Multiprocessing: processes; fork

COSC 208, Introduction to Computer Systems, 2022-04-19

After reviewing the `Process abstraction` in the notes, please complete the 2 warm-up questions.

Warm-up

Q1a: *Consider building a Lego kit as an analogy for operating systems' process abstraction. Match each component of the analogy with the corresponding component of a real computer system.*

- Analogy
  - Cabinet/drawers for storing Legos
  - Lego bricks
  - Building area (e.g., tabletop)
  - Instruction booklet
  - Following the assembly instructions
  - Current step for the instruction booklet
  - Completed kit
  - You
- Real system
  - CPU
  - persistent storage
  - process
  - program
  - program counter
  - program inputs
  - program outputs
  - registers and main memory

Q1b: *Complete each statement with 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

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;
}
```