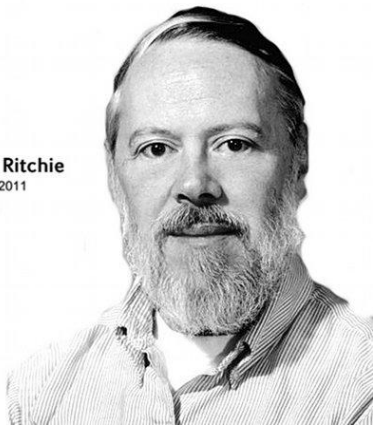


INTRODUCTION TO C

C - A HISTORY

- Derived from B (Basic)
 - Visual Basic is also derived from B
- Released by Dennis Ritchie in 1973
 - Also co-wrote Linux with Ken Thompson

Dennis Ritchie
1941-2011



WHY USE C?

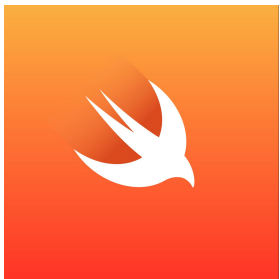
- It's fast:

<http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=nbody>

- Many large-scale programs (and other programming languages) are written in it
 - Linux
 - Python
 - Most compilers
 - Etc...

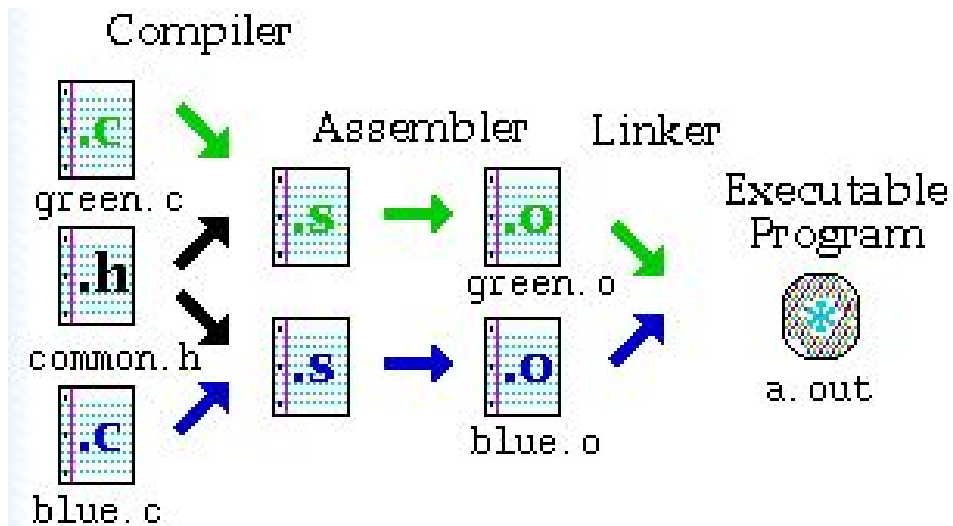
WHY DIFFERENT PROGRAMMING LANGUAGES?

- Each designed to optimize a certain task
 - Python: Ease of typing
 - Java: Integrated with web interfaces
 - R: Statistical Computing
 - Swift: Integration with Apple products



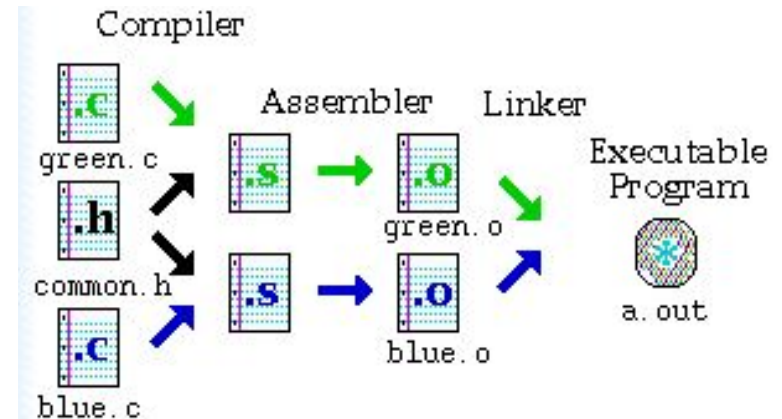
HOW DO THESE PROGRAMS ACTUALLY RUN?

