Multiprocessing: limited direct execution; system calls; processes

COSC 208, Introduction to Computer Systems, 2021-11-08

Announcements

- Project 2 Part B due tomorrow

Outline

- System calls
- Creating processes: actual code & `fork`

Warm-up: true or false

Q1: *True or False*

1. Code stored on secondary storage (e.g., a solid state drive) is called a process
2. Each process has its own code, heap, stack, and register values
3. The CPU is in user mode when executing application code, and kernel mode when executing OS code
4. A process can directly execute instructions on the CPU
5. A process can directly access input and output ports

System calls

- Example program

```c
#include <stdio.h>
#include <unistd.h>
int user() {
    int uid = getuid();
    return uid;
}
int main() {
    int u = user();
    printf("User %d is running this process\n", u);
}
```

- Assembly code

```
00000000004006ac <user>:
    4006ac: d10083ff    sub sp, sp, #0x20
    4006b0: f9000bfe    str x30, [sp, #16]
    4006b4: 94007713    bl  41e300 <__getuid>
    4006b8: b9000fe0    str w0, [sp, #12]
    4006bc: b9400fe0    ldr w0, [sp, #12]
    4006c0: f9400bfe    ldr x30, [sp, #16]
    4006c4: 910083ff    add sp, sp, #0x20
    4006c8: d65f03c0    ret
000000000041e300 <__getuid>:
    41e300: d28015c8    mov x8, #0xae
    41e304: d4000001    svc #0x0
    41e308: d65f03c0    ret
```

Creating processes

Q2: *What does the following code output?*

```c
int main(int argc, char **argv) {
    printf("Before fork\n");
    int pid = fork();
    printf("After fork\n");
    return 0;
}
```

Q3: *What does the following code output (assuming the new process has PID 1819)?*

```c
int main(int argc, char **argv) {
    printf("Before fork");
    int pid = fork();
    if (pid == 0) {
        printf("Child gets %d\n", pid);
    } else {
        printf("Parent gets %d\n", pid);
    }
    return 0;
}
```