Coding Assignment 1

CSE3318

Your first coding assignment will be to write a program to do the following

- use conditional compile statements
- use command line parameters
- file handling
- linked list handling
- use the `clock()` function to time functions in your program
- use a provided executable to create large files

These are all skills/code you will need for later Coding Assignments. You will be reusing parts of this assignment in other assignments.

Create a program than can open a file listed on the command line, read through that file, write each line from the file into a linked list, print the linked list and free the linked list. You will time each step and count and sum during each step.

Linked Lists

I have taken my 3 lecture recordings on Linked Lists from my CSE 1320 class and combined them into 1 video. It is the recorded Teams lecture that goes with the Linked List slides.

The slides and the video are posted under Review Materials.

Naming your program and turning it in

Name your program `Code1_xxxxxxxxxx.c` where xxxxxxxxxx is your student id (not netid). My file would be named `Code1_1000074079.c`.

Please place the following files in a zip file with a name of `Code1_xxxxxxxxxx.zip` and submit the zip file.

`Code1_xxxxxxxxxx.c`

`TestFile.txt`

A zip file is used to avoid Canvas's file renaming convention.

Reminder – ANY compiler warning(s) or error(s) when compiled with a newer version of gcc will result in an automatic 0. This applies no matter what the warning/error is.

Creating your `TestFile.txt`

Download the file, `FileGenerator.e`, from Canvas to where you are compiling code on your PC.

The executable should **run** on your PC with no issues. Do NOT compile it – just **run** it.

YOU ARE NOT REQUIRED TO RUN THIS ASSIGNMENT ON OMEGA BUT YOU WILL BE REQUIRED TO RUN THE NEXT ASSIGNMENT ON YOUR PC AND OMEGA. Download the Omega specific version of `FileGenerator.e` for Omega from Canvas if you want to try it on Omega.

Run it and it will give you a message about what parameters it expects. Use the file generator to create a file of 10 records. Use this file for testing and submit in your zip as "`TestFile.txt`".

Create a function to read the file and call a function to add nodes to the linked list

Parameters : `argc`, `argv` and the linked list head

Return Value : `void`

Use `argv[1]` to get the filename from the command line if it is exists. If the program is not run with a file name on the command line, then print the message shown in the sample output and exit. If the program is run with a file name on the command line, then open the file with `"r"`. Use `fgets()` to loop through the file. See "File Handling in C" in the "Review Materials" module of Canvas. As each line of the file is read, call `AddNodeToLL()` to add the number read from the file as a node in the linked list. Count how many records you read from the file/added to the linked list. Sum up the numbers read from the file/added to the linked list. Print the count and the sum to the screen - see sample output.

Create a function to add a node the linked list

Parameters : number read from file, address of linked list head

Return Value : `void`

Use `malloc()` to create a new node. Set the node's `number` to the passed in number. Traverse to the end of the linked list and add the node.

Create a function to print the linked list

Parameters : linked list head

Return Value : `void`

Traverse the linked list and print out information from the node as shown in the sample output.

Create a function to free the linked list node memory

Parameters : linked list head

Return Value : `void`

Traverse the linked list and free the memory of each node. Count how many nodes are freed and sum up the values store in the node. Print both values to the screen as shown in the sample output.

Add the conditional compile define of `PRINT` around a print statement that prints the address of the node, the `number` stored in the node and the address stored in the node's next pointer. See sample output. Here's an example print statement.

```
#ifdef PRINT
printf("\nFreeing %p %d %p\n", TempPtr, TempPtr->number, TempPtr->next_ptr);
#endif
```

`main()`

Add the ability to accept command line parameters. The "Review Materials" module contains the CSE 1320 lecture on command line parameters. The file name that the program will read will be provided on the command line.

Your program should be able to read a file name from the command line. You will run your program as

```
Executable_name TestFile.txt
```

or

```
Executable_name MyOtherTestFile.txt
```

Add conditional compile statements around the call to the function that prints the linked list.

```
#ifdef PRINT
```

Call function to print linked list and `clock()` calls and the print statement for the number of clock ticks

```
#endif
```

See the sample output on how your program should behave when compiled with the `-D` option.

Using `TestFile.txt`, run your

```
gcc Code1_xxxxxxxxxx.c -D PRINT
```

Your program should be able to handle any size file – your program will be graded with a file of a different size.

Using the `clock()` function in `main()`

Add the code to time your function calls. To do this, define two variables of type `clock_t`

```
clock_t start, end;
```

You will call the `clock()` function to get the start time. You will call the `clock()` function again to get the end time after calling each function in `main()`.

```
start = clock();
```

Call function

```
end = clock();
```

print the total tics by subtracting `start` from `end` – see sample output

You will need to get the number of ticks used to call the function to read the file, the function to print the linked list and the function to free the linked list. Be sure to subtract the start time from the end time.

My sample test file – **your file will look different**

```
more TestFile.txt
74079
21259
59758
25398
4381
62060
61981
59743
21162
32541
```

Compiling and running the program without a conditional compile statement

```
gcc Code1_1000074079.c
./a.out TestFile.txt

10 records were read for a total sum of 422362

318 tics to write the file into the linked list

10 nodes were deleted for a total sum of 422362

7 tics to free the linked list
```

Compiling and running the program without a conditional compile statement and without a file name

```
./a.out
File must be provided on command line...exiting
```

Compiling and running the program with a conditional compile statement (please note that your output will have different addresses and different values)

```
gcc Code1_1000074079.c -D PRINT
./a.out TestFile.txt

10 records were read for a total sum of 422362

393 tics to write the file into the linked list

0x55e8dbeee490 74079 0x55e8dbeee4b0

0x55e8dbeee4b0 21259 0x55e8dbeee4d0

0x55e8dbeee4d0 59758 0x55e8dbeee4f0

0x55e8dbeee4f0 25398 0x55e8dbeee510

0x55e8dbeee510 4381 0x55e8dbeee530

0x55e8dbeee530 62060 0x55e8dbeee550

0x55e8dbeee550 61981 0x55e8dbeee570

0x55e8dbeee570 59743 0x55e8dbeee590

0x55e8dbeee590 21162 0x55e8dbeee5b0

0x55e8dbeee5b0 32541 (nil)

10 records were read for a total sum of 422362

136 tics to print the linked list

Freeing 0x55e8dbeee490 74079 0x55e8dbeee4b0

Freeing 0x55e8dbeee4b0 21259 0x55e8dbeee4d0

Freeing 0x55e8dbeee4d0 59758 0x55e8dbeee4f0

Freeing 0x55e8dbeee4f0 25398 0x55e8dbeee510

Freeing 0x55e8dbeee510 4381 0x55e8dbeee530

Freeing 0x55e8dbeee530 62060 0x55e8dbeee550

Freeing 0x55e8dbeee550 61981 0x55e8dbeee570

Freeing 0x55e8dbeee570 59743 0x55e8dbeee590
```

```
Freeing 0x55e8dbeee590 21162 0x55e8dbeee5b0

Freeing 0x55e8dbeee5b0 32541 (nil)

10 nodes were deleted for a total sum of 422362

90 tics to free the linked list
```

Compiling and running the program with a conditional compile statement but without a file name

```
gcc Code1_1000074079.c -D PRINT
./a.out
File must be provided on command line...exiting
```