15-123:

Effective Programming in C and Unix

With Hunter Pitelka

Carnegie Mellon
School of Computer Science

Exam 2 Review

Recitation 8
Wednesday October 15th, 2008

Carnegie Mellon
School of Computer Science

Overview

- 2 questions, do them.
- Lets go over the solution
- Some quick Q&A.

Dynamic Memory

- Write a C program that will read in a file with a specified number of lines, build an array out of the file (each line will have exactly one integer).
- Print the array backwards.
- No error checking is necessary.
- usage: ./buildArray size filename

1.Dynamic Memory

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
        FILE* fp;
        int input,size,i;
        int *array;

        if(argc != 3){
                printf("Usage: %s size filename\n",argv[0]);
                exit(-1);
        }

        size = (int) strtol(argv[1],NULL,10);
        array = (int *) malloc(sizeof(int)*size);

        fp = fopen(argv[2],"r");
        i = 0;
        while(fscanf(fp,"%d\n",&input) != EOF){
                array[i] = input;
                i++;
        }

        for(i=size-1;i>=0;i--){
                printf("%d\t",array[i]);
        }
        printf("\n");
        free(array);
        array = NULL;
        return 1;
}
```

Dynamic Memory Questions

- How would you change this from reading numbers to strings?
- How would you change this to allow for invalid lines in the file?
- What would you do to check for a valid file?
- How would you test this program?

How to test!

- Write a shell script, duh!

```
#!/bin/bash

if [ $# -ne 2 ]; then
        echo "Usage: $0 filename buildArrayExecutable ";
        exit;
fi


cat $1 | nl | sort -r | cut -f2 | tr "\n" "\t" > bashOutput
echo >> bashOutput

lines=`cat $1 | wc -l`;
`./${2} $lines $1 > Coutput`

diff bashOutput Coutput

rm bashOutput Coutput
```

2. isInLL(linkedlist list, char *word);

```
int isInLL(linkedlist *list, char *word){
        node *current;

        if(NULL == list){
                return ERR_NULL_LIST;
        }
        if(NULL == item){
                return WARN_INVALID_ARGUMENT;
        }
        if(0 == list->count){
                return WARN_ELEMENT_NOT_IN_LIST;
        }
        current= list->head;
        while(strcmp(current->data,word) != 0){
                current = current->next;
                if(NULL == current){
                        /*we have reached the end of the linked list, the
element is not here */
                        return WARN_ELEMENT_NOT_IN_LIST;
                }
        }
        /* if the while loop ends, and we are here, then the item must be
current->data*/
        return SUCCESS;
}
```

Final Words

- STUDY!!1!!one!!!!!11!
- Write lots of test practice programs
- Expect lots of
  - memory
  - pointers
  - function calls
  - I/O processing

kthxbai