**Part 2: Memset**

Memset is a function used to set every byte in a block of memory equal to a uniform value. It is useful because it cleans up the trash values that might interfere with your functions.

**Part 1:** code

Sorry it’s so small. I had some weird behavior in my program… Any struggles I had were noted with comment blocks/lines. I used calloc() instead of malloc() + memset(), but I didn’t use memset() with the reallocations, I’m curious now if that is what caused some of my weird file interactions.

I wanted to be able to open any file by name--- I tested it out and ran my practice runs on the file at path :

/home/s10/test.txt

/\*

this program behaves like it should as far as interaction, but the file handling is

causing me some trouble... when I delete a node extra lines will appear, and when i insert/read the nodes

there will be random instances of (null) appearing in the file--even replacing whole lines....

---- I might have been playing around with calloc and realloc a bit too much so maybe that's the reason,

or maybe not... idk why its doing these things.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stddef.h>

struct listNode {

char \*line;

struct listNode \*next;

};

typedef struct listNode Node;

typedef Node \*NodePtr;

//--------------------------------------------------------------------------------- function prototypes

void openFile(FILE \*\*filePtr,char \*\*namePtr, char\*\*text, struct listNode \*\*start);//-- check

void fprintScreen(char \*text);//---------------------------------------- check

void printScreen(struct listNode \*\*start);

void instructions(int \*instructionsPtr);//---------------------------------------- check

void getLineNum(int \*lineNum);//-------------------------------------------------- check

void \_delete(NodePtr \*start, int lineNum);/\*-------------------------------------- confusion?--

- supposed to delete a line, but when the list is written to the file after deletion there was another

line???? everything from before +1 (with random letters..."tuvw") im lost.

\*/

void printFile(NodePtr \*start, FILE \*\*filePtr, char \*\*text);//-------------------- check

void \_insert(NodePtr \*start, int lineNum);

//--------------------------------------------------------------------------------- main()

int main(){

int counter=0;

NodePtr start=NULL;

FILE \*filePtr=NULL;

char \*file\_text;

int lineNum;

int instructionsNum = 0;

char \*name;

/\* if((start= calloc(1,sizeof(struct listNode)))==NULL){

printf("%5sThere was a problem allocating memory for the first node (should probably restart program)\n","");

\*/

start= calloc(1,sizeof(Node));

while (instructionsNum!=7){

instructions(&instructionsNum);

switch (instructionsNum){

//-----------------------------------open file openFile()

case 1:

name=NULL;

start=NULL;

openFile(&filePtr, &name, &file\_text, &start);

/\* comment

[^^the above^^]

opens file bound to FILE \*filePtr,

stores the name at char \*name,

writes the contents to char \*file\_text (dynamically allocating

memory),

and \*/

break;

//----------------------------------print current file to screen (bfore edits) fprintScreen

case 2:

fprintScreen(file\_text);

break;

//-----------------------------------print list to screen (after edits) printScreen()

case 3:

printScreen(&start);

break;

//-----------------------------------write list to file printFile()

case 4:

printFile(&start, &filePtr, &file\_text);

break;

//-----------------------------------delete a line \_delete()

case 5:

getLineNum(&lineNum);

\_delete(&start, lineNum);

break;

//-----------------------------------insert/prepend/append a line (\_insert())

case 6:

getLineNum(&lineNum);

\_insert(&start,lineNum);

break;

}

}

while((\*start).next!=NULL){

free(start);

start=(\*start).next;

}

free(file\_text);

free(name);

fclose(filePtr);

return 0;

}

//--------------------------------------------------------------------------------- \_delete()

void \_delete(NodePtr \*start, int lineNum){

NodePtr prevNode=NULL;

NodePtr currNode= \*start;

NodePtr nextNode=(\*currNode).next;

int counter=0;

while (counter<lineNum){

prevNode=currNode;

currNode=nextNode;

nextNode= (\*currNode).next;

counter++;

}

free(currNode);

currNode=NULL;

