# Lab 10 Worksheet: Shall we execute some commands?

## 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****.*

A. Switch to kernel space
B. Return to instruction after the system call
C. Handler finishes
D. Application makes a system call (ecall on RISC-V)
E. Switch to user space
F. Trap handler runs

### Part 1B

*Compare* ***traps****,* ***interrupts****, and* ***system calls*** *– how are they similar and how do they differ?*

## 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); }
 else { printf("B: pid = %d\n", pid); }
 wait(NULL);
 printf("C: pid = %d\n", pid);

 return 0;
}
```

**(a) "A: pid = _____"**

**(b) "B: pid = _____"**

**(c) How many line C's are printed? _____**

**Is there an order? _____**

CS 3410 Fall 2025

# Lab 10 Worksheet: Shall we execute some commands?

## 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”