CMSC 246 Systems Programming

Bryn Mawr College
Instructor: Geoffrey Towell

Go to class web page…

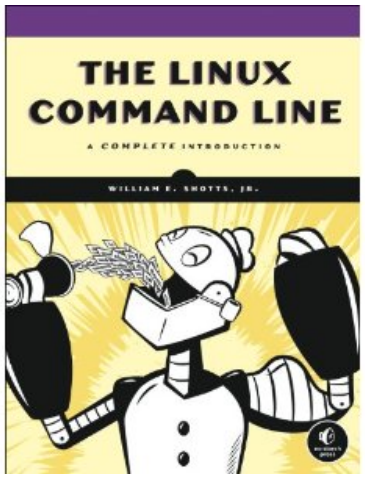
https://cs.brynmawr.edu/Courses/cs246/fall2019/

First Things

- CS account
  - Make sure you can log in
- Lab: Park 231/M 2:25pm-3:45pm
- Lab attendance is required. Lab exercise should be completed BEFORE you start your assignments
- Software: the unix command line!!! (and gcc)

Goals

- Learn Linux (CLI, not WIMP!)
- Learn C
- Learn Linux tools



THE LINUX COMMAND LINE

A COMPLETE INTRODUCTION

WILLIAM E. SHOTTS, JR.

THE C PROGRAMMING LANGUAGE

Evolution of C

Algol60

Designed by an international committee, 1960

CPL

Combined Programming Language
Cambridge & Univ. of London, 1963
Was an attempt to bring Algol down
To earth and retail contact with the
Realities of an actual computer.
Features:

- Big
- Too many features
- Hard to learn
- Intended for numerical as well as non-numerical applications

BCPL

Basic CPL
Designed by Martin Richards, Cambridge 1967
Intended as a tool for writing compilers.
Designed to allow for separate compilation.
Features:

- Typeless language (only binary words)
- Introduced static variables
- Compact code
- Provodes access to address of data objects
- Stream-based I/O

B

Designed by Ken Thompson, Bell Labs 1970
A true forerunner of C
Features:

- Typeless (with floating pt. capabilities
- Designed for separate compilation
- Easily implementable
- Pre-processor facility
- Expensive library

Evolution of C

Algol60

Designed by an international committee, 1960

CPL

Combined Programming Language
Cambridge & Univ. of London, 1963
Was an attempt to bring Algol down
To earth and retail contact with the
Realities of an actual computer.
Features:

- Big
- Too many features
- Hard to learn
- Intended for numerical as well as non-numerical applications

BCPL

Basic CPL
Designed by Martin Richards, Cambridge 1967
Intended as a tool for writing compilers.
Designed to allow for separate compilation.
Features:

- Typeless language (only binary words)
- Introduced static variables
- Compact code
- Provodes access to address of data objects
- Stream-based I/O

B

Designed by Ken Thompson, Bell Labs 1970
A true forerunner of C
Features:

- Typeless (with floating pt. capabilities
- Designed for separate compilation
- Easily implementable
- Pre-processor facility
- Expensive library

C

1971-72
Developed at Bell Laboratories by
Ken Thompson, Dennis Ritchie, and others.
C is a by-product of UNIX.
Ritchie began to develop an extended version of B.
He called his language NB ("New B") at first.
As the language began to diverge more from B,
he changed its name to C.
The language was stable enough by 1973 that
UNIX could be rewritten in C.

K&R C

Described in Kernighan and Ritchie,
The C Programming Language (1978)
De facto standard
Features:

- Standard I/O Library
- long int data type
- Unsigned int data type
- Compound assignment operators

C89/C90

ANSI standard X3.159-1989
Completed in 1988
Formally approved in December 1989
International standard ISO/IEC 9899:1990
A superset of K&R C
Heavily influenced by C++, 1979-83

- Function prototypes
- void pointers
- Modified syntax for parameter declarations
- Remained backwards compatible with K&R C

C99

International standard ISO/IEC 9899:1999
Incorporates changes from Amendment 1 (1995)
Features:

- Inline functions
- New data types (long long int, complex, etc.)
- Variable length arrays
- Support for IEEE 754 floating point
- Single line comments using //

Onwards to C11…

Languages and Cars

C is a racing car that goes incredibly fast but breaks down every fifty miles.
Java is a family station wagon. It's easy to drive, it's not too fast, and you can't hurt yourself.
Perl is supposed to be a pretty cool car, but the driver's manual is incomprehensible. Also, even if you can figure out how to drive a Perl car, you won't be able to drive anyone else's.
Python is a great beginner's car; you can drive it without a license. Unless you want to drive really fast or on really treacherous terrain, you may never need another car.
Lisp: At first it doesn't seem to be a car at all, but now and then you spot a few people driving it around. After a point you decide to learn more about it and you realize it's actually a car that can make more cars. You tell your friends, but they all laugh and say these cars look way too weird. You still keep one in your garage, hoping one day they will take over the streets.

