CS-2060 Technical Documentation

GE01 – January 22, 2024

Table of Contents

Setting up My Environment:

- CS2060 Set Up Environment and Version Control
- Choose C17 as standard for **all** projects.
- Ensure Clangy-tidy is being utilized alongside the Clangy compiler when compiling and analyzing code.
- Stdbool.h is the only library used that **is not** included in the standard library.

Versioning with Git and Backing up with Github:

- In git bash ensure the terminal is running in the repo's directory.
- Git commands to remember.
    - Git status → check which files have been modified and whether they have been committed.
    - Git add –all → adds all modified files and makes ready to commit.
    - Git commit -a -m "Example Description" → commits all files to master that have been added.
    - Git push origin → pushes Repo edits to server.
    - Git pull → pulls the latest edits sent to the server.

Important Resources:

- C Standard Library Reference
- C Standard Header Files

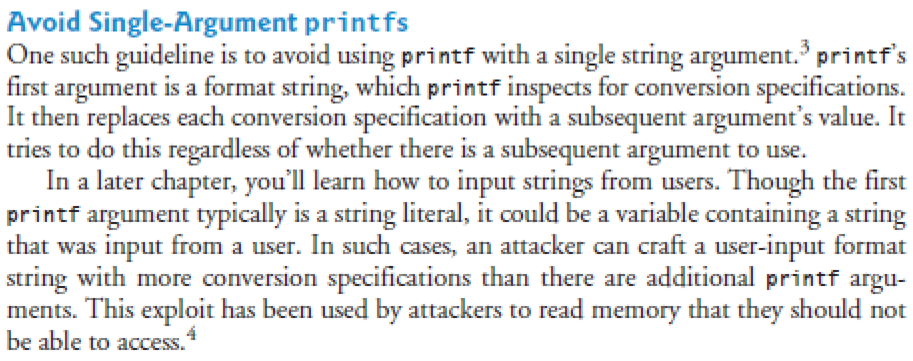
- Format Specifiers in C
- Textbook - "How to Program In C 9th Edition"

Secure C Coding (Chapter 2):

- SEI CERT C Coding Standard

1. Avoid Single Argument printfs

"C How to Program" Chapter 2.7
An attacker can craft a user-input format string with more conversion specifications than there are additional printf statements allowing them to read from memory. Utilizing \n in your string will create and display a new line after the function runs, keeping the user from being able to read the memory from the printf function.


Avoid Single-Argument printfs
One such guideline is to avoid using `printf` with a single string argument.[3] `printf`'s first argument is a format string, which `printf` inspects for conversion specifications. It then replaces each conversion specification with a subsequent argument's value. It tries to do this regardless of whether there is a subsequent argument to use.

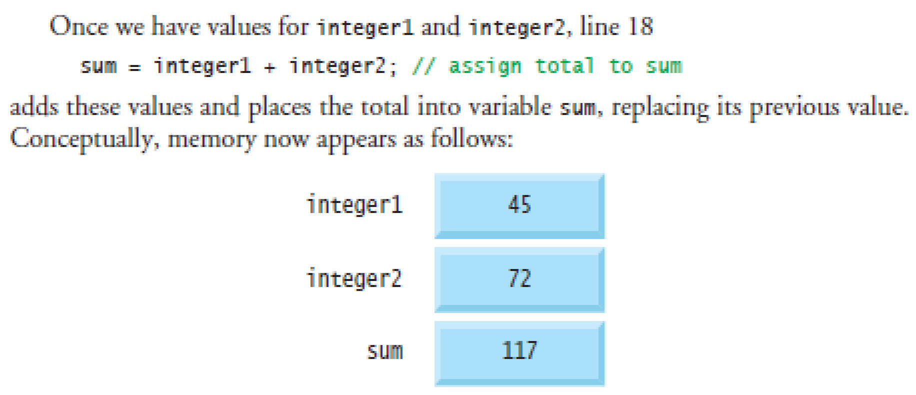
In a later chapter, you'll learn how to input strings from users. Though the first `printf` argument typically is a string literal, it could be a variable containing a string that was input from a user. In such cases, an attacker can craft a user-input format string with more conversion specifications than there are additional `printf` arguments. This exploit has been used by attackers to read memory that they should not be able to access.[4]

2. Scanf, printf, scanf_s and printf_s

"C How to Program" Chapter 2.7
Scanf_s and printf_s will not be utilized as they are a part of annex K which is implemented by Windows. Rules surrounding the use of scanf and printf will be described in depth in Section 3.13 of the textbook.

