Static Linking and Direct Access Files

9-16-2003

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- How would you write a linked list if you didn't have pointers?

Data Structures Without Pointers

- All a pointer does is refer to another part of memory. The strength of a pointer is that it can refer to any part of memory.
- If you have an array you can also refer to a chunk of memory in the array with an integer index. This way you can build data structures with links that are only in static memory. For this reason it is sometimes called static linking, but it can also be in a vector which isn't static.
- It requires a "context" and can't point to anything.

Free Lists

- To make this work we have to keep track of which elements of the "array" we aren't using so when we need a new one we can get one. This is typically done is a free list. It's a linked list through all free elements.
- The simplest implementation is a stack. If you make it a queue instead of a stack you can have some undo abilities.

Implementation Details

- The array where your elements are is of the "node" type. You probably want to keep this pool of nodes as a static value in the class. The head of the free list would also be a static variable.
- In C++ you could overload the new and delete operators on that type so that calls to new and delete take elements from the free list or return them to it. However, this doesn't give you exactly the behavior you want with static linking because new returns a pointer.

Static Linking and Files

- Why would you use static linking in a language with pointers?
  - Not using dynamic memory can be faster.
  - Even more important, pointers mean nothing outside of that program, integer links can be written to file and read back in.
- If you have a pool of elements with a free list going through it, writing the whole pool to disk can be a very fast way to save.

Binary Files

- In PAD1 you learned about how to read and write text data from/to FILEs in C. You can also write binary data to them. When you do this you write the exact bits in a chunk of memory to disk or read exact bits from disk to memory.
- Doing this for large chunks can be much faster than alternatives and it also typically takes less space than text.

Doing it in C

- If you want to use binary access you should first call fopen with a 'b' in the second string (like "rb+" to open for reading but allow writing).
- fread will read in data from a file to a specified block of memory.
- fwrite will write a block of memory out to the disk.
- Note that all pointers written are useless.

Functions for Direct Access

- What makes binary files truly powerful is that ability to do direct access. That is to say you can jump to any part of the file and read the contents.
- fseek is the command that lets you jump to a given part of the file.
- ftell will tell you where you are in a file at a given time. This can be useful in some situations.

Fixed/Variable Record Length

- If all the records you write to a file are the same length then jumping to a random one is easy. It works just like an array on disk.
- If you have variable length records then you will also need an index that tells you where each record starts. The index basically a binary file with fixed length records. They are "static pointers".

Minute Essay

- Assume you have a pool of 1000 Node elements for a linked list. Show code to write this to a file.
- Remember that assignment #1 is due today. There is an open lab this evening. I won't be staying extremely late because of the Distinguished Scientist that is visiting.
- Quiz #1 is next class.