Kohl Johnson

Activity 3

06-23-2024

## More Signals in Linux

**Theory of Operation:** The code works by having the parent process start with a message, write it to the shared buffer, sends a signal to the child, and then moves onto the next character. Once all characters have been sent to the buffer, the parent then sends the child a null character which lets it know that the entire message has been sent. The child process works by first waiting for a signal from the parent, and then reads the character from the lower value of the buffer, prints to the console, and then restarts that process over until the null character is received.

**Screencast Link:** <https://www.loom.com/share/7ca5f75e5ab84746b02bc5e8e27fb47b?sid=f995b963-f310-4d8d-ab4b-c4bbf68e5c64>

## More Signals and Mutexes in Linux

**Theory of Operation:** We start off by creating two threads, one being a counter thread and the other a monitor. The counter thread will lock the counter (using a mutex), increment the counter, and then sleep for 1 second before unlocking the mutex to give the monitor thread a chance to have a “miss”. The monitor will attempt to lock the mutex every three seconds. If successful, it will print the value of the counter to the console, otherwise it will increase a separate counter called “misses” and restart the process.

**Screencast Link:** <https://www.loom.com/share/73bbb4d49c2c4ac19985db3dd3fe143f?sid=a392aca5-109b-4020-8591-025857a1acbe>

## More Signals and Semaphores in Linux

**Theory of Operation:** The child process works by simulating a very long (or hung) process to give the parent process a chance to check for a hung process, and then to decide whether to kill it. The parent thread will first wait a couple seconds before calling a timer function. From there, the timer counts down from 10 before attempting to lock the semaphore. If it succeeds, the timer function returns and ok status otherwise it returns 0. When 0 is returned, the parent process kills the active child, then proves it is killed by waiting 5 seconds to check for output from the child. Lastly, it attempts to secure the semaphore before exiting.

**Screencast Link:** <https://www.loom.com/share/63919caaf6714c5bb28034b44a131e0c?sid=23fb9c50-be1f-49ed-8bf9-f53e7b3a586e>