C Review and Refresher

Andrew Fitz Gibbon

February 26, 2010

Typical Program Layout

- Libraries and other includes
- Constants, defines, and other globals
- Function prototypes
- int main()
- Function implementations

Common Libraries

- stdio.h
  - provides things like printf() or puts()
- stdlib.h
  - provides things like rand() or malloc()
- math.h
  - provides things like sqrt() or log()
- mpi.h
  - provides things like MPI_Send()

Canonical Hello World

```
#include <stdio.h>

int main () {
        puts("Hello World!");
}
```

Compiling and Running

```
$ gcc -o hello_world hello_world.c
$ ./hello_world
  Hello World!
$
```

Variables

- Floating point
  - float, double, long double
- Integer
  - short, int, [unsigned] long int, [unsigned] long long int
- Strings and Characters
  - char, char*

Arrays

- Single Dimensional

```
int array[ELEMENTS]
float array[ELEMENTS]
```

- Multi-Dimensional

```
int array[ROWS][COLUMNS]
float array[ROWS][COLUMNS]
```

Conditionals and Loops

- Conditionals

```
if (x == 0) {
  puts("Don't divide by x!");
} else if (x == 1) {
  puts("Dividing by x won't get you far!");
} else {
  puts("No claims on what x will get you.");
}
```

- Loops

```
for (int i = 0; i < 100; ++i) {
        puts("This message will appear 100 times");
}

int i = 0;
while (i < 100) {
        puts("This message will appear 100 times");
        ++i;
}
```

Argument Processing

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

int main (int argc, char ** argv) {
  printf("Program name: %s\n", argv[0]);

  printf("First argument: %s\n", argv[1]);

  printf("First argument as int: %d\n",
    (int)strtol(argv[1], (char **)NULL, 10));

  printf("First argument as float: %f\n",
    strtof(argv[1], (char **)NULL));
}
```

Pointers and Memory Allocation

- Pointers

```
#include <stdlib.h>
...
// Provides:
malloc(...); // see malloc(3) for more
free(...);
```

- Scalars

```
int * i = (int*)malloc(sizeof(int));
float * f = (float*)malloc(sizeof(float));
```

Pointers and Memory allocation (cont...)

- Arrays

```
int * a = (int*)malloc(sizeof(int)*ELEMENTS);
int ** a2 = (int**)malloc(sizeof(*int)*ROWS);
for (int i = 0; i < ROWS; i++)
  a2[i] = (int*)malloc(sizeof(int)*COLUMNS);
```

Functions

```
// Simple factorial example
// demonstrating recursive functions
#include <stdio.h>

double factorial(int);

int main() {
        printf("10! = %.0lf\n", factorial(10));
}

double factorial(int n) {
        return(n > 1 ? n * factorial(n-1) : n);
}
```

Custom Header Files

- Header file (fact.h)

```
double factorial(int n) {
        return (n > 1 ? n * factorial(n-1) : n);
}
```

- Source file (fact.c)

```
#include <stdio.h>
#include "fact.h"

int main() {
        printf("10! = %.0lf\n", factorial(10));
}
```

Makefiles

- Makefile Contents

```
CC=/usr/bin/gcc

#target: dependencies
#        directives (i.e. commands)

default: all

all: hello_world factorial

hello_world: hello_world.c
        $(CC) -o hello_world hello_world.c

factorial: fact.h fact.c
        $(CC) -o fact fact.c
```

- Running make

```
$ make [target]
```

Additional Resources

- http://www.google.com
- http://en.wikipedia.org/wiki/C_syntax
- http://en.wikipedia.org/wiki/Stdlib.h
- http://en.wikipedia.org/wiki/Math.h
- http://www.gnu.org/software/make/manual/html_node/Introduction.html#Introduction