

COMP26020 Programming Languages and Paradigms Part 1: C Programming

Introduction to C

C: Origin

- C was designed in the 70s by Dennis Ritchie at Bell Labs
- Originally used to develop applications for Unix, then to reimplement its OS kernel with Ken Thompson

C: Origin

- C was designed in the 70s by Dennis Ritchie at Bell Labs
- Originally used to develop applications for Unix, then to reimplement its OS kernel with Ken Thompson
- Ritchie & Thompson awarded the ACM Turing Award in 1983 *for their development of generic operating systems theory and specifically for the implementation of the UNIX operating system*

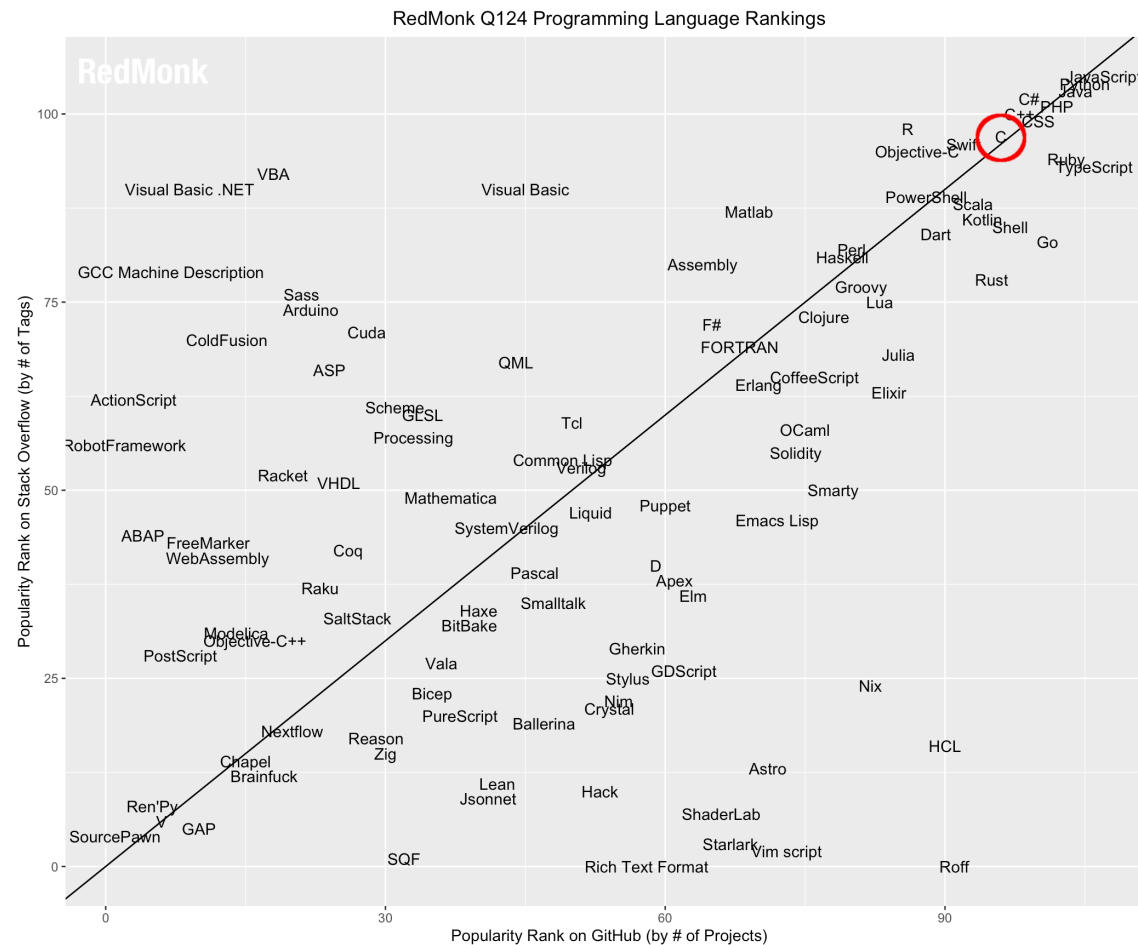


C: Popularity

- Old language,
but still
omnipresent
today
- In the top ten
of most
programming
languages
popularity
indexes

