

# CS 3330: C

25 August 2016

1

## Layers of Abstraction

`x += y`

`add %rbx, %rax`

60 03

"Higher-level" language: C

Assembly: X86-64

Machine code: Y86

(we'll talk later)

Logic and Registers

2

## Compilation Steps

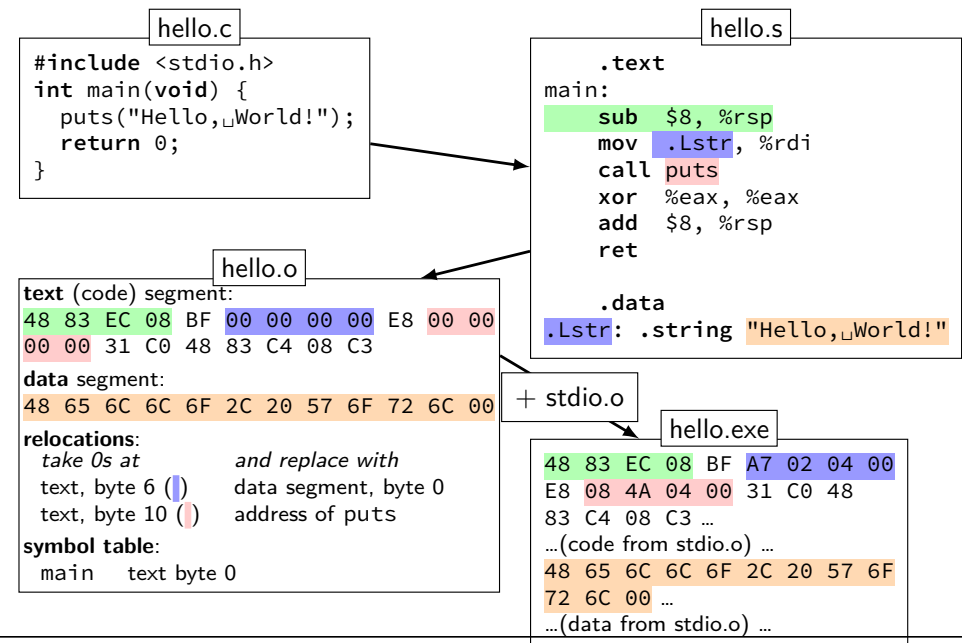
**compile:** `gcc -S file.c`  $\Rightarrow$  `file.s`  
**assemble:** `gcc -c file.s`  $\Rightarrow$  `file.o`  
**link:** `gcc -o file file.o`  $\Rightarrow$  `file (exec.)`

`c+a:` `gcc -c file.c`  $\Rightarrow$  `file.o`  
`c+a+l:` `gcc -o file file.c`  $\Rightarrow$  `file (exec.)`

...

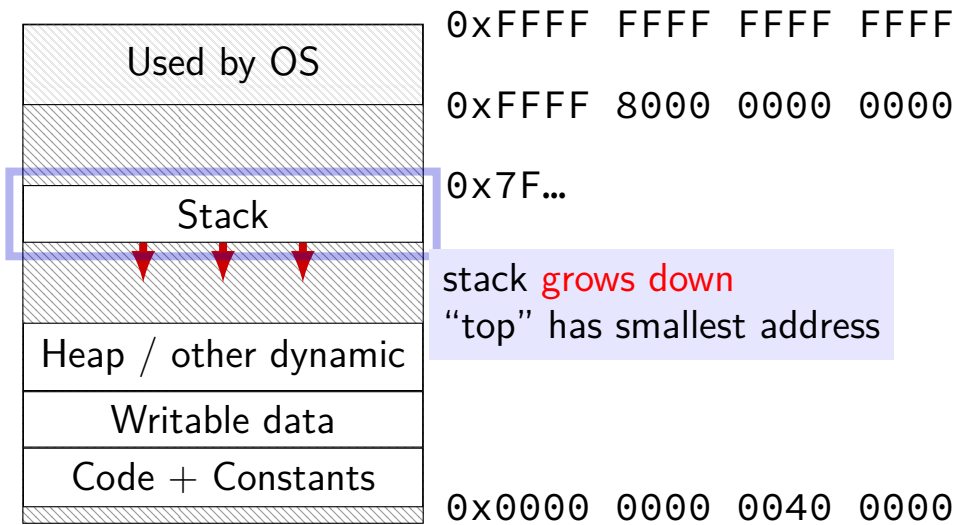
3

## What's in those files?



4

## Program Memory (x86-64 Linux)



5

## C Data Types

Varies between machines(!). For **this course**:

type	size (bytes)
char	1
short	2
int	4
long	8
float	4
double	8
void *	8
<i>anything</i> *	8