- Code is *compiled*
 - Translates human-written code (*source code*) into binary (*machine code*)



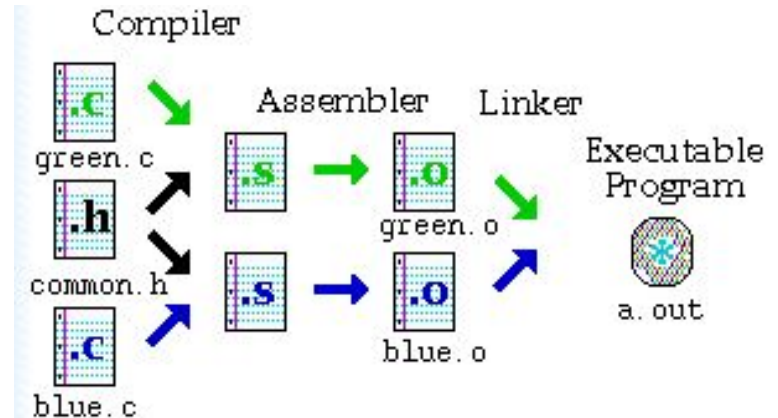
HOW DO THESE PROGRAMS ACTUALLY RUN?

- Compilation: Step 1 = Preprocessing
 - Essentially organizes code into easy-to-process files (from .c and .h files to .s))



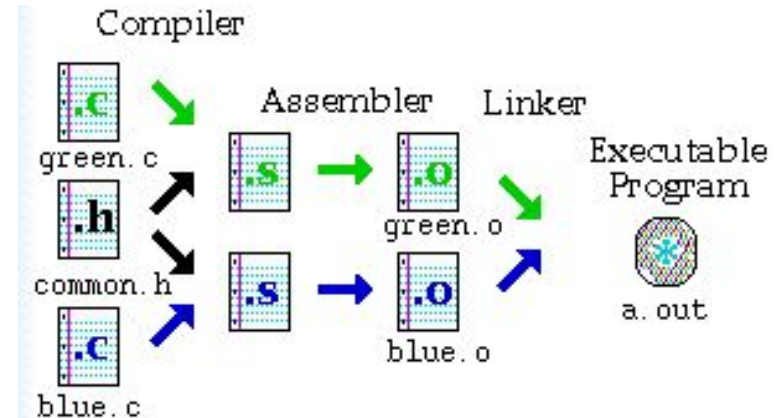
HOW DO THESE PROGRAMS ACTUALLY RUN?

- Compilation: Step 2 = Translation
 - The *Compiler* translates preprocessed code into object code - gives syntax errors



HOW DO THESE PROGRAMS ACTUALLY RUN?

- Compilation: Step 3 = Linking
 - The *Linker* takes the object code and combines all the code into an executable program



FIRST C PROGRAM

>> emacs hello.c

/*Use ctrl^x ctrl^s to
save*/

/*Use ctrl^x ctrl^c to
save and exit*/

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
    printf("Hello, world!\n");  
  
    return 0;  
}
```

COMPILATION

- To run preprocessor and compiler:

```
>> gcc -Wall hello.c -c -o hello.o
```

- To run linker:

```
>> gcc hello.o -o hello
```

- To execute program

```
>> ./hello
```



```
<hello-world />
```

COMPILATION - EASY WAY

- To run preprocessor, compiler, and linker:

```
>> gcc -Wall hello.c -o hello
```

- To execute program

```
>> ./hello
```



```
<hello-world />
```

C STANDARDS

Comments: Between `/*` and `*/`

Ignored by processors

Does not have to be
written in code

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
    printf("Hello, world!\n");  
    return 0;  
}
```

C STANDARDS

Preprocessor directives:
begin with `#include`

Include background code
that allows commands to
run

`<stdio.h>` includes the
code for the *printf*
command

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
    printf("Hello, world!\n");  
    return 0;  
}
```

C STANDARDS

Int main(void): File header

REQUIRED IN EVERY C PROGRAM

Without this, the code will not run

Serves as the “doorway” to the code

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
  
    printf("Hello, world!\n");  
  
    return 0;  
  
}
```

C STANDARDS

{ } : brackets show where the code is placed

Code outside brackets will give a compiler error

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
    printf("Hello, world!\n");  
    return 0;  
}
```

C STANDARDS

printf("Hello, world!\n");

Printf command: Prints the text inside the parentheses to the screen

\n: Newline character

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
  
    printf("Hello, world!\n");  
  
    return 0;  
  
}
```


C STANDARDS

printf("Hello, world!\n");

`\n`: Newline character.
Equivalent of hitting the
enter key in a Text Editor

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
    printf("Hello, world!\n");  
    return 0;  
}
```

C STANDARDS

printf("Hello, world!\n");

; : Semicolon determines
when the line of code ends

MUST BE INCLUDED AT THE
END OF EVERY LINE – gives
a compiler error if it
does not exist

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
  
    printf("Hello, world!\n");  
  
    return 0;  
  
}
```

C STANDARDS

return 0;

Tells the *int main(void)* function to terminate

Stops the code

Technically can be omitted: But is required for this class

```
/*This is a comment*/  
  
#include <stdio.h>  
  
int main(void) {  
  
    printf("Hello, world!\n");  
  
    return 0;  
  
}
```