A Software's Life Cycle

Professor Hossein Saiedian

EECS 348: Software Engineering

Spring 2023

KU THE UNIVERSITY OF KANSAS

A software's life cycles

- Development life cycle
- Maintenance life cycle
- Runtime life cycle

Development life cycle

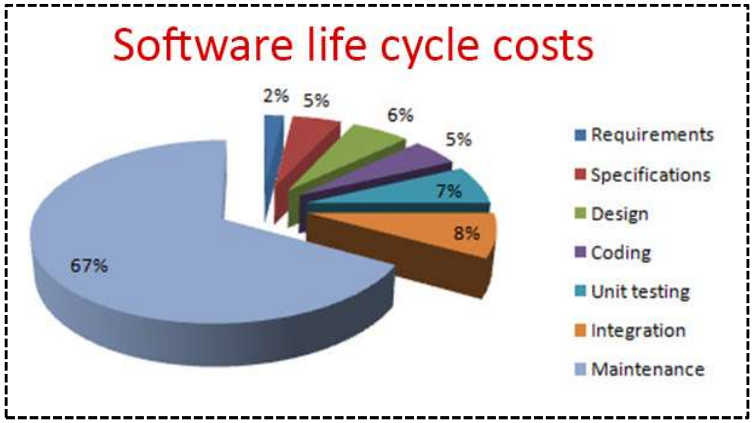
- In software engineering, we emphasize the *development* life cycle

Major development activities

- **Requirements engineering**: to identify, model, analyze, document, and validate the requirements
- **Design**: to develop a software solution that addresses the requirements
- **Construction**: to convert the design (solution) into code
- **Testing**: to conduct various testing techniques to identify and remove defects
- Different **development models** define the specific steps and ordering of the above activities

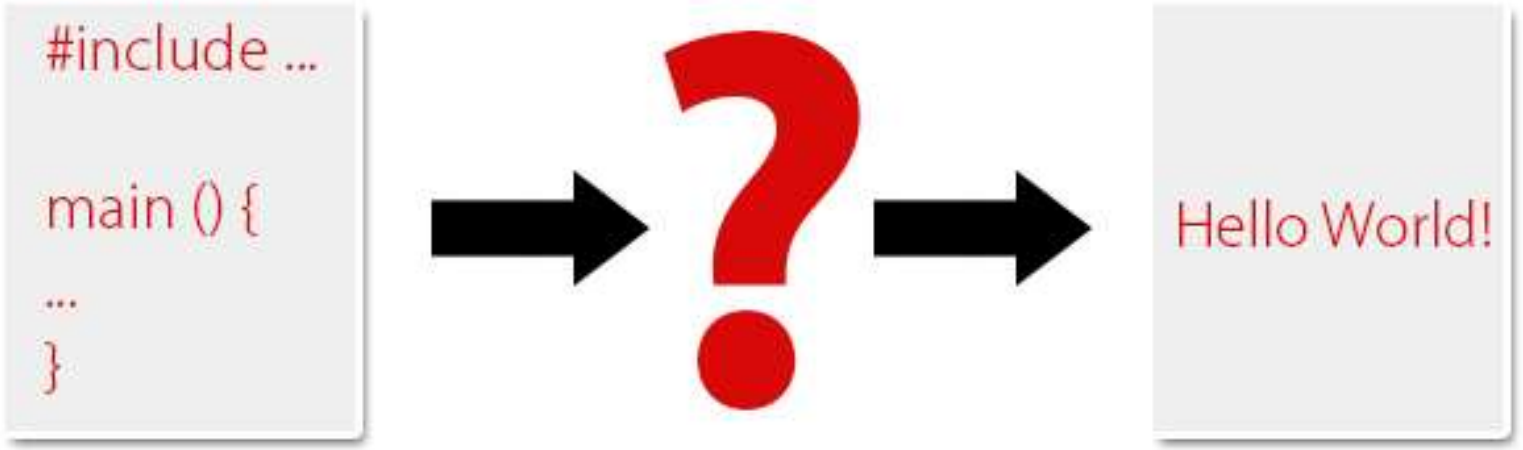
A program's maintenance life cycle

- In software engineering, we also emphasize the *maintenance* (or evolution) life cycle
- Types of maintenance
  - Corrective: to remove defects
  - Perfective: to add new features
  - Adaptive: to adapt to changes
  - Preventative (refactoring)

