Multiprocessing: wait; exec; non-preemptive scheduling

COSC 208, Introduction to Computer Systems, 2021-11-10

Outline

- Warm-up
- Waiting for processes
- Running a different program
- Scheduling processes
- First In First Out (FIFO) scheduling
- Shortest Job First (SJF) scheduling

Warm-up

Q1: *What does the following code output?*

```
int main(int argc, char **argv) {
    int value = 100;
    int pid = fork();
    if (pid == 0) {
        value -= 50;
    } else {
        value += 50;
    }
    printf("My value is %d\n", value);
    return 0;
}
```

Q2: *What does the following code output?*

```
int main(int argc, char **argv) {
    printf("Begin\n");
    int pid = fork();
    if (pid == 0) {
        printf("Child\n");
        return 0;
    } else {
        printf("Parent\n");
    }
    printf("End\n");
}
```

🛑 **STOP HERE** after completing the warm-up; if you have extra time take a few deep breaths to help reduce stress.

Waiting for processes

Q3: *What are all possible outputs of this program?*

```
int main() {
    int pid = fork();
    if (pid == 0) {
        printf("Child\n");
    } else {
        wait(NULL);
        printf("Parent\n");
    }
    return 0;
}
```

Q4: *What are all possible outputs of this program (assuming the new process has PID 13346)?*

```
int main() {
    int pid = fork();
    printf("A %d\n", pid);
    if (pid == 0) {
        printf("B\n");
    } else {
        wait(NULL);
        printf("C\n");
    }
}
```

🛑 **STOP HERE** after completing the above questions; if you have extra time take a few deep breaths to help reduce stress.

Running a different program

Example program

```
int main(int argc, char **argv) {
    printf("Begin\n");
    int pid = fork();
    if (pid == 0) {
        printf("Child\n");
        char *cmd[] = { "/usr/bin/date", NULL };
        execv(cmd[0], cmd);
    } else {
        printf("Parent\n");
    }
    printf("End\n");
    return 0;
}
```

Q5: *What is the output produced by running `./progA`, assuming no errors occur?* **progA:**

```
int main() {
    pid_t a = fork();
    if (a == 0) {
        execv("./progB", NULL);
        printf("A 2nd gen\n");
        return 0;
    } else {
        wait(NULL);
        printf("A 1st gen\n");
        return 0;
    }
}
```

progB:

```
int main() {
    pid_t b = fork();
    if (b == 0) {
        printf("B 2nd gen\n");
        return 0;
    } else {
        wait(NULL);
        printf("B 1st gen\n");
        return 0;
    }
}
```