(\*prevNode).next= nextNode;

}

//--------------------------------------------------------------------------------- \_insert()

void \_insert(NodePtr \*start, int lineNum){

NodePtr prevNode=NULL;

NodePtr currNode= \*start;

NodePtr nextNode=(\*currNode).next;

int counter=0;

NodePtr newNode=NULL;

char \*newLine;

char \*end\_marker;

size\_t size;

if ((newNode=calloc(1,sizeof(Node)))==NULL){

printf("%5sThere was a problem creating a node for the new line\n","");

}

while (counter<lineNum){

prevNode=currNode;

currNode=nextNode;

nextNode= (\*currNode).next;

counter++;

}

(\*prevNode).next=newNode;

(\*newNode).next=currNode;

if ((newLine=calloc(1,500))==NULL){

printf("%5sThere was a problem allocating memory for the new line.\n", "");

}

printf("%5sWhat would you like to insert at line %d:\n%10s","",counter+1,"");

fgets(newLine,499, stdin);

end\_marker=strchr(newLine,'\n');

size= end\_marker-newLine;

if ((newLine=realloc(newLine, size))==NULL){

printf("%5sThere was a problem modifying the size of the string.\n","");

}

else{

printf("%5sLine Added.\n","");

}

}

//--------------------------------------------------------------------------------- fprintScreen()

void fprintScreen(char \*text){

printf("%s", text);

// this method of printing works fine unlike below

}

//--------------------------------------------------------------------------------- printScreen()

void printScreen(struct listNode \*\*start){

NodePtr currNode= \*start;

while (currNode!=NULL){

printf("%s", currNode->line);

currNode= (\*currNode).next;

}

//this method of printing works up until the last line... it just prints (NULL)--- how do i fix that?

}

//--------------------------------------------------------------------------------- printFile()

void printFile(NodePtr \*start, FILE \*\*filePtr, char \*\*text){

NodePtr currNode= \*start;

unsigned int counter;

free(\*text);

if (((\*text)=calloc(1,sizeof(((\*currNode).line)-1)))==NULL){

printf("%5sthere was a problem generating an array.","");

}

while (currNode!=NULL){

while(counter<sizeof(((\*currNode).line)-1)){

\*(\*text+counter)= \*(\*currNode).line+(sizeof((\*currNode).line)-abs(counter-sizeof(\*text)));

// I spent a while trying to make this ^^ line work like so: " (\*currNode)->line+counter " but for some reason

// it didnt want to work, and I still dont understand why, just looking at it they look like they would do the same thing

counter++;

}

if ((currNode=(\*currNode).next)==NULL){

printf("%5sEvery Line has been successfully compiled into an array.\n","");

}

else {

if ((\*text=realloc((\*text),sizeof((\*text))+sizeof((\*currNode).line)))==NULL){

printf("%5sThere was a problem reallocating space for one of the lines\n","");

// ive been getting a lot of errors about the left operand being an lvalue---??

// takes a lot of fiddling to get it working, but i dont know how it's fixing it.

}

}

}

if ((fwrite(\*text, sizeof(\*text), 1, \*filePtr))==sizeof(\*text)){

printf("%5sthe file was successfully saved.\n","");

}

}

//--------------------------------------------------------------------------------- openFile()

void openFile(FILE \*\*filePtr,char \*\*namePtr, char \*\*text, NodePtr \*start){

unsigned long counter;

unsigned int counter\_ception;

char \*ntemp;

char \*new;

char \*new\_lScout=NULL;

char \*currArrPos=NULL;

NodePtr currNode=NULL;

NodePtr temp=NULL;

size\_t size;

size\_t f\_size;

char \*linePtr;

\*namePtr= calloc(100,1);

printf("%5sWhich file would you like to open?\n%10s: ","","");

fgets(\*namePtr, 99, stdin);

ntemp= strchr(\*namePtr, '\n');

\*ntemp = '\0';

size= ntemp- \*namePtr;

if ((new= realloc(\*namePtr, size))!= NULL){

\*namePtr= new;

