# Homework 3: Unix File I/O

**Due**: Monday 21 October 2013, 23:59:00 (11:59 PM) Pacific USA time zone.

Points on this assignment: 75 points with 5 bonus points available.

Work submitted late will be penalized as described in the course syllabus. You must submit your work twice for this and all other homework assignments in this class. Ecampus wants to archive your work through Blackboard and EECS needs you to submit through TEACH to be graded. If you do not submit your assignment through TEACH, it cannot be graded (and you will be disappointed with your grade). Make sure you submit your work through TEACH. Submit your work for this assignment as a *single tar.bzip file* through TEACH. The same single tar.bzip file should also be submitted through Blackboard.

---

Place all of the files you produce for this assignment in a single directory, called `Homework3`. Do not put spaces in your directory names or filenames.

In this assignment, you will be working with Unix file I/O system calls. This assignment will also be only in C, with a bit of Makefile fun. This is a fairly short programming assignment in C. It also provides you with a good start to building the main portion of homework #4. This assignment is intended to be a building block for homework #4. This is the only assignment that is a building block for a following assignment.

In **all** your source files (including the Makefile), you need to have 4 things at the top of every file as comments:

1. Your name
2. Your email address (ONID or engineering)
3. The class name and section (this is CS311-400)
4. The assignment number (this is homework #3)

Remember that the programming work in this class is intended to be individual work, not group work.

1. **5 points**. When you are ready to submit your files for this assignment, make sure you submit a *single bzip file*. Review homework #1, problem #1 if you need a refresher on how to do this. If your file is not a single bzip file, you cannot receive points on this assignment.

2. **70 points**. Write a C program (`homework3.c`) to seek to various locations in a Unix file.

**50 points of the 70** total for this portion of the homework: Your C program will open a file in the file system. The file to be opened will be identified with the `-f` *`filename`* command line argument. Once opened, you will seek to the offset location identified by the `-o` *`offset1`* command line option (this is from the beginning of the file, a lower case o (oh)). From there, you will read a number of bytes from the file the number is identified by the `-l` *`len`* command line option and write those bytes to the console (followed by a newline character). After that, you will seek again using the value passed with the `-O` *`offset2`* command line option (this is an upper case O (Oh)). The seek done with *offset2* is relative to the current position in the data file (not from the beginning of the file like *offset1*). Again, read *len* number of bytes from the file and print them to the console. When run, your program will process the `-o` option before the `-O` option.

**For 5 points extra credit**, after processing the `-o` and `-O` options, print the last number of bytes from the file, based on the `-e` *`elen`* command line option.

| | |
|---|---|
| `-v` | Verbose, show all the other command line arguments and any other print statements you find useful. These statements should not show up if the –v option is not given. |
| `-f` *`filename`* | File name, the name of the file in the file system which will have its contents read. Can be any path to a file. |
| `-l` *`len`* | Length of bytes read, the number of bytes to be read and written to the console from the –o and –O options. The value of this parameter must be greater than 0. (20 points) |
| `-o` *`offset1`* | Offset from beginning of file, the offset into the named file from where a number of bytes will be read and written to the console. This is a lower case o (oh). The value for *offset1* must be greater than or equal to 0. (10 points) |
| `-O` *`offset2`* | Offset from current position in file where a number of bytes will be read and written to the console. This is an upper case O (Oh). The value for *offdset2* can be positive or negative. (20 points) |
| `-e` *`elen`* | **Extra credit 5 points**: the number of bytes from the end of the file that will be read and written to the console. |

Your program should not care the order in which the command line options are given. Do not expect to see the command line options in any specific order. Do not presume that the `-f` option precedes the `-l` option. Do not presume that `-o` precedes `-O` on the command line.

I have placed 2 files on `eos-class` that I will be using to test your code. They are in `/usr/local/classes/eecs/fall2013/cs311-400/src/Homework3`. The files are named `iliad.txt` and `websters.txt`. Both files are from Project Gutenberg. You do not need to make your own copies of these files on `eos-class`.

You can either specify the path or make symbolic links to them in your development directory.

When accessing the file, use open() to create a file descriptor (not fopen()). Use read() to read bytes from the file into a buffer (not fscanf()). When you write the contents of the buffer to the terminal, use write() (not printf()). When you output other text to the terminal (such as for verbose), go ahead and use printf(), just not on the buffers from the file.

To give you something to test, you can use the dd command to pull out a number of bytes form a file. You can use this command:

```
dd skip=1234 count=50 bs=1 if=<path to file>/iliad.txt
```

results in:

> n common. Even in these,
> however, a certain elasti

That will print the 50 bytes starting 1,234 bytes into the file.

You can also use tail to show the last bytes of a file (for the extra credit).

```
tail -c 100 <path to file>/iliad.txt
```

results in:

> help produce our new eBooks, and how to
> subscribe to our email newsletter to hear about new eBooks.