C: Popularity

- Old language, but still **omnipresent** today
- In the top ten of most programming languages popularity indexes



C: Characteristics

- **Pros:**
 - *Low-level*: close to the hardware, lots of freedom for the programmer to manipulate the machine
 - Fast, low memory footprint
 - Portable
 - Established a popular syntax

C: Characteristics

- **Pros:**
 - *Low-level*: close to the hardware, lots of freedom for the programmer to manipulate the machine
 - Fast, low memory footprint
 - Portable
 - Established a popular syntax
- **Cons:** programmer freedom comes at a cost: **a large area for making mistakes**
 - Lack of memory safety, risks of undefined behaviour at runtime

C: Characteristics

- **Pros:**
 - *Low-level*: close to the hardware, lots of freedom for the programmer to manipulate the machine
 - Fast, low memory footprint
 - Portable
 - Established a popular syntax
- **Cons:** programmer freedom comes at a cost: **a large area for making mistakes**
 - Lack of memory safety, risks of undefined behaviour at runtime

Still extensively used in: systems software, high performance computing, embedded systems, etc.

Popular Software Written in C



Linux

iOS



APACHE



git

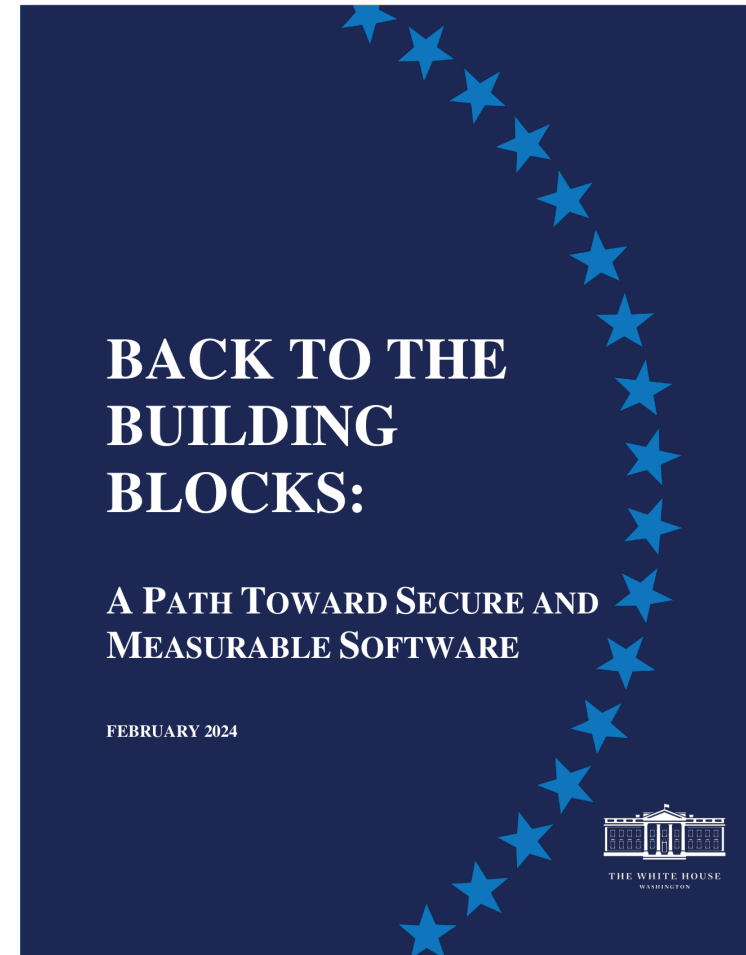


redis



Programming Mistakes in C

- Programming mistakes in C are common and lead to:
 - Build failing 😞
 - Crashes 😞
 - Program misbehaving 😡
 - Hard to reproduce bugs 😡
 - Security vulnerabilities 😱
- C is not particularly recommended for new projects
- Still a gigantic amount of legacy C code running today that not going anywhere anytime soon



Why Learn C Today?

