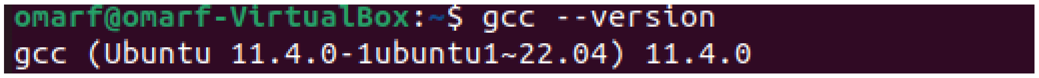
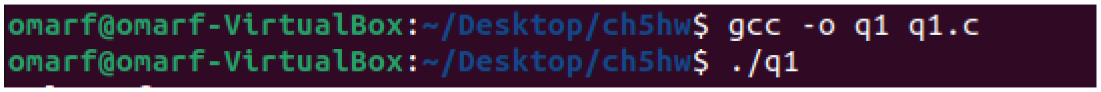
Omar Facundo
CSCD 340

Ch 5 Process API



```
omarf@omarf-VirtualBox:~$ gcc --version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
```

All programs were run using these same commands and followed the same naming convention.



```
omarf@omarf-VirtualBox:~/Desktop/ch5hw$ gcc -o q1 q1.c
omarf@omarf-VirtualBox:~/Desktop/ch5hw$ ./q1
```

1. When you use fork(), it creates a copy of the parent process. Since both child and parent processes have their own address, they can't interfere with each other, so they both keep their own copy of the variable.
2. Both parent and child processes can access the file descriptor when using open(). Both can also write to the file. When running to a file concurrently, one process will write and whichever process happens next, it will overwrite the first process write.
3. It can be done without calling wait. One way it can be done is by using a loop that doesn't do anything but lasts a good amount of time in the parent process.
4. Parachute problem.
5. wait() returns the PID when it is successful and returns -1 if it fails. Using wait in the child process will return -1 because there is no wait process that can happen in the child.
6. waitpid() is a more specific wait(). It lets you wait for a specific child process instead of waiting for all of them to finish.
7. STDOUT FILENO closes the file descriptor, so the child wont be able to use print but no error will occur.
8. Parachute problem.