NYU, Tandon School of Engineering
Extended Bridge to CS
Cherron Griffith

Homework 19
February 28, 2025

Unix Fork Function

The `fork()` function in UNIX is used to create a new process by duplicating the process on which it was called. The process that was duplicated is called the **parent process**, and the newly created process is called the **child process**. The child process is an almost exact copy of the parent process at the moment `fork()` is executed. The child process receives a new entry in the OS process table, with its own unique Process ID (PID), and it inherits a copy of the parent process' memory space. Once `fork()` is finished executing, it returns different values based on the processes. `fork()` returns a positive number in the parent process to represent the PID of the child process, and returns a `0` in the child process. If the process creation fails, `fork()` returns `-1`.

When `fork()` is executed, the processes will go through five states, depending on scheduling and system behavior. Initially, the parent process is most likely in the **ready** state. The child process would start in the **new/created** state, since it has been generated by the system call but has not yet been scheduled to run. Immediately after `fork()` is called, both the parent and child processes enter the **ready** state while they wait for the CPU to begin execution.

Once execution begins, the state of the processes depends on whether the parent process calls the `wait()` function on the child process. If `wait()` is not called, both the parent and child processes can be in the **running** state or **ready** state, depending on the OS scheduler. The child process will eventually enter the **exit/terminated** state when it completes execution. If the parent process calls `wait()`, it moves to the **blocked/waiting** state until the child process terminates (reaches exit/terminated state). In this case, the child process runs independently in the **running** state.

When the child process finishes execution, it first enters the **exit/terminated** state until the parent process retrieves its exit status. Once the parent successfully collects the exit status using `wait()`, the child process is completely removed from the process table, and the parent process continues execution in the **running** state.

Citations

---

[1] GeeksforGeeks, "fork() in C," GeeksforGeeks, Jun. 16, 2015. https://www.geeksforgeeks.org/fork-system-call/ (accessed Feb. 24, 2025).

[2] M. Kerrisk, "fork(2) - Linux manual page," man7.org, Feb. 02, 2025. https://man7.org/linux/man-pages/man2/fork.2.html (accessed Feb. 24, 2025).

[3] A. S. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed. Harlow: Pearson Education, 2015.