}

else {

printf("\n\n%5sthere was a problem processing the name of the file.","");

return;

}

if ((\*filePtr=fopen( (char\*) (\*namePtr), "rb+" ))==NULL){

printf("%5sThere was a probelm opening the file.\n\n","");

}

else{

printf("%5sFile successfully accessed.\n\n","");

}

fseek(\*filePtr, 0, SEEK\_END);

f\_size= ftell(\*filePtr);

rewind(\*filePtr);

if ((\*text= calloc(f\_size,1))==NULL){

printf("%5sThere was a problem allocating the initial chunk of memory for a dynamic array of the file contents.\n\n", "");

}

else{

fread(\*text, 1, f\_size, \*filePtr);

}

/\* comment

[^^above^^]

f\_size would be (f\_size/sizeof(char)), but sizeof(char) is 1 so I just

simplified it

-fread() looks like I flipped size\_t and

sizeof(type) from the prototype.

\*/

currArrPos= \*text;

new\_lScout= currArrPos;

if ((\*start=calloc(1, sizeof(struct listNode)))==NULL){

printf("%5sThere was a problem allocating the first struct listNode in the list", "");

}

currNode= \*start;

if (((\*currNode).line= calloc((new\_lScout-currArrPos),1))==NULL){

printf("%5s there was a problem allocating space for a line storage array.","");

}

/\*

After writing all this I noticed that the stdin used for fgets() is a file ptr, which made me wonder

if I could have just read each line from the file straight into the nodes.

I didnt know if it would increment the file pointer to the start of the next line or not,

and I've rewritten this twice already, so I just left it as it was.

but still, would fgets() work for this?

\*/

for (counter=0; counter<f\_size; counter++){

if (\*new\_lScout!='\n' && \*new\_lScout!='\0'){

new\_lScout+=1;

}

else{

if (((\*currNode).line= calloc((new\_lScout-currArrPos),1))==NULL){

printf("%5s there was a problem reallocating space for the terminating character '\0'.","");

}

linePtr= (\*currNode).line;

for(counter\_ception=0; counter\_ception<(new\_lScout-currArrPos); counter\_ception++){

\*(linePtr+ (int)counter\_ception)= \*currArrPos;

currArrPos+=1;

// ^^^^^^^^^^^^^^--->currArrPos+=sizeof(char)

}

if (\*new\_lScout=='\n'){

if ((temp= calloc(1,sizeof(struct listNode)))==NULL){

printf("%5sThere was a problem allocating memory for one of the struct listNodes ofthe list.","");

}

else{

(\*currNode).next= temp;

currNode= temp;

}

}

else {

(\*currNode).next=NULL;

}

new\_lScout+=1;

}

}

}

//--------------------------------------------------------------------------------- instructions()

void instructions(int \*instructionsPtr){

char trash[10];

printf("%5sWhat would you like to do? \n%10s%s\n%10s%s\n%10s%s\n%10s%s\n%10s%s\n%10s%s\n%10s%s\n\n%5s:%3s",

"",

"","(1) open a file for line-by-line editing",

"","(2) Print the current file to the screen (current file before edits) ",

"","(3) print the current list to the screen (current file after edits)",

"","(4) Print list to file (save edits to the file !!!this cannot be undone!!!)",

"","(5) Delete a line",

"","(6) Insert / Prepend / Append lines",

"","(7) end program",

"","");

scanf("%d", instructionsPtr);

fgets(trash, 4, stdin);

if (\*instructionsPtr<1 || \*instructionsPtr> 7){

printf("\n\n%5sINVALID SELECTION (1-7)\n\n","");

}

}

//--------------------------------------------------------------------------------- getLineNum()

void getLineNum(int \*lineNum){

char trash[10];

printf("%5sWhere would you like to insert/delete (Line number) \n%10s%3s Target Line Number\n%5s: ","","###)","");

scanf("%d", lineNum);

fgets(trash, 5, stdin);

return;

}