CSE 30: Computer Organization and Systems Programming

Lecture 5:
Memory organization (contd)
C run time environment

Diba Mirza
University of California, San Diego
Announcements

• HW1/PA1 is now available on TED, due next Monday (10/12) at 11:30pm
Units of Memory

1KB
1MB
1GB
1TB
1PB
1EX
Memory Organization

• If N bits are used to represent memory addresses in a computer, we say it has an N-bit address space.

• ARM is a 32-bit architecture:
  1. 32 bit address space
  2. 32 bit registers
  3. 32 bit instructions
  4. 32 bit (4 byte) integers
Memory

PI Q: How much memory can be supported on a 32-bit machine?
A. 2GB
B. 4GB
C. 8GB
D. 16GB

How about a 64-bit machine?
The C runtime environment
Steps in program translation

Code Time

Program in C

Helloworld.c

Compile Time

Compiler

Executable:
Program in machine code
+Data in binary

1000110001100010000000000000000000
1000110011110010000000000000100
1010110011110010000000000000000
1010110001100010000000000000100
10101100011000100000000000000100

Run Time

Hardware
What does gcc do?

$ gcc hello.c

“Source”
Program in C

```
#include <stdio.h>
void func1(int a, char *b)
{
    if(a > 0)
    {
        *b = ‘a’ ;
    }
}
int main()
{
    ....
    func1();
    printf("\abc");
}
```

“Executable”:
Equivalent program in machine language

```
0000 1001 1100 0110
1010 1111 0101 1000
1010 1111 0101 1000
0000 1001 1100 0110
1100 0110 1010 1111
0101 1000 0000 1001
0101 1000 0000 1001
1100 0110 1010 1111
```
What does gcc do?

$ gcc hello.c$

$ ./a.out  (executable loaded in memory and processed) Also referred to as “running” the C program

“Source”: Program in C

“Executable”: Equivalent program in machine language
Steps in gcc

• The translation is actually done in a number of steps

hello.c -> Compiler (cpp) -> hello.o

 hello.s -> Assembler (as) -> hello.o

 Compiler (cpp) -> Linker (ld) -> a.out

gcc
Include code written by others

- Code written by others (libraries) can be included
- `ld` (linkage editor) merges one or more object files with the relevant libraries to produce a single executable

![Diagram of the code compilation process](image)

- `hello.c`
- `cpp`
- `cc1`
- `cc1`
- `hello.o`
- `ld`
- `hello.s`
- `as`
- `gcc`
- `a.out`
- Library files e.g. `math.o`: the math library
Steps in gcc

- Ask compiler to show temporary files:
  
  $ gcc –S hello.c
  $ gcc –c hello.c
  $ gcc –o prog_hello hello.c
void foo (int, int); /* This is the function declaration*/

void foo(int a, int b) {
    /* This is the function definition */
}

C functions

foofile.c
#include <stdio.h>

void main( ) {
    int i = 15;
    printf(“ Hello World %d \n”, i);
}

$gcc –o hello hello.c
How is ‘other’ code included?

- Include Header files (.h) that contain function declarations - the function interface
- The corresponding .c files contain the actual code

file1.h

```c
void func1(int, char *);
int func2(char *, char *);
```

file1.c

```c
#include <stdio.h>

void func1(int a, char *b)
{
    if(a > 0)
        { *b = ‘a’ ; }
}
int main()
{
    ...... 
    func1(i, pj);
    printf(“\abc”);
} 
```
What happens when we compile hello.c as follows?

$ gcc -o hello hello.c

A. An executable called “a.out” is generated
B. An executable called “hello” is generated
C. Compiler error
D. “Hello world” is printed to standard output
void f1(int, char *);
int f2(char *, char *);

#include <stdio.h>
void func1(int a, char *b)
{
    if(a > 0)
    {
        *b = 'a';
    }
}
int main()
{
    func1();
    printf("abc");
}