UMass Boston CS 240
Homework 6
Due 11/20/2019 24:00

1 Tail

The assignment is K&R Exercise 5-13 on page 118. Make a subdirectory hw6 in your home directory for this assignment.

The program reads 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. You may assume n is less than 2,000, and each individual line has no more than 1,024 characters, including the newline and the end of string char.

The default n is 10. Write code using `getopt()` to process the command line option for n. For example, if the command is:

```
$ tail -n 20 < input.txt
```

your program should print the last 20 lines.

The challenge of this assignment is to hold the lines in memory efficiently. Use the template on page 108 to define an array of `char *` like this:

```
char *array[2000];
// array[0] is a pointer to the oldest line
// array[1] is a pointer to the second oldest line
// etc.
```

You can use one buffer that is long enough to hold the longest possible input line, like this:

```
char buffer[1024];
```

Instead of `getlines()` as in the book, you can use `fgets()` to read the input line. After you have one line, do the following:

1. Use `strlen()` to find out its actual length – remember to add 1 to `strlen()` so that there is space for the end-of-string char
2. Use `malloc()` to get enough space for a copy of the text
3. Use `strcpy()` to copy the text from `buffer` to the newly allocated memory
4. Save the pointer to the newly allocated memory in `array`

As you read more and more lines from standard input, you will fill the array of pointers. When it is filled, you discard the oldest line, which is in `array[0]`. Remember to call `free()` to free up the memory first. After that, you can ripple the other pointers up one place to make space: copy `array[1]` to `array[0]`, copy `array[2]` to `array[1]`, and so on, and copy `array[n - 1]` to `array[n - 2]`. Note that this copies the pointers – do not `strcpy()` the texts.