- Still the default programming languages in many different domains
- C is not going anywhere anytime soon, vast amount of C codebases still relevant today
 - We will need C experts in the decades to come
- Proximity to the hardware: knowing C helps understanding how a computer works
- Knowing C's syntax will make it easy to learn many other programming languages

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

[03-c-introduction/hello-world.c](#) 

Assuming the code is in `hello-world.c`, to compile and run on the Linux command line:

```
$ gcc hello-world.c -o hello
$ ./hello
hello, world!
```

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries
- `main` will run first when the program is executed

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries
- `main` will run first when the program is executed
 - It returns an integer (`int`) and takes no parameters

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries
- `main` will run first when the program is executed
 - It returns an integer (`int`) and takes no parameters
 - Contains two statements, in C statements ends with `;`

Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries
- `main` will run first when the program is executed
 - It returns an integer (`int`) and takes no parameters
 - Contains two statements, in C statements ends with `;`
- `printf` used to print text on the *standard output* (console)

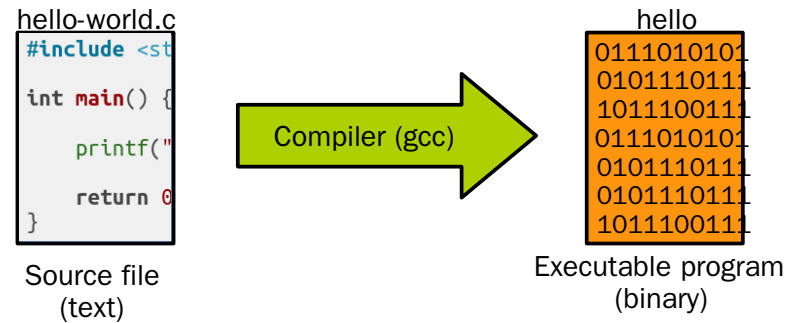
Hello World

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");
    return 0;
}
```

- `#include` to get access to functions from external libraries
- `main` will run first when the program is executed
 - It returns an integer (`int`) and takes no parameters
 - Contains two statements, in C statements ends with `;`
- `printf` used to print text on the *standard output* (console)
- `return` to go back to the calling context, and exit if returning from `main`

Compilation



```
$ gcc hello-world.c -o hello
```

Compilation

```
#include <stdio.h>

int main() {
    printf("hello, world!\n");

    return 0;
}
```

Compilation

```
#include <stdio.h>

nt main() {
    printf("hello, world!\n");

    return 0;
}
```

```
$ gcc hello-world.c -o hello
hello-world.c:3:1: error: unknown type name 'nt'; did you mean 'int'?
  3 | nt main() {
    |   ^~
    | int
```

Compilation

```
#include <stdio.h>

nt main() {
    printf("hello, world!\n");

    return 0;
}
```

```
$ gcc hello-world.c -o hello
hello-world.c:3:1: error: unknown type name 'nt'; did you mean 'int'?
  3 | nt main() {
    |   ^~
    | int
```

Multiple issues? fix them in the order they are reported

Compilation: Warning and Errors

- Errors are unrecoverable
- Warnings *may* indicate a problem

```
#include <stdio.h>

int main() {
    int x;
    printf("hello, world!\n");

    return 0;
}
```

```
gcc -Wall hello-world.c -o hello
hello-world.c: In function 'main':
hello-world.c:4:9: warning: unused variable 'x' [-Wunused-variable]
    int x;
    ^
```


Compilation: Warning and Errors

- Errors are unrecoverable
- Warnings *may* indicate a problem
- **Warnings almost always indicate a problem and you should fix them**
- Some warnings (picky ones) are disabled by default, enable them with the `-Wall -pedantic` flags

Comments

```
/* listing2.c, illustrate the use of comments */  
  
#include <stdio.h> // necessary to get access to printf  
  
/* this function simply prints out 'hello world' and returns */  
int main() {  
  
    printf("hello, world!\n"); // here we print ...  
  
    /* the line below will not be compiled: */  
    // printf("goodbye!\n");  
  
    return 0;                /* ... and return */  
}
```

[03-c-introduction/comments.c](#) 

```
/* style 1 */  
// style 2
```

Use comments to explain what your code does, uncommented code is hard to understand

Summary

- Brief intro to C
 - Hello world
 - Compilation process
 - Comments
-

Feedback form: <https://bit.ly/3yzIVIs>

