

COMP 264 Lecture Notes

R. I. Greenberg

1 More on Pointers and Linked Lists

I got the following from asking ChatGPT to “Write a C program to read command line arguments into a linked list”: `argstoll/original.c`

Here is the code with nice indentation:

`argstoll/reindented.c`

The above code does correctly handle freeing the linked list unlike the following naive approach one may have tried in the `freeList` function: `argstoll/freeingerror.c`. We’ll see how we can use GDB on this to see where the segmentation fault occurs with the `bt` command.

Now, going back to the file `argstoll/reindented.c`, let’s just do some variable renaming so that the name `head` will be used in a consistent fashion and use `ptrtohead` where appropriate. Also slightly compacting code by removing curly braces on a while loop body containing just one statement. That gives us

`argstoll/consistenthead.c`.

Next, let’s avoid having any occurrence of `**`, making `InsertEnd` called like other procedures. But if we are not careful, we can have a failure due to the subroutine not returning a modified head:

`argstoll/nodoubleptr.c`

We can fix this by adding logic to `main` so that `InsertEnd` only needs to operate on a non-empty list, and simplifying `InsertEnd`:

`argstoll/fixednodoubleptr.c`

Inserting at the end of the list isn’t very efficient, since you have to march through the whole list each time. Let’s work towards a different approach by eliminating the `InsertEnd` procedure. Here we just do the same thing inline in the main procedure:

`argstoll/inline.c`

Now let’s modify this to do the insertion at the front of the list, which makes the program run in linear time but means the list is reversed:

`argstoll/insertfront.c`

Now let's try to do linear time with insertion at rear, but here there is an error causing a segmentation fault:

argstoll/linearinsertrearwerror.c

Here there is a `printf` to help with debugging, and we can note here how `fflush` helps to get printout even though there is about to be a segmentation fault. Noting that the program crashes only when `argc>2` helps indicate where the error is. You also could use `gdb` to pin down where the crash is occurring and use `gdb` print commands to check values of variables at that point; will demo in class.

I will let you see if you can fix the error. Your upcoming homework also will give you a chance to do some exploration of linked list programming.