This is a pretty short assignment. You should need 100-150 lines of C code (maybe less). If you are going over that: you are checking for too many kinds of errors, putting in a LOT of comments, going way to far, or are lost. If you are struggling with this assignment, you really need to read chapter 4 in TLPI and spend some quality time with listing 4-3.

**20 Points of the 70** for this portion of the homework is for creating a Makefile to build your application. Your Makefile should include at least 4 targets:

| | |
|---|---|
| all | Builds all dependent C code modules for all applications in the directory. (5 points) |
| homework3 | Builds all dependent C code modules for the homework3 application in the directory. (5 points) |
| homework3.o | Builds the homework3.c module for the homework3 application in the directory, based off of any changed dependent modules. (5 points) |

| | |
|---|---|
| clean | Deletes all executable programs (homework3), object files (files ending in .o produced by gcc), and any editor droppings (# files from vi and ~ files from emacs). Make sure you use this before you bundle all your files together for submission. (5 points) |

The Makefile from question 7 in homework #1 should get you through this really easily. When I build your program, I should be able to just type `make` to have it completely build.

**Example output**

Here is a sample result from my code:

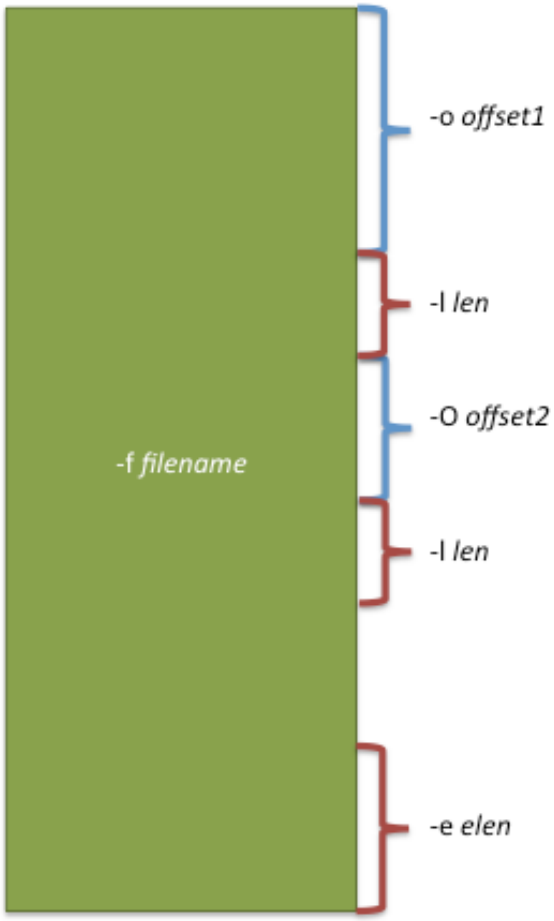
```
$ ./homework3 -f iliad.txt -o 4914 -O -372 -l 372 -e 500 -v
./homework3 -f iliad.txt -o 4914 -O -372 -l 372 -e 500

<offset 1> ----------------------------------------------------
Sing, goddess, the wrath of Achilles Peleus' son, the ruinous wrath that
brought on the Achaians woes innumerable, and hurled down into Hades
many strong souls of heroes, and gave their bodies to be a prey to dogs
and all winged fowls; and so the counsel of Zeus wrought out its
accomplishment from the day when first strife parted Atreides king of
men and noble Achilles.
<offset 2> ----------------------------------------------------
Sing, goddess, the wrath of Achilles Peleus' son, the ruinous wrath that
brought on the Achaians woes innumerable, and hurled down into Hades
many strong souls of heroes, and gave their bodies to be a prey to dogs
and all winged fowls; and so the counsel of Zeus wrought out its
accomplishment from the day when first strife parted Atreides king of
men and noble Achilles.
<end bytes> ----------------------------------------------------
he U.S.
unless a copyright notice is included.  Thus, we do not necessarily
keep eBooks in compliance with any particular paper edition.


Most people start at our Web site which has the main PG search facility:

     http://www.gutenberg.org

This Web site includes information about Project Gutenberg-tm,
including how to make donations to the Project Gutenberg Literary
Archive Foundation, how to help produce our new eBooks, and how to
subscribe to our email newsletter to hear about new eBooks.
----------------------------------------------------------------------
```

There are a couple things you should notice from the example above.

1. I put a line of dashes between chunks of text pulled from the file; it makes it easier to read. I did make use of printf() for those lines. I urge you to do the same.
2. The value for the –O parameter is the negative of the value for the –l parameter.

3. The value for the –e parameter is larger than the value for the –l parameter.
4. I used the –v parameter to print out the arguments passed on the command line.

The figure below provides a graphical representation of how the command line parameters control reading from the file (when all arguments are positive).




---

Things to include with the assignment (in a single bzipped file):

1. C source code for the solutions to the posed problem (all files).
2. A Makefile to build your code.

Please combine all of the above files into a single tar.bzip file prior to submission.