Dynamic Memory

10-31-2006

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the reading?
- Do you have any questions about the assignment?
- How to compare strings without []. I got some correct solutions, but none were optimal.

Dynamic Memory and the Heap

- So far we have always talked about the computer giving us memory from the stack when we call new functions. This memory is supposed to be fixed in size, and it stops being valid once we have exited a function.
- Sometimes you need memory that is dynamic and size and which exist across a broader range of functions. For this we have the heap.
- Heap memory sits at the opposite end of what a program has access to from the stack and the two grow toward one another.
- The heap is less organized, but “costs” more to deal with.

The malloc Command

- The most basic memory allocation command in C is malloc. Use the man pages to see the form of this function.
- Notice that it returns a void*. You will need to type cast this to the specific pointer type that you want.
- What you pass into malloc is the size of the chunk of memory that you want. Typically you specify this with the help of the sizeof operator. That makes for more portable code.
- malloc returns just a pointer to that chunk of memory. As with all pointers, it does not know how much memory it points to. You can treat this pointer as an array.

calloc and realloc

- There are two other memory allocation commands in C. I have to admit that I never use them.
- calloc basically does the same thing as malloc, only it initializes the memory to zero while malloc leaves it as garbage. The format for calloc is also different to reflect it is intended for arrays.
- The realloc command will try to “change the size of a memory block”. This can be an expensive command and might actually allocate a new, larger chunk of memory and copy stuff over from the original chunk.

free

- Any time you allocate memory you need to deallocate it later. The reason for this is that unlike the stack, nothing ever “pops” off the heap unless you tell it to. Failure to free memory is called a memory leak and over time it will cause a long running program to consume more and more memory until it runs out and crashes.
- For the free command you simply pass in the pointer to the memory that you are no longer using. It will then be returned to the heap so that later on you could get it back through a call to malloc, calloc, or realloc.

NULL Pointers

- NULL is a pointer value that is always known to be invalid.
- If any of the memory allocation commands can't get the requested memory they will return NULL, just like fopen did.
- I highly recommend that you initialize pointers to be NULL. It is basically impossible to tell if garbage is a valid pointer or not, but NULL you can check against.

Applications of Dynamic Memory

- What are some applications where having dynamic memory could be helpful?
- How should you choose between dynamic memory for arrays and static memory?

Dynamic 2-D Arrays

- How can we make structures like 2-D arrays using pointers?
- What abilities do we have with dynamic 2-D arrays that we didn't have with the static ones?

Playing with Strings

- If I tell you to write a program where you have to print out the length of a string, how will you do it? What about copying a string (like doing an assignment, but with strings)? How would you compare if one string comes before another alphabetically?
- All of these things can be done with fairly simple functions that loop over the arrays. They are so common though that there are library routines to do them and other things.
- Next class we will go into the details of playing with strings both with and without library aid.

Minute Essay

- Assume the user has input a number into int num. Create a triangular 2-D array where the first row has 1 element, the next has 2, and so on until the last has num elements in it.
- Assignment #6 is due on Thursday.