Blocking Pipe

Consult the submit server for deadline date and time

1 Overview

Alter your pipe to block on reading and writing. Do not return EWOULDBLOCK.

Limit the buffer to 8192 bytes. A write of 8192 bytes into an empty buffer should not block. A write of 8193 bytes should.

Writes will not be partial. That is, a 32K write should return 32768 when the reader reads at least 24K of data. No test will write more than 32768 bytes at a time.

Reads will return a number of bytes equal to the min between the buffer size provided by the user and the amount of data available in the buffer.

A waiting reader or writer should awaken if the other side closes. The reader will return zero for the end of file, the writer should return EPIPE (regardless of how many bytes have been successfully written before the reader died). If signal functionality is present, SIGPIPE should be sent.

2 Key Functions

Use the condition variable functions in synch.h and synch.c: Cond_Wait(), Cond_Signal(), and Cond_Init(). They operate exactly like condition variables described in class, and associate with a mutex, meaning that Mutex_Lock(), Mutex_Unlock(), and Mutex_Init() will be relevant.

3 Tests

The `blkpipe.c` has three modes:

1. "bigwrite" writes 32K from the child, reads in 256 byte increments, ensures that data are read and that the writer finishes before the readers.
2. "littlewrite" writes one byte at a time, reads up to 256 at a time.
3. "dualwrite" writes from two processes, looking for race conditions.

The modes can be accessed by integer argument to blkpipe, e.g., "blkpipe 0" for bigwrite. If parent and child, and optional grandchild, are "happy", the test is successful.

There may be secret tests, likely to cover the Close() functionality.

4 Hints

You will need two condition variables and a mutex; This should be a roughly 20 line modification to a working pipe implementation for the synchronization, maybe a bit more for setting the buffer size limits as needed.

Close all on Exit. In developing this exercise, I had to alter the solution's Exit() function to Close all open file descriptors. Although this may not be necessary for a working solution to pass the three public tests, while developing, if a child process exits because of an assertion failure, the parent might hang because it will not realize that the pipe has no writers.