid = 2

fork

hello

pid = 3

getpid() → 3

getppid() → 2
Binary File

<table>
<thead>
<tr>
<th>3</th>
<th>0</th>
<th>3</th>
<th>d1</th>
</tr>
</thead>
<tbody>
<tr>
<td>d2</td>
<td>d3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>d4</td>
<td>d5</td>
<td>d6</td>
<td>d7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>d8</td>
<td>d9</td>
</tr>
</tbody>
</table>

64 bytes.

<table>
<thead>
<tr>
<th>index</th>
<th>data_nums</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
</tr>
<tr>
<td>d2</td>
</tr>
<tr>
<td>d3</td>
</tr>
<tr>
<td>d4</td>
</tr>
<tr>
<td>d5</td>
</tr>
<tr>
<td>d6</td>
</tr>
<tr>
<td>d7</td>
</tr>
<tr>
<td>d8</td>
</tr>
<tr>
<td>d9</td>
</tr>
</tbody>
</table>