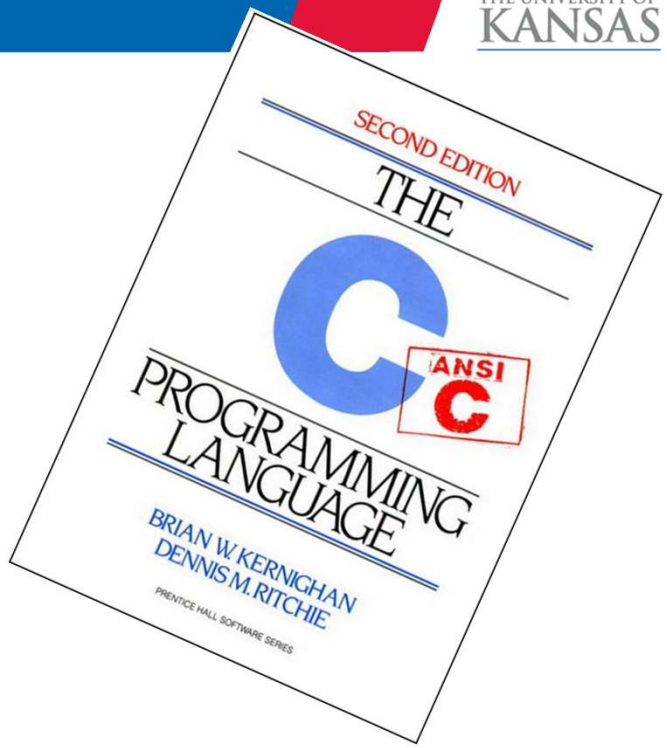


What about a program's internal life cycle?

- But what happens to a program at run time?



The “Hello, world!” program



- From the classic *K&R C* book

```
#include <stdio.h>
int main () {
    printf(“Hello, world!\n”);
}
```

- Millions of copies sold; translated to 25 languages

The “Hello, world!” program in ASCII

- Most modern systems represent characters using the ASCII standard

```
#include <stdio.h> int main () {printf("Hello, world!\n");}
```

35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
104 62 32 105 110 116 32 109 97 105 110 32 40 41 32 123 112
114 105 110 116 102 40 34 72 101 108 108 111 44 32 119 111
114 108 100 33 92 110 34 41 59 125

The “Hello, world!” program in binary

- All characters are internally stored in binary

```
#include <stdio.h> int main () {printf("Hello,
world!\n");}
```

00100011 01101001 01101110 01100011 01101100 01110101 01100100
01100101 00100000 00111100 01110011 01110100 01100100 01101001
01101111 00101110 01101000 00111110 00100000 01101001 01101110
01110100 00100000 01101101 01100001 01101001 01101110 00100000
00101000 00101001 00100000 01111011 01110000 01110010 01101001
01101110 01110100 01100110 00101000 00100010 01001000 01100101
01101100 01101100 01101111 00101100 00100000 01110111 01101111
01110010 01101100 01100100 00100001 01011100 01101110 00100010
00101001 00111011 01111101

Compiling the "Hello, world!" program

- Let's suppose we name our program `hello-pgm.c`
- To compile the program on a Linux system we enter the following

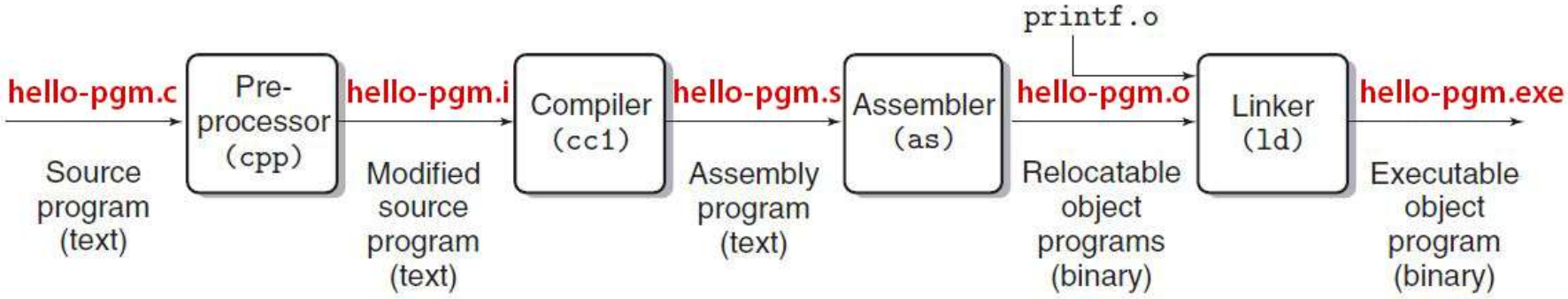
```
unix> gcc -o hello-pgm.exe hello-pgm.c
```

where

`gcc` is the C compiler

`hello-pgm.exe` will be the executable code

A (simplified) compilation process



- The preprocessor (`cpp`) handles directives such as `#include` (e.g., inserts `stdio.h` into program)
- The compiler (`cc1`) translates to assembler code
- The assembler (`as`) translates into machine instructions
- The linker (`ld`) merges in other programs (or library executables) needed for the final functionality

