```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include <errno.h>
#include <string.h>
#include "fifo.h"

#define WRITERS 5
#define NUMITR 200

int procNum;
pid_t *pid_table;

void throwError(char *message, char *file)
{
    if (file)
        fprintf(stderr, "%s [%s]: Error code %i: %s\n", message, file, errno, strerror(errno));
    else
        fprintf(stderr, "%s\n", message);
    exit(-1);
}

int main(int argc, char **argv)
{
    struct fifo *f;
    int i, j;
    unsigned long datum;
    FILE *writes = fopen("writes.txt", "w"), *reads = fopen("reads.txt", "w");
    if (writes == NULL || reads == NULL)
        throwError("Error: Unable to open reads and writes log", NULL);

    f = (struct fifo *)mmap(NULL, sizeof(struct fifo), PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    pid_table = (pid_t *)mmap(NULL, ((sizeof(pid_t)) * NUM_PROC), PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);

    if (f == MAP_FAILED)
        throwError("Error: Failed to mmap", NULL);

    fifo_init(f);

    // MAKE WRITER PORCESSES
    for (i = 0; i < WRITERS; i++)
    {
        if ((pid_table[i] = fork()) < 0)
            throwError("Error: Failed to fork process.", "Writer");
```

```
        else if (pid_table[i] == 0)
        {
            pid_table[i] = getpid();
            procNum = i;

            for (j = 0; j < NUMITR; j++)
            {
                datum = pid_table[i] * 10000 + j;
                fifo_wr(f, datum);
                printf("WRITE %lu by PID: %d\n", datum, pid_table[i]);
                fprintf(writes, "%lu\n", datum);
            }
            return 0;
        }
    }

    // MAKE SINGLE READER PROCESS
    if ((pid_table[WRITERS] = fork()) < 0)
        throwError("Error: Failed to fork process.", "Reader");
    else if (pid_table[WRITERS] == 0)
    {
        pid_table[WRITERS] = getpid();
        procNum = WRITERS;

        for (i = 0; i < (WRITERS * NUMITR); i++)
        {
            datum = fifo_rd(f);
            printf("READ %lu by PID: %d\n", datum, pid_table[WRITERS]);
            fprintf(reads, "%lu\n", datum);
        }
        return 0;
    }

    for (i = 0; i < (WRITERS + 1); i++)
    {
        if (waitpid(pid_table[i], NULL, 0) < 0)
            throwError("Error: Unable to wait for child process to complete",
NULL);
    }
    return 0;
}
```