$Id: lab4c-stdio-getopt.mm,v 1.16 2016-01-12 17:30:40-08 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps012b-wm/Labs-cmps012m/lab4c-stdio-getopt
URL: http://www2.ucsc.edu/courses/cmps012b-wm/:/Labs-cmps012m/lab4c-stdio-getopt/

This lab will introduce you to the use of files in ANSI C, and scanning options from the command line. Use the command `man`(1) to read about various commands and functions that are mentioned in this document. For each function, the synopsis shows you what header needs to be included, and also displays the prototypes for relevant library functions.

## 1. Program Specification

The program specification is given in the format of a Unix `man`(1) page.

**NAME**

cmatch — print matching lines from files

**SYNOPSIS**

`cmatch` [`-iln`] *string* [*filename* . . .]

**DESCRIPTION**

`cmatch` searches the named input files for lines containing the string. By default the matching lines are printed. If more than one file is specified, lines of output are preceded by the filename.

**OPTIONS**

Options are scanned using `getopt`(3) and are subject to its restrictions and conventions.

`-i` Ignore case distinctions between the string and the contents of files.

`-l` Suppress normal output. Print only the names of files for which a match is found.

`-n` Prefix each line of output with the line number starting from 1 within each file.

**OPERANDS**

The first operand is required and is the string to be used to search the files. The rest of the operands are filenames to be processed in sequence. If no filenames are specified, stdin is read. If a file is specified as a single minus (`-`) stdin is read at that point.

**EXIT STATUS**

0 No errors were detected.

1 Errors were detected and messages printed to stderr, either for invalid options or inaccessible files.

**SEE ALSO**

`grep`(1)

## 2. Implementation Sequence

Following are suggestions as to implementation sequence, which may be followed more or less, but are not absolutely required if you find a better way.

(a) Study the behavior of `pmatch.perl`, which is an implementation of your lab in Perl.

(b) Create a build script called `mk` as you did with the previous project. This script uses

```
gcc -g -O0 -Wall -Wextra -std=gnu99
```

to compile `cmatch.c` into the executable binary `cmatch`. Note that an executable binary generally does not have a suffix.

(c) Whenever you see a reference to a function in C, read the man page for it. For example, the function `strstr`(3) can be discovered with the command

```
man -s 3 strstr
```

Note that in this case, the synopsis shows you what the arguments and results to the function is, and that you need to include the header `<string.h>`.

(d) The example `catbychar.c` shows how to read a file one character at a time.

(e) The example `catbyline.c` shows how to read a file one line at a time, assuming the maximum line length is known.

(f) A file is opened with `fopen`(3) and closed with `fclose`(3), unless it is `stdin`, which is already opened when the program begins.

(g) Error messages are printed to `stderr`, and `fflush`(3) should be called before and after printing a message in order to ensure that the buffers are cleared.

(h) The example `getoptex.c` shows how to use `getopt`(3) to scan options given to a program and record each as a boolean flag in a struct. Invalid options are handled.

(i) For `getopt`(3), the variable `opterr` allows us to print messages our own way, `optopt` is the option character actually found, and `optind` is the index into `argv` which is the first operand.

(j) Open a file called `cmatch.c`, which is mostly a copy of `catbyline.c`, and add the function from `getoptex.c` to ensure that options are scanned.

(k) For each line read, instead of just printing it, check it against the pattern string. Use either `strstr`(3) or `strcasestr`(3) as required by the options.

(l) Add in code to print filenames and line numbers as per the options, and make sure the exit status is correct.

### 3. What to Submit

Submit the file `cmatch.c` as per the specifications, and the build script `mk`. Also submit `PARTNER` if you do pair programming.