15-213 Recitation

2/18/2012

Announcements

- Buflab due tomorrow
- Cachelab out tomorrow
- Any questions?

Outline

- Cachelab preview
- Useful C functions for cachelab

Cachelab

- Part 1: you have to create a cache simulator (not an actual cache!) that will record the hits, misses and evictions of the cache for a given trace
- Part 2: you will be asked to write code to calculate some matrix operations, while efficiently using the cache
- You'll learn more about caches this week in lecture

Cachelab(2)

- You will not have any starter code for this lab
  - This is important because you will need to be able to create programs from scratch at some point in the future
- Some necessary items for doing the lab include :
  - Parsing command line arguments using getopt()
  - Using “make”
  - Opening and reading from a file

#include

- Remember to include files that we will be using functions from
- For cache lab you will need at least the following

```
#include <getopt.h>
#include <stdlib.h>
#include <unistd.h>
#include "cachelab-tools.h"
```

getopt

- getopt() automates parsing elements on the unix command line
  - Typically called in a loop to retrieve arguments
  - Its return value is stored in a local variable
  - When getopt() returns -1, there are no more options

To use getopt, your program must include the header file unistd.h

getopt(2)

- A switch statement is used on the local variable holding the return value from getopt()
  - Each command line input case can be taken care of separately
  - “optarg” is an important variable – it will point to the value of the option argument
    - You will have to use the atoi() function to convert the inputs in char* format to ints
- Think about how to handle invalid inputs

Example

```
int main(int argc, char** argv){
int opt, x;
while(-1 != (opt = getopt(argc,argv,“x:") ) ){ //looping over arguments
    switch(opt) { //determine which argument it’s processing
    case 'x':
        x=atoi(optarg);
        break;
    default:
        printf(“wrong argument\n");
    }
}
```

Suppose the program executable was called “foo”. Then we would call “./foo -x 1 “ to pass the value 1 to variable x.

fopen

- The fopen() function opens and i/o stream to a file and returns a pointer to that stream
  - Takes 2 parameters: filename and open type (r, w, etc.)
- In cachelab you will use this to open trace files
- Remember to use fclose() on any files that you open

fscanf

- The fscanf() function is just like scanf() except it can specify a stream to read from (scanf always reads from stdin)
  - parameters: file pointer, format string with information on how to read file, and the rest are pointers to variables to storing data from file
  - Typically want to use this function in a loop until it hits the end of file
- fscanf will be useful in reading from the trace files

Example

```
FILE * pFile; //pointer to FILE object
pFile = fopen ("myfile.txt",“r"); //open file for
                                  //reading
int x, y;
char c;
//read two ints and a char from file
fscanf(pFile, “%d %d %c”, &x, &y, &c);
fclose(pFile); //remember to close file when done
```

Malloc/free

- Use malloc this to allocate memory on the heap
  - Return value of malloc is void*, so cast to desired type
  - Ex: int *p = (int *)malloc(sizeof(int));
- Always free what you malloc, otherwise may get memory leak
  - Ex: free(p);
- Don’t free memory you didn’t allocate

Failure Conditions and Error Checking

- Don’t assume correct inputs and outputs
- Look at and handle the bad cases as well
  - i.e. malloc may return NULL, user may enter wrong number of inputs into program, inputs themselves may not be valid, etc.

Makefile

- Makefiles are handy files that spare you from the burden of recompiling many source files on the command line whenever you want to update your program
  - It is always named Makefile
  - Invoke it using “make”
  - To remove object files and executables, call “make clean”

Reminder on Coding Style

- Remove dead code
  - Ex: printf statements for debugging, unused variables, etc.
- Use descriptive variable and function names

Tutorials

- getopt: http://www.gnu.org/software/libc/manual/html_node/Getopt.html
- fscanf: http://crasseux.com/books/ctutorial/fscanf.html
- Google for more