Processes

CS 241

Sept. 23, 2013

What is a process?

- A **process** is an instance of a running program.
  - A process is not the binary, it is an instance of the binary running!

- A **process** provides two key abstractions:
  - Private virtual address space
    - Each process has its own virtual address space (0x0 ➔ 0xff…ff)… *all the memory stuff we just covered!*
  - Logical control flow
    - Each program seems to have continued, exclusive use of the CPU(s)

In the beginning…

- When your computer starts:
  1. The CPU runs the system's firmware:
     - Old standard: BIOS
     - Newer standard: EFI / UEFI
  2. The firmware will run a Power-On System Test (POST) to check the hardware.
  3. The firmware will pass control off to the operating system.
     - When the OS starts, it initializes a single process.

Process #1

- Every OS has an **init** process, the initial process on the system after boot.

- In Linux, there is only one primary way to create a new process:

  `fork()`

fork()

- **`fork()`** creates a new process by duplicating the calling process. The new process, referred to as the child, is a duplicate of the calling process, referred to as the parent.
  - Since **`fork()`** creates a new process, **`fork()`** returns twice! Return value:
    - **0**: child process
    - **>0**: parent process (returns the ID of the child)

Example #1

```
void main() {
    pid_t pid = fork();
    if ( pid == 0 )
        printf("Child process\n");
    else
        printf("Parent process\n")
}
```

Example #2

```
void main() {
    int i = 0;
    for (i = 0; i < 3; i++)
        fork();

    printf("Illinois\n");
}
```

Example #3

```
void main() {
    int i = 0;
    for (i = 0; i < 3; i++) {
        fork();
        printf("%d\n", i);
    }

    printf("Illinois\n");
}
```

Example #4

```
void main() {
    int i = 0;
    for (i = 0; i < 10; i++)
        if (fork())
            break;
}
```