Intro to Memory (Chapter 2):

- **All** variables have → name, type, value, and location in memory.
- A variable's value only changes if it is being redefined.
  - **Scanf("%d", &integer1);** → sets **integer1** to the scanned value. This can be utilized in the program without changing its value.
  - **Integer1 = 5;** → changes the value of **integer1** (if already defined) to the new value 5.



Once we have values for integer1 and integer2, line 18

```
sum = integer1 + integer2; // assign total to sum
```

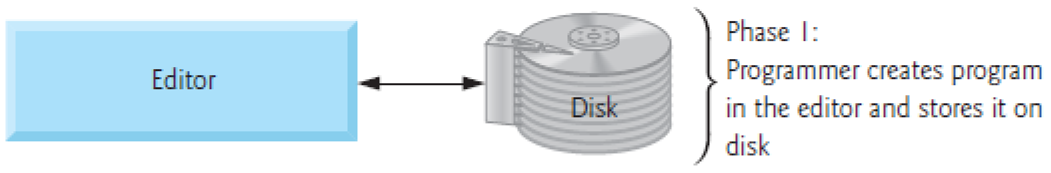
adds these values and places the total into variable sum, replacing its previous value. Conceptually, memory now appears as follows:

integer1	45
integer2	72
sum	117

Typical Phases of C Development Environment (Chapter 1):

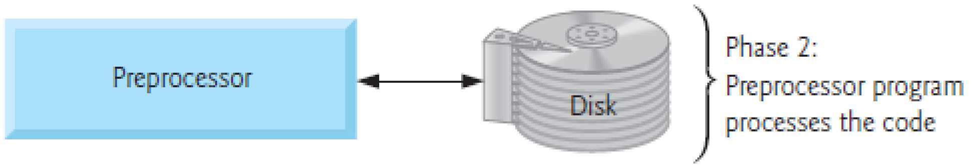
(Book, Pg. 21 - 24)

1. Phase I: Creating a Program

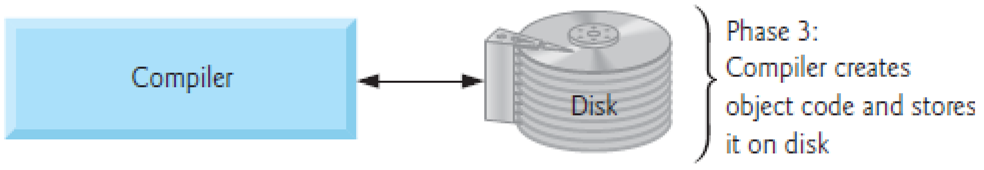

Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk

- Popular editors on **Linux** include vi and emacs
- **Windows** and **Apple** devices have editors included in their IDEs such as Microsoft Visual Studio and Apple Xcode
- C programs end with a .c extension

2. Phase II and III: Preprocessing and Compiling a C Program

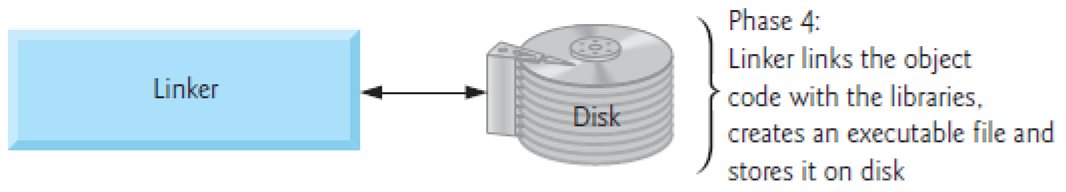


- The compilation command invokes the preprocessor before the translation of code.
  - See The C Preprocessor



