References

- `http://en.wikipedia.org/wiki/C_file_input/output`
- C programming resources (course website)

Exercise 1. In class, we saw how to write a C program which displays the contents of a specified file. (See the next page for code details.) This program illustrates the following ideas:

- The use of the `argc` and `argv` data structures.
- The use of a few I/O functions from the Standard C Library.

Modify this program so it acts a bit more like the Unix `cat` utility. When your program is completed, it should be capable of acting on one or more file names given on the command line. Your program should perform error checking on the file names given; output an error message for each file name which cannot be opened.

Put your program in a file named `mycat.c` and compile it with the following command:

```
gcc -ansi -Wall -o mycat mycat.c
```

This command will verify your program meets the ANSI standards for C, will give you potentially useful warnings about your program, and create an executable file named `mycat`.

Exercise 2. Using the provided program as a guide, implement a program named `mycp.c` which will copy the contents of a source file to a destination file, thus acting like a "poor man's" Unix `cp` utility. (This exercise is similar to Problem 2.27 from the dinosaur book, except we will use functions from the Standard C Library.) Compile your program with the following command:

```
gcc -ansi -Wall -o mycp mycp.c
```

What to Submit

Put copies of your two completed C programs into a folder, then drag this folder onto the EIU submit icon. Be sure that your name appears in each program as a comment somewhere near the top of the file. Your programs should be generously commented.

Source Code

```
/*
  Poor man's cat --- display the contents of one file

  Usage:  pmcat filename

  MAT 4970
  Bill Slough
  February 7, 2012
*/

#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        /* Exactly two command-line arguments are expected; complain and quit. */
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        return 1;
    }
    else {
        /* Is the provided filename really a file? */
        FILE *in = fopen(argv[1], "r");
        if (in == NULL) {
            fprintf(stderr, "%s: %s: Could not open.\n", argv[0], argv[1]);
            return 2;
        }
        else {
            /* OK; everything looks good ---
               process the file character by character */
            int ch;
            while ((ch = fgetc(in)) != EOF) {
                printf("%c", ch);
            }
            fclose(in);
            return 0;
        }
    }
}
```

Listing 1: A C program to display the contents of a file.