Properties of C

- Low-level
- Small
- Permissive
- Fast

Strengths of C

- Efficiency
- Portability
- Power
- Flexibility
- Standard library
- Integration with UNIX

Weaknesses of C

- Programs can be error-prone.
- Programs can be difficult to understand.
    - International Obfuscated C Code Contest

```
#include<stdio.h>
int a = 256;int main(){for(char b[a+a+a],
*c=b ,*d=b+ a ,*e=b+a+a,*f,*g=fgets(e,(b[
a]=b [a+a] =a- a,a) , stdin);c[0]=a-a,f=c
,c=d ,d=e ,e=f, f= g,g =0,g = fgets(e,a+a
-a+ a -a+a -a+ a- +a,stdin ),f +a-a ; pu\
tchar(+10)) { for( int h= 1,i=1,j, k=0 ,l
=e[0]==32,m,n=0,o=c [ 0]== 32, p, q=0;d[q
];j=k,k=l,m=n,n=o,p=(j)+(k* 2 )+(l =(i =
e[ q]&&i ) &&e[q +1 ]== 32,l*4)+(m* 8 )+(
16*  n  )+(  o  =(h =c[ q]&&h)&&c[q+1]==
32,o* (16+16) )+0-0 +0, putchar(" ......"
/*\  (  |||  )  |/|/ / */".')|)\\\\\\\\'"
"" "|||"   "|||" "|'" ")|)\\\\\\\\'/|/(/"
"(/'/|/\\|\\|'/|/(/(/'/|/\\|\\|"[d[q++]==
32?p:0]));}}/* typographic tributaries */
```

- Programs can be difficult to modify.

Effective Use of C

- Learn how to avoid pitfalls.
- Use software tools to make programs more reliable.
- Take advantage of existing code libraries.
- Adopt a sensible set of coding conventions.
- Avoid “tricks” and overly complex code.
- Stick to the standard.
- Try and adapt the good habits from programming in Java!

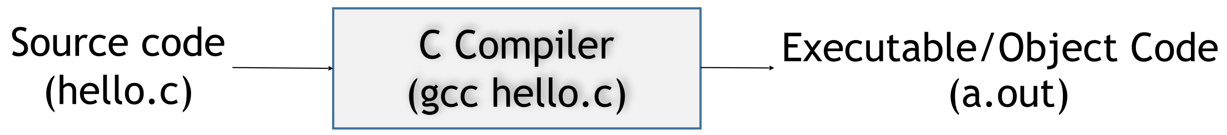
First C Program: Hello, World!

```
/**************
 * Name: Xena W. Princess
 * Purpose: My first C Program, prints: Hello, World!
 * Written on August 5, 2019
 **************/

#include <stdio.h>

int main(void) {
  printf(“Hello, World!.\n");
  return 0;
} // end of main()
```

- This program might be stored in a file named `hello.c`.
- The file name doesn’t matter
    - `.c` extension may be required, usually useful
        - used by gcc to indicate language

Compilation Process

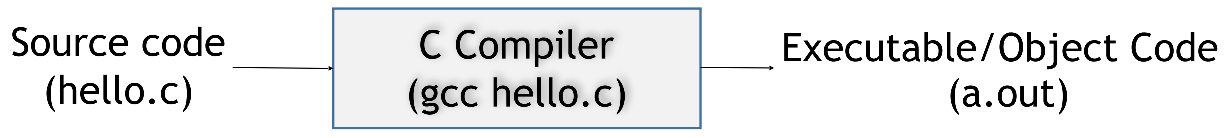
```
[xena@codewarrior cs246]$ gcc -lc -xc hello.c
```



```
[xena@codewarrior cs246]$ ./a.out
Hello, World!
[xena@codewarrior cs246]$
```

