Files and Malloc

CSSE 221

Fundamentals of Software Development Honors

Rose-Hulman Institute of Technology

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Announcements

- Questions on files or malloc?
- Please pass the last quiz of the term, "Files and Malloc"
  - C Programs through Strings due now
  - Any paper box-and-pointers still out there?

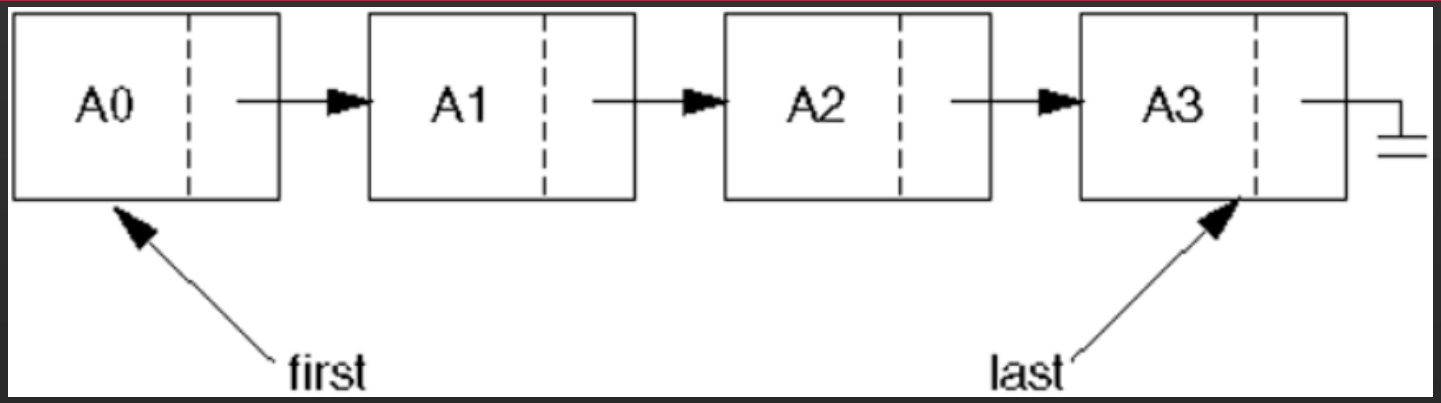
- Simulation grades not yet available
- All C Projects due Friday 5:00 pm
- Final Exam Monday morning,
  - 8 am to 12 pm, O169
  - Start organizing your questions!

Official help

- Login via secureCRT to
  sliderule.csse.rose-hulman.edu
  addiator.rose-hulman.edu
- This gives a shell.
- Can use it to get help about many things in C
  - For example, type "man strncat"

Starting LinkedLists

Linked Lists



- Basic version:
  - The LinkedList struct has only 2 fields: pointers to the first and last nodes.
  - Each node keeps track of the next node in the list.

Keys to success

- Always draw box-and-pointer pictures to help you figure out algorithms for inserting and deleting, etc.
- Be sure to handle special cases: is inserting into an empty list any different from inserting into a non-empty one?
- Handle memory with care:
  - Every time a node is created, we'll malloc a node.
  - So every time we remove the node, we'll free it!

LinkedListBasic.h

```
typedef struct {
    Node * first;
    Node * last;
} LinkedList;
```

```
typedef struct {
    int data;
    Node* next;
} Node;
```

Unfortunately, you can't recursively create a typedef!

This alternate way to declare structs works:

```
struct _Node {
    int data;
    struct _Node* next;
};
```

```
typedef struct _Node
    Node;
```

Lots of functions to write

- makeNode
- makeList
- addFirst, addLast
- display
- getSize
- removeFirst, removeLast
- setAt(pos), insertAt(pos), removeAt(pos)
- deleteList
- displayRecursive
- reverse, reverseRecursive

- Check out the LinkedListBasic project from your repos.
- Let's look together at the header and write some of the code.
- Then you'll work on these a lot in class.

Break

- http://xkcd.com/379/

Of course, the assert doesn't work.

Eliminating Special Cases

- Head and tail nodes:
  - Head: an extra node at the beginning of the linked list implementation that points to the node containing the first List item. The contents of the head node are not part of the List. This is stored in the list instead of "first"
  - Tail: an extra node at the end of the list, for symmetry in doubly-linked lists.
- Thus there are two nodes in the representation of the empty list, three nodes in the representation of a one-element list, etc.

An enhanced version

- Once you are done, you should check out the LinkedListEnhanced project. It includes the following enhancements:
  - Doubly-linked
  - Dummy nodes (head and tail) to remove special cases.
  - Size field to make getSize a constant-time operation.