Running the executable object code

- To run at command line, we enter

```
unix> hello-pgm.exe
```

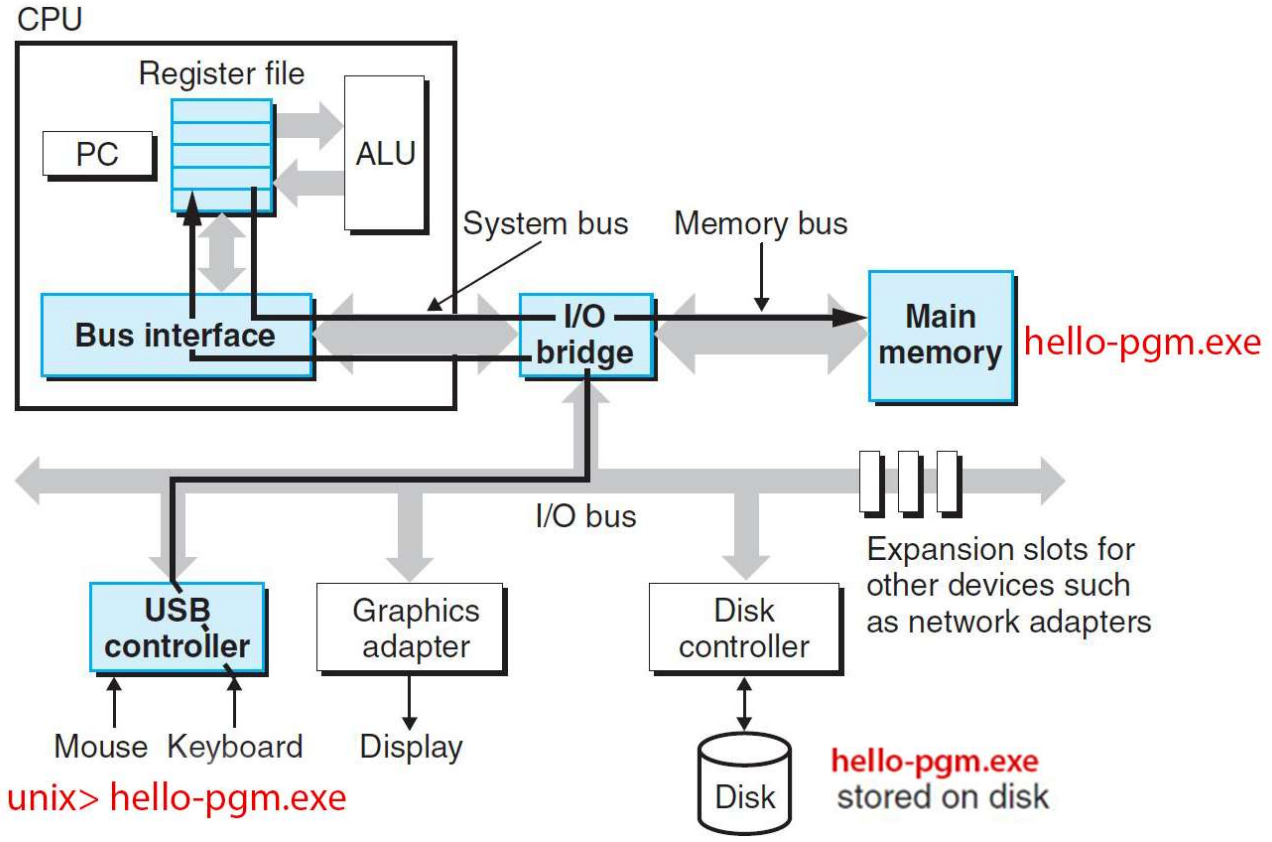
- We will get the following output

```
Hello, world!
```

- How does this happen inside our computer system?

