CS 211 RECITATIONS WEEK 5

Wenjie Qiu
Teaching Assitant
Office Hour: Thu. noon – 1pm
wenjie.qiu@rutgers.edu

Content

- Common C Bugs (see Jeff's slides)
- Arrays vs. Pointers
- Pipes and Redirection
- Shifts and Masks
- Data Sizes
- Endianness

Array vs. Pointers

- char *s1 = “hello”;
- char s2[] = “hello”;

- s1 is a string literal
  - *Compiler will create a string in your binary*
  - *However, it will be stored at a read-only location*
  - *Modify, like s1[0] = ‘H’ is prohibited*
  - *Create if you do not want to modify it, just print/use it*
- s2 is a character array
  - *Store on the stack, will be cleaned after function return*
  - *Can be changed (of course!)*

Pipes & Redirections

- We can redirect the output from a program to a file
  - *./program > output.txt (clean up the output.txt and then redirect)*
  - *./program >> output.txt (append the output)*
  - *Very useful when you want to record multiple runs*
- Special file descriptor : stdin (0) & stdout (1) & stderr (2)
  - *FILE *fptr = open('…'); // fptr is the file descriptor*
  - *find . –name "pattern" 1>out.txt 2>err.txt*
  - *send the output to out.txt, send the error to err.txt*
  - *What if we want to combine?*
  - *find . –name "pattern" 2>&1 > combine.txt*
- We want ignore the error messages
  - *make 2>/dev/null*
  - *make 2>&1 > /dev/null (totally silent)*

Pipes & Redirections

- Pipes
  - *Redirect program1's output to program2's input*
  - *./program1 | ./program2*

- Execute multiple targets
  - *xargs*
  - *ls | grep "pattern" | xargs wc* *(select files match the pattern and then do a word count)*
  - *ls | grep "pattern" | xargs cat* *(select files match the pattern and then print their contents)*

Shifts and Masks

- `x & m` - do a bitwise AND operation
- `x | m` - do a bitwise OR operation
- `x ^ m` - do a bitwise XOR operation
- `~x` - do a bitwise NOT operation

- `x << n` - shift x left by n bits
- `x >> n` - shift x right by n bits

Note:

- A left shift = multiplying by 2
- A right shift = dividing by 2 (and discarding remainders)

- Shifting is a good when doing multiplication/division, good for performance.
- But modern compilers might smart enough to generate such code with appropriate optimization levels. Not necessarily to write such code.
- Be aware of integer overflow and underflow, since shifting will not do any checking!

Data Sizes

C type	Min size	Typical size
char	1	1
short int	2	2
int	2	4
long int	4	8
pointer		8
float		4
double		8

- Typical is typical, but we need to be careful.
- Your int might have more bytes than others!

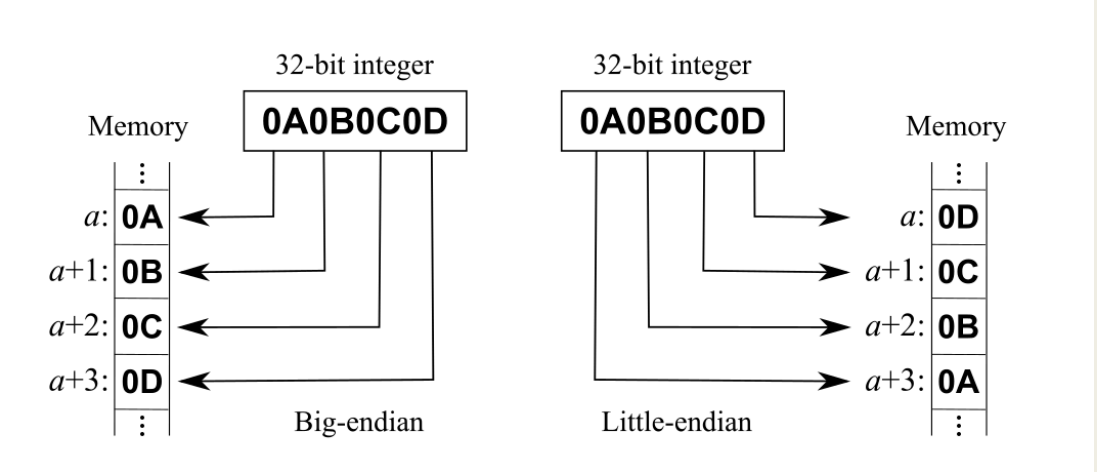
- <stdint.h>
- int8_t
- int16_t
- int32_t
- uint8_t
- uint16_t
- uint32_t
- …

Endianness

- Little Endian
  - *Least significant*

  Byte (LSB) first
- Big Endian
  - *Most significant*

  Byte (MSB) first
- How to check?
  - *Ask your Operating System*
  - *Check using C code*




```
int n = 1;
// little endian if true
if(*(char *)&n == 1) {...}
```