CIS 4480/5480 Recitation 0 - Processes, Valgrind and Style

Welcome to recitation!!! 😃

Exercise 1: GDB Debugging

You are given a file called `pokemon_buggy.c`. This program looked like it should print some info about a Pokémon… but it crashed or printed nonsense! Use your GDB and debugging skills to answer the following questions.

Bug #1:

a) What was the first bug you found and fixed?

b) What line caused the crash?

c) Why did the program crash at that point?

d) What did you change to fix it?

Bug #2

a) What was the second bug you encountered after fixing the first?

b) Did the program crash again, or just behave strangely?

c) How did you use gdb to identify the problem?

d) Where was the invalid access? What caused it?

Exercise 2: Processes and File Access

```c
#include <fcntl.h>
#include <stdlib.h>

int main() {
  pid_t child = fork();
  int fd = open("file.txt", O_WRONLY);
  if (fd == -1) {
    exit(EXIT_FAILURE);
  }
  write(fd, "this is parent or child.", 25);
  close(fd);
  return 0;
}
```

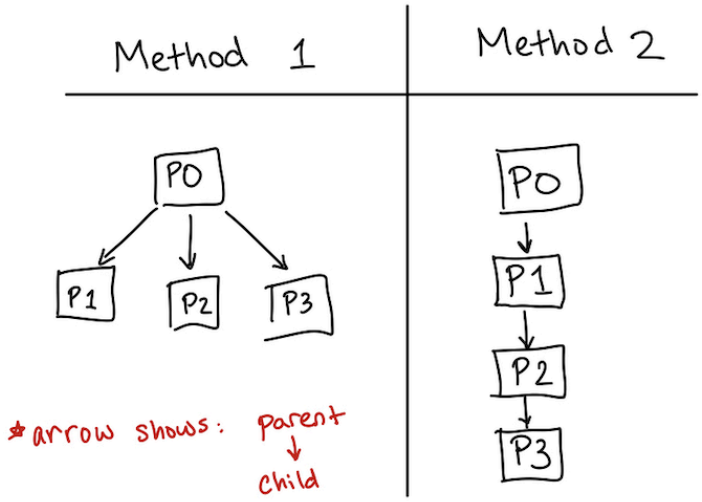
Questions to answer:

- Which processes have access to `file.txt`?
    a) Parent
    b) Child
    c) Both
    d) Neither

- If the parent closes the file, can the child still write to `file.txt`? **Explain your answer**.

Exercise 3: The Process Family Tree

Here are two diagrams, where each labeled box represents a process. P0 is the "original process" that forks P1. Arrows show the parent-child relationship. The order of processes spawning from first to last is: P0, P1, P2, P3.



Questions to answer:

- Using either C code, psuedocode, or a written description, describe how you would fork 3 processes to achieve diagram 1 and diagram 2.

Diagram 1	Diagram 2

- Let's say I have 3 independent tasks: T1, T2, and T3.
    - P1 will exec T1
    - P2 will exec T2
    - P3 will exec T3
    - T1, T2, and T3 all require I/O calls to be made (i.e. reading from or writing to a file)
    - P0 must wait until T1, T2, and T3 have finished.

  Which diagram will result in the faster runtime? Explain your answer.

Exercise 4: Waiting

```c
int main(void){
  int level_1 = fork();
  if (level_1 == 0) {
    int level_2a = fork();
    if (level_2a == 0) {
      printf("A");
    } else {
      wait(NULL);
      printf("B");
    }
  } else {
    int level_2b = fork();
    if (level_2b == 0) {
      printf("C");
      exit(0);
    }
    printf("D");
  }
  printf("0");
  return (0);
}
```

Questions to Answer:

1. Draw a diagram of all processes and clearly indicate all parent-child relationships. You may model your diagram after the one shown in Exercise 2, if you would like.

2. Which of the following are possible outputs? Select all that apply:
   a. B0AC0D0
   b. D0CA0B0
   c. D0A0B0C
   d. CAD00B0
   e. ABCD000

Exercise 5: Exit Questions

From 1-5, answer the following:

How fast is the recitation pacing?

(not fast) 1 2 3 4 5 (fast)

How helpful is the recitation lecture portion?

(not helpful) 1 2 3 4 5 (helpful)

How helpful is the recitation worksheet portion?

(not helpful) 1 2 3 4 5 (helpful)

Would you like to see more practice, or more content-review?

- Content review!
- Practice!
- There's a good balance of both!

Any feedback?

Mark any topics you would like to see/practice next week

- Wait versus Waitpid
- Masks
- Handlers
- Pipes
- File descriptors
- Debugging (GDB or Valgrind)
- Other: _____