Lab 7

Stacks
Assignment 3

October 20th, 2010
James Marshall

Previous Lab

- Command Line Arguments
- Recursion and CLA example

Motivation

- Stacks
  - Call Stack (remember our recursion examples)
  - Browser “back” button
  - Matching: ( ) { } [ ]
  - Forth (popular with embedded systems)

Programing Assignment 3

- Easier than the last 2!
- But longer...
- Still linked lists!

Input

- Messages
  - “Note that you are guaranteed that each message in messagesfile.txt is more recent (has later date) than all the messages before it in that file.”
- Queries

Requirements - Queries

- LIST-MESSAGES-BY-DATE
- LIST-MESSAGES-FROM email-address-string
- DELETE-MOST-RECENT-MESSAGE
- DISPLAY k

Queries imply Stack

- Input ordered by date.
- DELETE-MOST-RECENT-MESSAGE
  - pop!
- LIST-MESSAGES-BY-DATE
  - Easy if already ordered

Other queries

- LIST-MESSAGES-FROM email-address-string
  - Doesn't fit with stack, but still do-able
- DISPLAY *k*
  - Again, doesn't scream stack, but not problem

Stack Review

- LIFO
  - Last In – First Out
- Two main operations
  - Push()
  - Pop()
- We will need extra
  - Peek()... or a way to traverse the stack
  - For the “other” queries

Queues (briefly)

- Similar to Stacks
- FIFO
  - First In – First Out
- Take from the “front” - like a stack
- Add to the “back” - unlike a stack

Stack Implementation

- You already know how!
- Linked List! Huzzah!
  - Constrained (simpler)
  - No arbitrary insertions
- Remember, input is already date ordered
  - No sorting (such as insertion sort)

Implementation Cont.

```
struct node {
  // Put data to store here
  struct node *next;
};

// new was allocated previously
void push(node ** top, node * new){
  new->next = *top;
  *top = new;
}
```

Implementation Cont.

```
node* pop(node ** top) {
   node* temp = *top;
   *top = (*top)->next;
   // Don't forget to de-allocate temp
   // elsewhere
   return temp;
}
```

Other Slides

- Mira's
  - Show how to implement a char stack
- Bragg's
  - A practical example on paren matching
- But what do you need in your stack?

Read Message Data

FROM: ayoussef (<= 50 chars)

DATE: 09-15-2005 (MM-DD-YYYY)

SUBJECT: Do your homework (<= 60 chars)

BODY:

My advice to you is to start to do your programming assignment 1 today. -AY

########## (Always 10 '#'s)

I'm sorry.

- Parsing strings is a pain
- But, the input is very well formatted
- Not too bad, just test early

Remember File I/O?

- Use fopen and fclose

    FILE *fp;

  ```
  fp = fopen(“input.txt”, “r”);
  ```

- Then use fprintf and fscanf

  ```
  int lenght;

  fscanf(fp, “%d”, &length);
  ```

- Remember, when “%s” you need to supply a char* that is long enough

Problems

- Can't hard-code filename
  - Command line args... last lab
- fscanf
  - %s : “Matches a sequence of bytes that are not white-space characters.”
    - http://opengroup.org/onlinepubs/007908775/xsh/fscanf.html
  - Should work, stops on spaces and newlines
  - What about MM-DD-YYYY?
  - What about body?

strtok

- Not necessary, but may be easier

```
char* token, input, delimiter;

// Get first token
token = strtok(input, delimiter);

// Get next token
token = strtok(null, delimiter);
```

What To Do Now

- Start writing code.
  - Implement a stack
  - Read file names from command line args
  - Push() and Pop(), test!
  - Try the example input files provided
  - Implement the queries
- But most importantly.....

Start NOW!