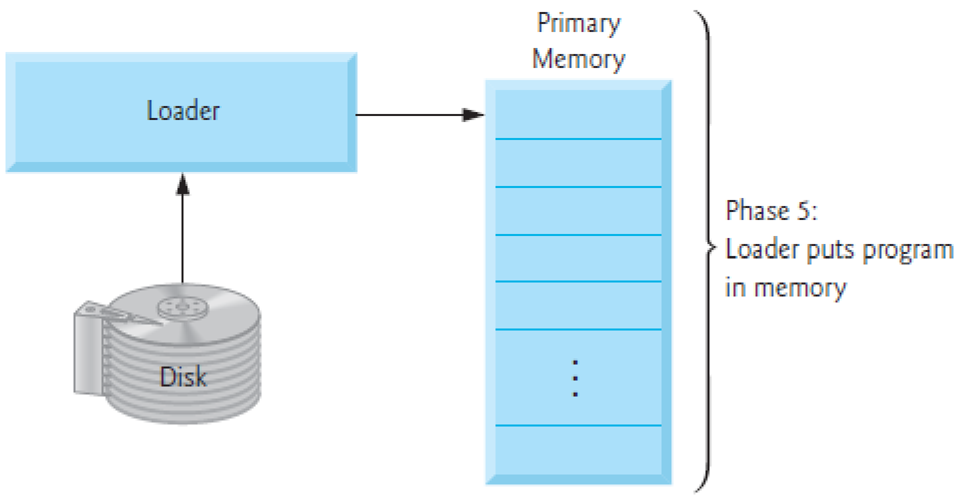
- Compiler translates C language into “machine code” or “object code.”
  - See Compilation Process in C

3. Phase IV: Linking


Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk

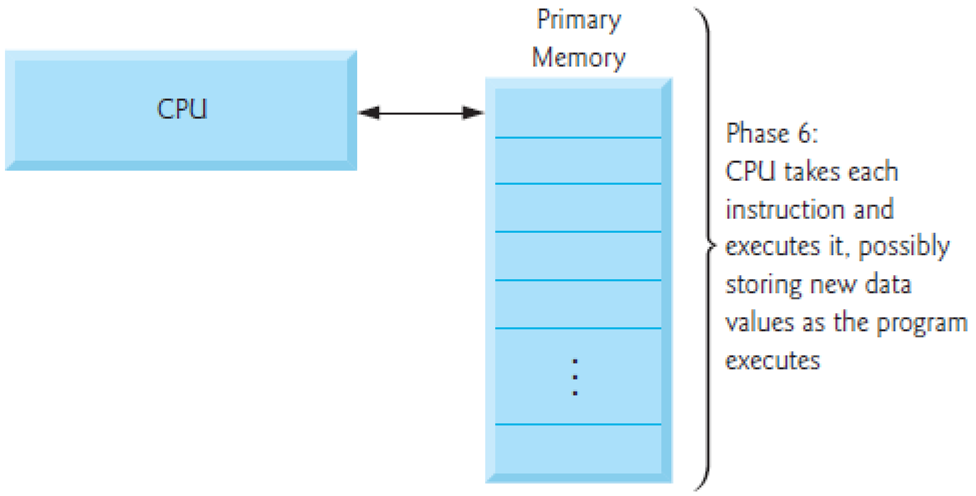
- Linker links files from public, open source, or private libraries to “fill in the holes” in the program to produce an executable image.
- To compile and link a program using C18, type **gcc -std=c18 exampleprogram.c**

4. **Phase V: Loading**



   a. As shown in the diagram, this phase simply loads the program into the memory of the hardware being used.

5. **Phase VI: Execution**


Primary Memory
CPU
Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes

   a. Lastly, the computer processes the program using the CPU one instruction at a time from the “stack” of code.
      i. To load a program in Linux, type ./a.out and press *Enter*

Standard Input, Output, and Error Streams:

- Runtime Errors occur when the program compiles but runs into an error later in the code (Dividing by zero).
- Standard input/output streams (stdin/stdout)
- Standard error streams (stderr)