6

## Truth

`bool`

`x == 4` is an **int**  
1 if true; 0 if false

7

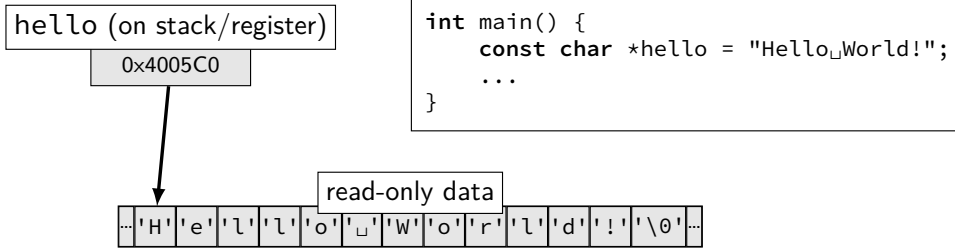
## False values in C

0

including null pointers — 0 cast to a pointer

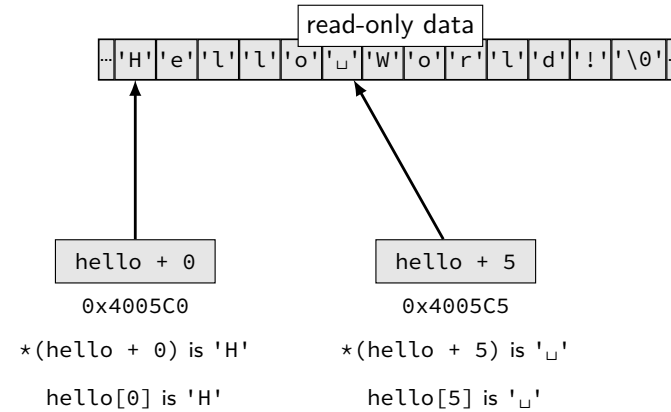
8

# Strings in C



9

# Pointer Arithmetic



10

# Arrays and Pointers

\*(foo + bar) **exactly the same** as foo[bar]

arrays **'decay'** into pointers

11

# Exercise

```
1 char foo[4] = "foo";  
2    // {'f', 'o', 'o', '\0'}  
3 char *pointer;  
4 pointer = foo;  
5 *pointer = 'b';  
6 pointer = pointer + 2;  
7 pointer[0] = 'z';  
8 *(foo + 1) = 'a';
```

Final value of foo?

A. "fao"

B. "zao"

C. "baz"

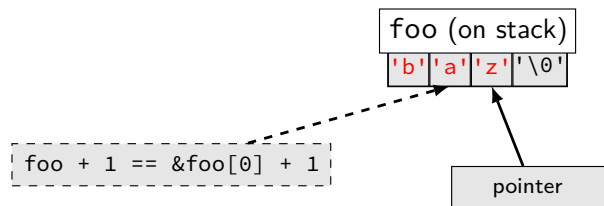
D. "bao"

E. something else/crash

12

## Exercise explanation

```
1 char foo[4] = "foo";
2    // {'f', 'o', 'o', '\0'}
3 char *pointer;
4 pointer = foo;
5 *pointer = 'b';
6 pointer = pointer + 2;
7 pointer[0] = 'z';    better style: *pointer = 'z';
8 *(foo + 1) = 'a';    better style: foo[1] = 'a';
```



13

## Arrays of non-bytes

array[2] and \*(array + 2) still the same

```
1 int numbers[4] = {10, 11, 12, 13};
2 int *pointer;
3 pointer = numbers;
4 *pointer = 20;    // numbers[0] = 20;
5 pointer = pointer + 2;
6 /* adds 8 (2 ints) to address */
7 *pointer = 30;    // numbers[2] = 30;
8 // numbers is {20, 11, 30, 13}
```

14

## Arrays: not quite pointers (1)

```
int array[100];
int *pointer;
```

Legal: `pointer = array;`  
same as `pointer = &(array[0]);`

Illegal: ~~`array = pointer;`~~

15

## Arrays: not quite pointers (2)

```
int array[100];
int *pointer = array;
```

`sizeof(array) == 400`  
size of all elements

`sizeof(pointer) == 8`  
size of address

`sizeof(&array[0]) == ???`  
(`&array[0]` same as `&(array[0])`)

16

# Interlude: Command Line Tips

```
cr4bd@reiss-lenovo:~$ man man
```

17

## man man

```
File Edit View Search Terminal Help
MAN(1) Manual pager utils MAN(1)

NAME
  man - an interface to the on-line reference manuals

SYNOPSIS
  man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-t] [-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...
  man -k [apropos options] regexp ...
  man -K [-w|-W] [-S list] [-t] [-I] [--regex] [section] term ...
  man -f [whatIs options] page ...
  man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
  man -w|-W [-C file] [-d] [-D] page ...
  man -c [-C file] [-d] [-D] page ...
  man [-?V]

DESCRIPTION
  man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and to show only the first page found, even if page exists in several sections.

Manual page man(1) line 1 (press h for help or q to quit)
```

