

CSC373 Worksheet 2

August 16, 2020

Source: [link](#)

1 Homework (Code)

1. Write a program that calls `fork()`. Before calling `fork()`, have the main process access a variable (e.g., `x`) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of `x`?
2. Write a program that opens a file (with the `open()` system call) and then calls `fork()` to create a new process. Can both the child and parent access the file descriptor returned by `open()`? What happens when they are writing to the file concurrently, i.e., at the same time?
3. Write another program using `fork()`. The child process should print “hello”; the parent process should print “goodbye”. You should try to ensure that the child process always prints first; can you do this without calling `wait()` in the parent?
4. Write a program that calls `fork()` and then calls some form of `exec()` to run the program `/bin/ls`. See if you can try all of the variants of `exec()`, including (on Linux) `execl()`, `execle()`, `execlp()`, `execv()`, `execvp()`, and `execvpe()`. Why do you think there are so many variants of the same basic call?
5. Now write a program that uses `wait()` to wait for the child process to finish in the parent. What does `wait()` return? What happens if you use `wait()` in the child?
6. Write a slight modification of the previous program, this time using `waitpid()` instead of `wait()`. When would `waitpid()` be useful?
7. Write a program that creates a child process, and then in the child closes standard output (`STDOUT_FILENO`). What happens if the child calls `printf()` to print some output after closing the descriptor?
8. Write a program that creates two children, and connects the standard output of one to the standard input of the other, using the `pipe()` system call.