`$Id: lab4c-stdio-getopt.mm,v 1.1 2012-01-31 18:19:48-08 - - $`
`/afs/cats.ucsc.edu/courses/cmps012b-wm/Labs-cmps012m/lab4c-stdio-getopt`

1. Overview

This lab will again use the command line and `getopt`(3). In addition, you will open, read, and close files using the facilities in `stdio`(3). Use the `man`(1) command to read each of the man pages as they are mentioned in this lab.

2. Program Specification

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

NAME

bcat — concatenate and display files

SYNOPSIS

`bcat` [`-mns`] [*filename* . . .]

DESCRIPTION

The `bcat` utility reads files in sequence and copies each file to `stdout`, with options that control display attributes.

OPTIONS

All options precede all operands and are scanned via `getopt`(3). The following options are supported :

`-m` In the style of `more`(1), a title is printed in front of each file. A title consists of exactly 5 lines : an empty line, a line of 64 colons, a line with the name of the file (a minus sign (`-`) is used for `stdin),` a line of 64 colons, an empty line.

`-n` Line numbers are displayed to the left of each line in a field of width 6 followed by 2 spaces.

`-s` Multiple empty lines are squeezed into a single empty line. That is, if three or more consecutive newline characters (`\n`) appear on input, only the first two are copied. They are still counted, though, for the purposes of printing line numbers if the `-n` option is specified.

OPERANDS

Each operand is the name of an input file. If no filenames are specified, `bcat` reads from `stdin` If a filename is given as a minus sign (`-`), `stdin` will be read at that point. The file `stdin` is never closed and multiple occurrences are accepted without complaint.

EXIT STATUS

0 Normal successful completion.

1 An error has occurred. Program execution continues if possible in the presence of an error.

SEE ALSO

`cat`(1), `more`(1), `basename`(3), `errno`(3), `fclose`(3), `fopen`(3), `getc`(3), `getopt`(3), `putchar`(3), `strcmp`(3), `strerror`(3).

3. Lab Sequence

Following is a suggested implementation sequence.

(1) Note the chapter number in parentheses after a reference to a command or function. The notation `printf`(3), for example, means the function `printf` in section 3 of the manual.

```
man -s 3 printf
```

(2) After studying the programs provided, begin with `bcat.c` and add options analysis to it. Modify the function `catfile` so that it has additional parameters : the filename from `argv`, and a

reference to the options structure.

(3) Make `catfile` print out a file title before going into a loop if the `-m` option is specified.

(4) Make it print line numbers in `"%6d␣␣"` format if the `-n` option is specified. Note that there are exactly two spaces after the format. The symbol ␣ is a visible space; that is, a graphic which represents a space character. To do this you will need a flag, which is initially true. Immediately after reading a character and determinint that it is not `EOF`, if the flag is set, print the line number and turn off the flag. Every time you see a newline, turn it back on.

(5) The `-s` option can be implemented by making `catfile` keep a count of newlines just seen. Set the count to zero any time a non-newline character is read. Increment it any time a newline character is read. Suppress echoing of the newline whenever this count is more than 2. Be careful not to print line numbers for lines that are skipped.

(6) Modify `Makefile` to have a `submit` target, which submits all required files. Submit `Makefile` and `bcat.c`. The make file is suitable for single file programs, but not for modular programs. Make sure you submit your version of `bcat.c` and not the original one.

(7) Verify the submit. Ensure that you have submitted everything that needs to be submitted and that nothing has been submitted that should not be. Verify that neither `checksource` nor `lint` complains.

(8) It is acceptable for lint to make the following complaint:

```
function returns value which is always ignored
```

provided that the complaint refers to a function defined in one of the standard header files and not one of your files.

(9) Reading assignment: `/afs/cats.ucsc.edu/courses/cmps012b-wm/Coding-style/`

4. What to Submit

Submit `Makefile`, `bcat.c`, and `README` If you are doing pair programming also submit `PARTNER`