18

## man man

```
File Edit View Search Terminal Help
EXAMPLES
  man ls
  Display the manual page for the item (program) ls.

  man -a intro
  Display, in succession, all of the available intro manual pages contained within the manual. It is possible to quit between successive displays or skip any of them.

  man -t alias | lpr -Pps
  Format the manual page referenced by 'alias', usually a shell manual page, into the default troff or groff format and pipe it to the printer named ps. The default output for groff is usually PostScript. man --help should advise as to which processor is bound to the -t option.

  man -L -Tdvi ./foo.1x.gz > ./foo.1x.dvi
  This command will decompress and format the nroff source manual page ./foo.1x.gz into a device independent (dvi) file. The redirection is necessary as the -T flag causes output to be directed to stdout with no pager. The output could be viewed with a program such as xdvi or further processed into PostScript using a program such as dvips.

  man -k printf
  Search the short descriptions and manual page names for the keyword printf as regular expression. Print out any matches. Equivalent to apropos printf.

  man -f smail
  Lookup the manual pages referenced by smail and print out the short descriptions of any found. Equivalent to whatis smail.

Manual page man(1) line 68 (press h for help or q to quit)
```

19

## man chmod

```
File Edit View Search Terminal Help
CHMOD(1) User Commands CHMOD(1)

NAME
  chmod - change file mode bits

SYNOPSIS
  chmod [OPTION]... MODE[,MODE]... FILE...
  chmod [OPTION]... OCTAL-MODE FILE...
  chmod [OPTION]... --reference=RFILE FILE...

DESCRIPTION
  This manual page documents the GNU version of chmod. chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

  The format of a symbolic mode is [ugoa...][[-+=[perms...]]...], where perms is either zero or more letters from the set rwxXst, or a single letter from the set ugo. Multiple symbolic modes can be given, separated by commas.

  A combination of the letters ugoa controls which users' access to the file will be changed: the user who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users (a). If none of these are given, the effect is as if (a) were given, but bits that are set in the umask are not affected.

  The operator + causes the selected file mode bits to be added to the existing file mode bits of each file; - causes them to be removed; and = causes them to be added and causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.

  The letters rwxXst select file mode bits for the affected users: read (r), write (w),

Manual page chmod(1) line 1/125 27% (press h for help or q to quit)
```

20

## chmod

```
chmod --recursive og-r /home/USER
```

user (yourself) / group / others  
- remove / + add  
read / write / execute or search

21

## tar

the standard Linux/Unix file archive utility

Table of contents: `tar tf filename.tar`

eXtract: `tar xvf filename.tar`

Create: `tar cvf filename.tar directory`

(v: verbose; f: file — default is tape)

22

## Tab completion and history

23

## Back To C

24

# stdio.h

C does not have <iostream>

Instead <stdio.h>

25

# stdio

```
cr4bd@power1
: /if22/cr4bd ; man stdio
```

```
...
STDIO(3)                                Linux Programmer's Manual          STDIO(3)

NAME
    stdio - standard input/output library functions

SYNOPSIS
    #include <stdio.h>

    FILE *stdin;
    FILE *stdout;
    FILE *stderr;

DESCRIPTION
    The standard I/O library provides a simple and efficient
    buffered stream I/O interface. Input and output is mapped into
    logical data streams and the physical I/O characteristics are
    concealed. The functions and macros are listed below; more
    information is available from the individual man pages.
```

26

# stdio

```
STDIO(3)                                Linux Programmer's Manual          STDIO(3)
```

```
NAME
    stdio - standard input/output library functions
```

...

## List of functions

Function	Description
----------	-------------

clearerr	check and reset stream status
fclose	close a stream

...

printf	formatted output conversion
--------	-----------------------------

...

27

# printf

```
int custNo = 1000;
const char *name = "Jane_Smith"
printf("Customer_#%d:_%s\n",
      custNo, name);
// "Customer #1000: Jane Smith"
// same as:
cout << "Customer_#" << custNo
     << ":%" << name << endl;
```

format string must **match types** of argument

28

## printf formats quick reference

Specifier	Argument Type	Example(s)
%s	char *	Hello, World!
%p	any pointer	0x4005d4
%d	int/short/char	42
%u	unsigned int/short/char	42
%x	unsigned int/short/char	2a
%ld	long	42
%f	double/float	42.000000 0.000000
%e	double/float	4.200000e+01 4.200000e-19
%g	double/float	42, 4.2e-19
%%	(no argument)	%

29

## goto

```
for (...) {  
    for (...) {  
        if (thingAt(i, j) == 0) {  
            goto found;  
        }  
    }  
}  
printf("not found!\n");  
return;  
found:  
printf("found!\n");
```

assembly:  
jmp found

assembly:  
found:

30