Compilation Process – GNU C Compiler

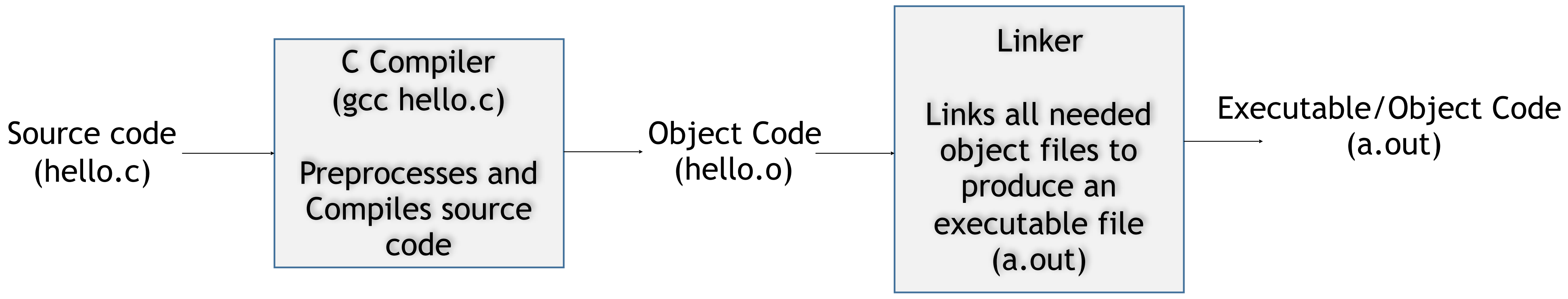
```
[xena@codewarrior cs246]$ gcc –o hello hello.c
```



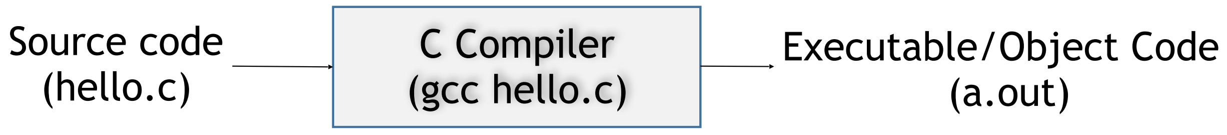
```
[xena@codewarrior cs246]$ ./hello
Hello, World!
[xena@codewarrior cs246]$
```

Compilation Process

Compilation is a 3-step process



The gcc command, in its simplest form, integrates all three steps.



Compilation Process

Compilation is a 3-step process

1. Preprocessing
   Source code commands that begin with a # are preprocessed. E.g.,

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

2. Compiling
   Source code is translated into object code (m/c language)
   Single pass … function order matters!

3. Linking
   All libraries/modules used by the program are linked to produce an executable object code

Preprocessing is normally integrated into the compiler. Linking is done by a separate program/command.

C Program Structure (for now)

directives

```
int main(void) {
    statements
}
```

```
#include <stdio.h>

int main(void) {
  printf(“Hello, World!.\n");
  return 0;
} // end of main()
```

- Before a C program is compiled, it is first edited by a preprocessor.
- Commands intended for the preprocessor are called directives.
- `<stdio.h>` is a ***header*** containing information about C’s standard I/O library.

`main()`

- The `main()` function is mandatory.
- `Main()` is special: it gets called automatically when the program is executed.
- `main` returns a status code; the value 0 indicates normal program termination.
- If there’s no `return` statement at the end of the `main` function, many compilers will produce a warning message.

Printing Strings

- The statement

  ```
  printf("To C, or not to C: that is the question.\n");
  ```

  could be replaced by two calls of `printf`:

  ```
  printf("To C, or not to C: ");
  printf("that is the question.\n");
  ```

- The new-line character can appear more than once in a string literal:

  ```
  printf("Brevity is the soul of wit.\n  --Shakespeare\n");
  ```

Comments – Two styles /*…*/ or //

- Begins with `/*` and end with `*/`.

```
/* No comment */
```

- Comments can also be written in the following way:

```
// No comment
```

- Advantages of `//` comments:
  - Safer: there's no chance that an unterminated comment will accidentally consume part of a program.
  - Multiline comments stand out better.

Another Program (variables, assignment, formatted output)

```
File: small.c
#include <stdio.h>

int main(void) {

    int A, B, C;

    A = 24;
    B = 18;
    C = A + B;

    printf(“C = %d\n”, C);
} // main()

[xena@codewarrior cs246]$ gcc -o small small.c
[xena@codewarrior cs246]$ ./small
C = 42
[xena@codewarrior cs246]$
```