CS 240 - Homework 6

Assigned: Monday, 4.10.17	Due: Wednesday, 4.19.17, 4:00pm

Make a subdirectory "hw5" in your cs240 folder for this assignment and copy the files from /courses/cs240/s17/jmcurran/GROUP/hw5.

tail.c

The program takes lines from standard input and keeps the last n of them in memory as it goes through standard input. When it gets to an EOF, it prints the last n lines out.

You may assume:	The size of the tail (n) is less than 2000
	each individual line is no longer than 1000 characters including the newline and the null terminator

Overview

For this assignment, you will write two source files and a header file.

tail.c:	interprets the command line argument (this should be a dash followed by a number e.g., $ tail -5 if no argument is provided, default to 10)
	calls init_lineholder(int nlines) with the number from the command line option
	does a loop calling getline and insert_line(char *line)
	When getline returns 0 (indicating EOF on stdin), it calls print_lines().
lineholder.c:	contains a static array of pointers for lines
	implements init_lineholder, insert_line, and print_lines
	init_lineholder initializes the "first" slot and related variables
	insert_line adds a line to the array
	it must allocate memory for the new line
	it must free the memory for a line no longer needed, if any
	print_lines prints the lines in the array and frees the memory used for them
lineholder.h	prototypes for the three calls with appropriate comments explaining what they do for the caller (any comments on *how* they do it belong in lineholder.c.)

Write a makefile that compiles tail.c and lineholder.c, with the appropriate dependencies, and builds the executable tail. (tail should be the default target.)

Provide a make clean rule to remove the object files.

The default target should build the executable using the -m32 switch to generate a 32-bit application.

getline

Read the manual page for the getline function (part of stdio.h). You can access this from the command line:
$ man getline

You will use this function to read each line from standard input. Note that getline takes three arguments:

char **lineptr	a pointer to a pointer to a char the pointer to a char is a pointer to the memory location where the line being read in will be stored.
size_t *n	a pointer to a size_t variable holding the size of the block of memory where the line being read will be stored.
FILE *stream	We will cover file pointers later. For now, you can just specify standard input here by passing stdin.

getline is capable of allocating the memory to store the line for us, but for this assignment, I want you to allocate the memory.

We know that each line will be no more than 1,000 characters in length, so we can allocate that much space and store the pointer to it that malloc provides in a variable. Use the following names:

buffer	This is a char pointer to the allocated memory.
b_size	This is a variable of type size_t (this is an unsigned int type defined in stdlib) that holds the size of buffer.

Strategy

The challenge here is to hold the lines in memory efficiently. Only the last n lines should be held in memory, not all the lines. You should set up an array of char * pointers. Each pointer in this array of character pointers should point to one of the n lines you are storing.

Use malloc to allocate memory for each line you store. When you are done with a line, use free(pointer) to release the memory previously allocated with malloc. All the mallocs should be of just the right length to hold the line they are allocating memory for (whose length we know at this point -- remember that the return value of getline is the number of characters read, not including the terminating null character).

Be sure you explain your method of adding the n+1st line and freeing lines no longer needed in a good header comment at the beginning of lineholder.c. You have a choice of methods for adding the n+1st line to the array of n lines. Here are two ideas:

- Ripple the pointers to previous lines down one slot in the array and put the new one at slot n-1. First you need to free the one at slot 0 before it is pushed out of the array.
- Maintain a moving "first" slot variable. This variable stays at 0 for the first n line additions. When line n+1 is read, first moves to slot 1. Free the old line at slot 0 and put the new one there. When another line comes in, release the line at slot 1, put the new one there, and set first to 2. When you reach EOF and output, you'll need to wrap around from slot n-1 to slot 0 and proceed through first-1.

deliverables

Your hw5 folder should contain the following files:

- tail.c
- lineholder.c
- lineholder.h
- makefile

NOTE:	It is important that your files are thoroughly commented and you explain why you made the decisions you did and what each part of your code does. You need to talk about why you allocate/free memory where you do, how you are reassigning pointers as you read lines, etc. Your grade will be dependent on your ability to explain your design decisions.

You need to show your thought process.

Reminder

Homework is a solo effort. Any code you turn in must be your own. Never look at another student's code or show them your code.