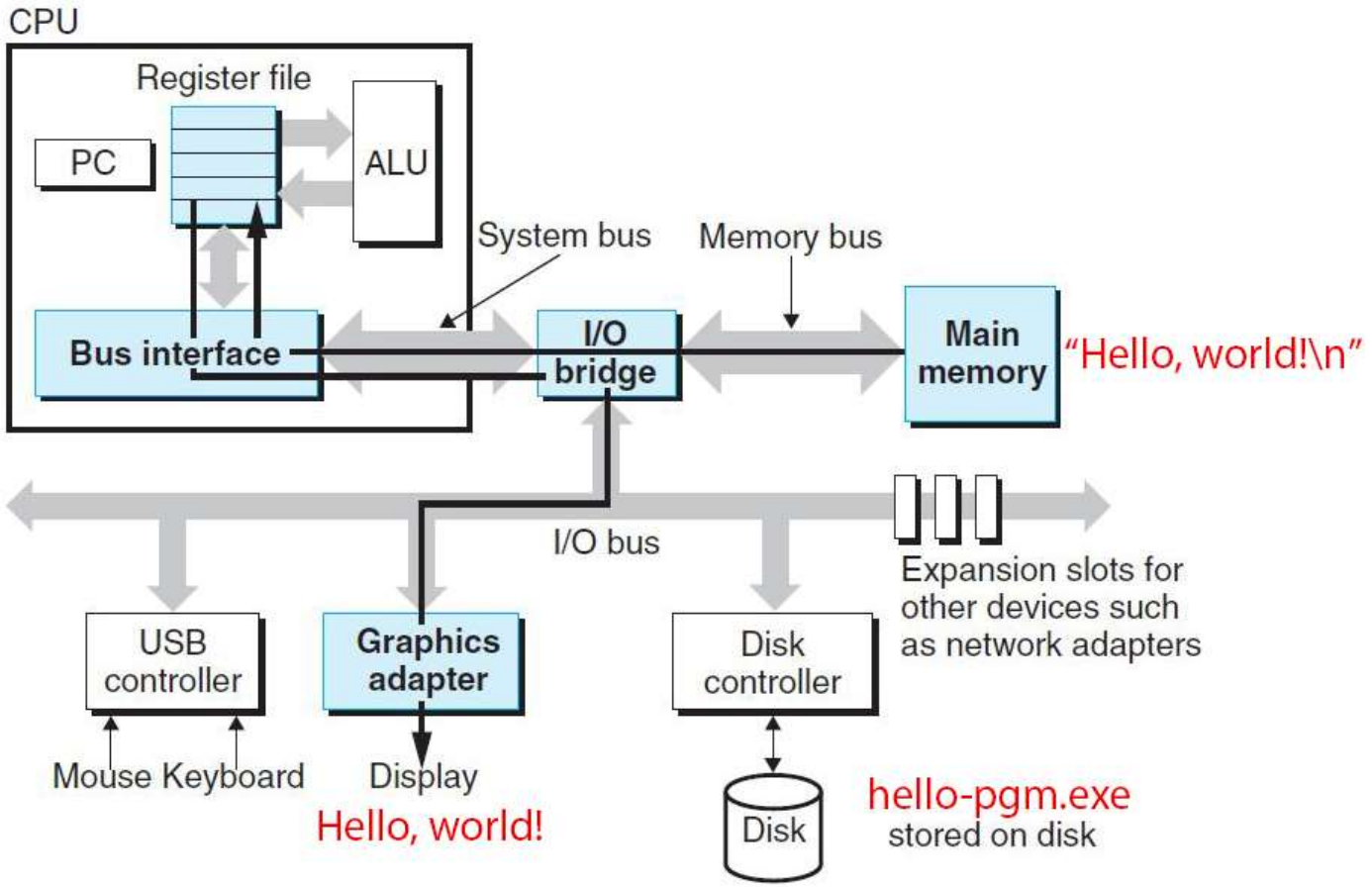
How the "running" happens

- User enters `hello-pgm.exe`; each character is read from the keyboard, stored into a register, then stored in memory; the system will look for the program to execute



The program output is displayed

- Once the program instructions are executed, the results (output) is moved from the register to the display device



A program's life cycles: Summary

KU THE UNIVERSITY OF KANSAS

- It is important that we understand a program's various life cycles
- We very briefly looked at three life cycles
  - **Development** (focus of the course)
  - Maintenance
  - Internal (in a computer system)
- It is important to note that many individuals, processes, and computer components are involved