Kevin Jiang

ECE357

October 21st, 2020

Problem Set 4

1. Problem 1
    a. Press Ctrl+\ to generate a SIGQUIT signal, which dumps core after termination of process
    b. When a background process needs to output something to the user's terminal, SIGTTOUT will be sent to the foreground process. The foreground process is frozen and detached from terminal, unfrozen and reattached to the terminal after the background process is done outputting.
    c. Since this signal is a real-time signal, the signal will be queued 3 times. After B unblocks the signal, the signal handler will be called 3 separate times.
    d. The initial call to setjmp will return 0. The if statement will not run. 0 is not a valid address in the process memory, so SIGSEGV will be sent to the process when it reaches *p--. The signal handler will run, printing out "In handler instance 1" and then calling longjmp, which brings the program back to where setjmp was called. This time the if statement will run, and the program will print "The other side". When *p++ is reached, SIGSEGV is sent to the process again. However, because SIGSEGV is still blocked for this process, instead of calling the signal handler, a termination with core dump will occur.
2. Problem 2
    a. No, because the writes to the pipe are always less than 4K, only 1K actually. So the writes of "AAAA….." and "BBBB….." will be atomic, so there is no way for "ABA" to appear. And since the pipe is FIFO, the bytes of "A" and "B" with not be mixed
    b. If that line is removed, the program will print out "AAAA…..BBBB….." but never return 0, because the read system call will block until there is new data written into the pipe. Without the close, there will be no EOF written into the pipe to allow the read system call to return 0 and end the while loop.