# CS429
*Computer Architecture*

# Introduction to C
# September 5, 2012

## Topics

- Simple C program
  - Basic structure, functions, separate files
- Compilation
  - Phases, options
- Assembler
  - GNU style, byte ordering, code and data segments
- Tools for inspecting binary
  - Programs:  od, objdump

## A Simple C Program

- A first program is to just print a short message.
- We assume our target is a 32-bit, X86-compatible machine.
- This program prints “Hello!” to its “standard output”.
- We will use “gcc” to compile this program.

```
/* Simple Program */

#include "stdio.h"

int main ( )
{
  printf( "Hello!\n" );
}
```

## Simple C Program with Return Status

- This program returns a status code using exit( n );
- We compile with “gcc -O2 -o <objectFile> <sourceFile.c>”
- Be wary of the PowerPoint fonts.

```
/* Simple Program */

#include “stdio.h”       //  For the printf command
#include “stdlib.h”      //  For the exit command

int main ( )
{
  printf( "Hello!\n" );
  exit( 11 );
}
```

## Program with Environment Variables

- **This program has environment input arguments**
- **Variables argc and argv reflect the command line.**
- **Variable env reflects the environment variables.**

```
/* Simple Program */

#include “stdio.h”        //  For the printf command
#include “stdlib.h”       //  For the exit command

int main ( int argc,  char *argv[],  char *env[] )
{
  printf("Status: number of command-line args.\n" );
  exit( argc );
}
```

## The Command Line Arguments

```
#include "stdio.h"
#include "stdlib.h"

int main (int argc, char *argv[], char *env[] )  {
  int i;
  if( argc == 1 )
    printf( "The command line argument is:\n" );
  else
    printf( "The %d command line arguments are:\n", argc );

  for( i = 0; i < argc; i++ )
      printf( "Arg %3d:  %s\n", i, argv[ i ] );
  exit( argc );
}
```

## The Command Line Arguments

```
#include "stdio.h"
#include "stdlib.h”
int main (int argc, char *argv[], char *env[] )  {
  int i;
  printf( "The environment strings are:\n" );

  i = 0;
  while( env[i] != NULL )
    {
      printf( "Arg %3d:  %s\n", i, env[ i ] );
      i++;
    }
  exit( i );
}
```

# The GNU GCC Compiler

## GCC is a cross compiler

- It runs on many machines
- Input languages:  C, C++, Fortran, Java, and others
- Many target languages:  X86, PowerPC, ARM, MC680x0, …

## Documentation available on-line

## GCC works in phases:

- gcc -v -O2 -o <object-file> <source-file>.c

## GCC can be used to print assembler

- gcc -S -O2 <source-file>.c

## Assembler Output From gcc

- Produces assembler output, doesn't run "gas" assembler

```
gcc -S -O2 -c sum.c   ==>
```

**sum.c**

```
int sum( int x, int y )
{
  int t = x + y;
  return t;
}
```

**sum.s**

```
                .file       "sum.c"
                .text
                .p2align 4,,15
.globl sum
                .type     sum, @function
sum:
                pushl    %ebp
                movl     %esp, %ebp
                movl     12(%ebp), %eax
                addl      8(%ebp), %eax
                popl      %ebp
                ret
```

## Assembler Output From binary

- "objdump" can be used to view the binary.

```
sum.o:     file format elf32-i386

Disassembly of section .text:

00000000 <sum>:
   0:   55                push   %ebp
   1:   89 e5             mov    %esp,%ebp
   3:   8b 45 0c          mov    0xc(%ebp),%eax
   6:   03 45 08          add    0x8(%ebp),%eax
   9:   5d                pop    %ebp
   a:   c3                ret
```

## Show Bytes Program

```
#include <stdio.h>
typedef unsigned char *byte_pointer;

void show_bytes( byte_pointer start, int len ) {
  int i;
  for( i =0; i < len ; i++ )
    printf(" %.2x", start[i] );
  printf("\n");
}

int main(int argc, char *argv[], char *env[] ) {
  int    i = 15213;
  float  f = 15213.0;
  double d = 15213.0;
  int   *p = &i;

  show_bytes( (byte_pointer) &i, sizeof(i) );
  show_bytes( (byte_pointer) &f, sizeof(f) );
  show_bytes( (byte_pointer) &f, sizeof(d) );
  show_bytes( (byte_pointer) &p, sizeof(p) );
}
```

## C Tutorials Available on the Web

Use a search engine, and type

- "C tutorial" and you will get a lot of options.
- I thought this next reference was good.
  - http://www.iu.hio.no/~mark/CTutorial/CTutorial.html
- A local reference by Christian Miller (UTCS grad student)
  - http://www.cs.utexas.edu/users/hunt/class/2012-fall/cs429/lectures/Miller-C-Intro.pdf

"The C Programming Language"

- Brian Kernighan and Dennis Ritchie is a standard guide.
- Prentice-Hall Publisher, I suggest the 2nd Edition

A modern architecture is not an ISA alone

- All developments target a system
  - Hardware and software combined
  - Benchmarks are widely used to illustrate performance
  - Be careful -- your experience may be different