

## Lab 9 Worksheet: Shall we execute some subprocesses?

### 1. Call Me Maybe

#### Part 1A

When a program performs a system call, several steps need to be taken as control transfers between the user space and the operating system kernel. Given the following steps, please arrange them in the **correct order of execution**.

- Switch to kernel space
- Return to instruction after the system call
- Handler finishes
- Application makes a system call (ecall on RISC-V)
- Switch to user space
- Trap handler runs

#### Part 1B

Suppose that we have a long-running blocking system call, like `write` or `waitpid`. Let's say our program is interrupted by `Ctrl + C` (a `SIGINT`), while the system call is running. What should the kernel do? Should the system call fail immediately, restart automatically, or something in between? What are the tradeoffs? Does the "right answer" change if we receive a different type of signal?

### 2. Fork or be Forked

Assume the **parent process's PID is 5** and that when the `fork` call succeeds, the **child process is assigned PID 10**. Please fill in the results of the print statements for parts (a) and (b) and answer part (c):

```
// omitted imports
int main(void) {
    pid_t pid = fork();
    if (pid) { printf("A: pid = %d\n", pid); } (a) "A: pid = _____"

    else { printf("B: pid = %d\n", pid); }      (b) "B: pid = _____"
    wait(NULL);
    printf("C: pid = %d\n", pid);              (c) How many line C's are printed? _____

                                                Is there an order? _____

    return 0;
}
```

## Lab 9 Worksheet: Shall we execute some subprocesses?

### 3. Caught Napping

#### Part 3A

Examine the following partial implementation of `myint`, which sleeps for  $n$  seconds (in 1-second chunks) before sending `SIGINT` to itself. Please complete the main function.

```
// omitted imports
int main(int argc, char **argv) {
    for (int i = 0; i < atoi(argv[1]); i++) { (a)_____ (1); }
    kill((b)_____, (c)_____);
    exit(0);
}
```

#### Part 3B

If we insert `pid_t pid = fork();` before the `for` loop, how do we modify the program such that `SIGTERM` is sent to the child process? What is the exit code of the main program?

#### Part 3C

Explain briefly the difference between the `SIGINT`, `SIGTERM`, and `SIGKILL` signals.

### 4. Reaping Child Processes

Examine the following program and answer the following questions:

```
int main() {
    int status;
    pid_t pid = fork();
    if (pid == 0) {
        exit(3);
    }
    waitpid(pid, &status, 0);
    if (WIFEXITED(status)) {
        printf("Exited");
    } else if (WIFSIGNALED(status)) {
        printf("Terminated");
    }
    return 0;
}
```

What will be printed when this program is run?

- A. "Exited"
- B. "Terminated"

Suppose `exit(3)` is replaced with `abort()`. What will be printed now?

- A. "Exited"
- B. "Terminated"

Suppose the child crashes due to an invalid memory access (segmentation fault). What will the program print?

- A. "Exited"
- B. "Terminated"