CMPSCI 230
Computer Systems Principles

Processes
Objectives

• To understand what a process is
• To learn the basics of exceptional control flow
• To learn how to create child processes
How to run programs?

How does the operating system run programs?
Program Abstraction

How does the operating system run programs?

Need abstraction of executing program
Process Abstraction

How does the operating system run programs?

Need abstraction of executing program

\textbf{process} = \textit{memory state} + \textit{machine state}
Multi-processing

The OS must handle multiple processes in execution at any given time...

How do we do this?

Exceptional control flow
Exceptions + Processes

What is a process?

An independent *logical control flow* that provides the illusion that our program has exclusive use of the processor

A private address space that provides the illusion that our program has exclusive use of the memory system
How do we control processes?
How do we control processes?

Need a way to identify processes
Obtaining Process IDs

00_getpid.c
How do we control processes?

Need a way to create and destroy processes
Create & Terminate Processes

A process is in one of three states

Running  Stopped  Terminated
Create & Terminate Processes

A process is in one of three states

- **Running**: In the running state it is either executing on the CPU or is waiting to be executed and eventually scheduled by the kernel.
Create & Terminate Processes

A process is in one of three states

Running  Stopped  Terminated

In the stopped state the execution of the process is suspended and will not be scheduled. A process stops if it receives a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal (more on signals later)
Create & Terminate Processes

A process is in one of three states

Running  Stopped  Terminated

In the terminated state the process is stopped permanently. A process becomes terminated for one of three reasons: (1) receiving a signal to terminate, (2) returning from main, or (3) calling the `exit` function.
Create & Terminate Processes

A process is in one of three states:

- **Running**
- **Stopped**
- **Terminated**

In the terminated state the process is stopped permanently. A process becomes terminated for one of three reasons: (1) receiving a signal to terminate, (2) returning from main, or (3) calling the `exit` function.

01_exit.c  02_exit.c
Create & Terminate Processes

A process is in one of three states

- Running
- Stopped
- Terminated

So, how do we create a running process?
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```

fork() creates a new process. It is almost identical to the parent – but, has a different PID. It gets a copy of the parent’s virtual address space, both heap and stack.
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```

fork() returns **different results** to the parent and child:

- **Parent** gets the pid of the new child
- **Child** gets 0 (so it can easily know it’s the child) - if it needs its own pid it can simply call getpid to obtain it
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```
Create & Terminate Processes

We use the fork() function to create a new process.

```c
pid_t pid = fork();
if (pid == 0) {
    printf("hello from child\n");
} else {
    printf("hello from parent\n");
}
```

```c
03_fork.c
```
Create & Terminate Processes

What does this print out?

```c
void main()
{
    printf("L0\n");
    fork();
    printf("L1\n");
    fork();
    printf("Bye\n");
}
```
Create & Terminate Processes

What does this print out?

```c
void main()
{
    printf("L0\n");
    fork();
    printf("L1\n");
    fork();
    printf("L2\n");
    fork();
    printf("Bye\n");
}
```

Does it always print in order?
Create & Terminate Processes

What does this print out?

```c
void main()
{
    printf("L0\n");
    if (fork() != 0) {
        printf("L1\n");
        if (fork() != 0) {
            printf("L2\n");
        fork();
        }
    }
    printf("Bye\n");
}
```

iClicker question: How many lines of output will it produce?

A) 4
B) 5
C) 6
D) 7
E) 8

05_fork.c
Create & Terminate Processes

What does this print out?

```c
void main()
{
    printf("L0\n");
    if (fork() != 0) {
        printf("L1\n");
        if (fork() != 0) {
            printf("L2\n");
            fork();
        }
    }
    printf("Bye\n");
}
```

05_fork.c