filewrite.c

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

int main(int argc, char ** argv)
{
    char* fname;
    int file_descriptor;
    if (argc != 2) {
        fprintf(stderr, "usage: filewrite filename\n");
        exit(1);
    }
    fname = argv[1];
    file_descriptor = open(fname, O_WRONLY | O_CREAT);
    if (file_descriptor < 0) {
        fprintf(stderr, "Major problem, dude\n");
        exit(1);
    }
    /* this is the interesting line */
    write(file_descriptor, "this text says nothing at all\n", 30);
    /* cleanup (why do we need this?) */
    close(file_descriptor);
    exit(0);
}
```
our_head.c -- a C program that prints the first L lines of its input,
where L defaults to 10 but can be specified by the caller of the
program.

(This is not an example of good C coding. For example, it is
inefficient. Also, the program does not check many error
conditions.)

#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

int main(int argc, char** argv)
{
    int i = 0;
    int nlines;
    char ch;
    int ret;

    /* next lines say:
     * if user supplied an argument, i.e., the # of lines, put that
     * argument into the 'nlines' variable.
     * if user did not supply an argument, set the # of lines to be 10
     */
    if (argc == 2) {
        nlines = atoi(argv[1]);
    } else if (argc == 1) {
        nlines = 10;
    } else {
        fprintf(stderr, "usage: our_head [nlines]\n");
        exit(1);
    }

    /* we'll loop for a maximum of 'nlines'. */
    for (i = 0; i < nlines; i++) {
        do {
            /* read in the first character from file_descriptor 0 */
            ret = read(0, &ch, 1);
            /* if there are no more characters to read, then exit */
            if (ret == 0) exit(0);
            /* what does this line do? .... */
            write(1, &ch, 1);
            /* .... what about this next line? */
            while (ch != '\n') {
                write(1, &ch, 1);
            }
        } while (ch != '\n');
    }
    exit(0);