(Lab 5) C Programming - Compilation, Syntax, Semantics

CS2013 Systems Programming

Department of CSE, IIT Palakkad

IIT Palakkad

Aug 28, 2025

Quiz 5 (15 minutes, **Do not copy the question**)

1. If we run these two commands, will `output.txt` obtained differ ? Explain your reasoning in 2-3 short sentences.

```
$ find log /var 2>&1 | sort > output.txt
```

```
$ find log /var | sort > output.txt 2>&1 > /dev/null
```

2. Sonu and Monu share a repo with `master` as the only branch. Sonu created a new branch named `bindass`, added some code and pushed. Monu pulled from the repo but could not find any change in the repo. What advice will you give Monu so that Monu can see the pushed code ?

3. Your home already contains a file named `data`. What happens if you try to run the following and why ?

```
$ mkdir data
```

Plan

- Compiling a C program
- Syntax of C programs vs Python
- Semantics of C language

Terminology

Syntax = Structure   Semantics = Meaning

Reduce friction in Git use

- Configure your bash for git usage
- Run `$ find /usr/share/doc -name git-prompt.sh`

Reduce friction in Git use

- Configure your bash for git usage
- Run `$ find /usr/share/doc -name git-prompt.sh`
- Open the file with nano/vim
  - `$ nano <path_found>`
- Read the comments
- Implement them !

Compiler and Compilation

- Recap: working with assembly difficulty
  - Why ? needs deep understanding of machine architecture.

Compiler and Compilation

- Recap: working with assembly difficulty
  - Why ? needs deep understanding of machine architecture.
- Compiler helps focus the programmer on solving the problem
- Demo Hello world: Python versus C
- Running python versus C

Compiler and Compilation

- Recap: working with assembly difficulty
  - Why ? needs deep understanding of machine architecture.
- Compiler helps focus the programmer on solving the problem
- Demo Hello world: Python versus C
- Running python versus C
- Compiler, Interpreter, Hybrid (Just-in-time compilation)
- Preprocessor and its requirement

C program from creation to execution

- Demo
- Object code and Assembly
- $\xrightarrow[\text{.c code}]{\text{Your program}}$ C Preprocessing $\xrightarrow[\text{.c code}]{\text{macro subst}}$ Compilation $\xrightarrow[\text{.o Object file}]{}$ Linking $\xrightarrow[\text{Executable}]{}$ Loading $\xrightarrow[\text{code}]{\text{relocatable}}$ Execution
- Role of linker (covered in later slides)
- Role of loader (covered in later slides)

C language standards

- First one - Kernighan and Richie (K&R C language)
- Initial version C89. Designed by ANSI
- Underwent changes in 90s. Another standard C99.
- This course: mostly stick to C99
- Others - C95, C11, C17, C23 (all ISO standards)
- Why standards ? Too many variants, compilers, interoperability, issues

Types, Variables and Scope

- Basic types in C
    - char
    - int
    - long
    - float
    - double
- All variables have scope given by braces $\{ \}$.
- Sample code
- Line in C end with semi-colon (;)

Types, Variables and Scope

- Basic types in C
    - char
    - int
    - long
    - float
    - double
- All variables have scope given by braces { }.
- Sample code
- Line in C end with semi-colon (;) except when then don't !

Types, Variables and Scope

- Basic types in C
    - char
    - int
    - long
    - float
    - double
- All variables have scope given by braces { }.
- Sample code
- Line in C end with semi-colon (;) except when then don't !
- **Variables needs to be declared and initialized before use**

C Types and declaration

Type name	Usual size*	Values stored	How to declare
char	1 byte	integers	char x;
short	2 bytes	signed integers	short x;
int	4 bytes	signed integers	int x;
long	4 or 8 bytes	signed integers	long x;
long long	8 bytes	signed integers	long long x;
float	4 bytes	signed real numbers	float x;
double	8 bytes	signed real numbers	double x;

* – depending on architecture

Signed versus Unsigned

- Signed versus unsigned (for short, int and long).

Signed versus Unsigned

- Signed versus unsigned (for short, int and long).
- `int x` - signed integer variable x;
    - Values: -2^{31} to $2^{31} - 1$.
- `unsigned char x` - unsigned character variable x;
    - Values: 0 to $2^{8} - 1$.

Signed versus Unsigned

- Signed versus unsigned (for short, int and long).
- `int x` - signed integer variable x;
  - Values: -2^{31} to $2^{31} - 1$.
- `unsigned char x` - unsigned character variable x;
  - Values: 0 to $2^8 - 1$.

Trick: If the type uses n bytes (short/int/long ONLY), then

Range for n bytes (short, int and log)

- unsigned range is 0 to $2^{8n} - 1$.
- signed range is -2^{8n-1} to $2^{8n-1} - 1$.
- float, double as per IEEE 754 representation.

System Design (Arch + OS + Userprogs)

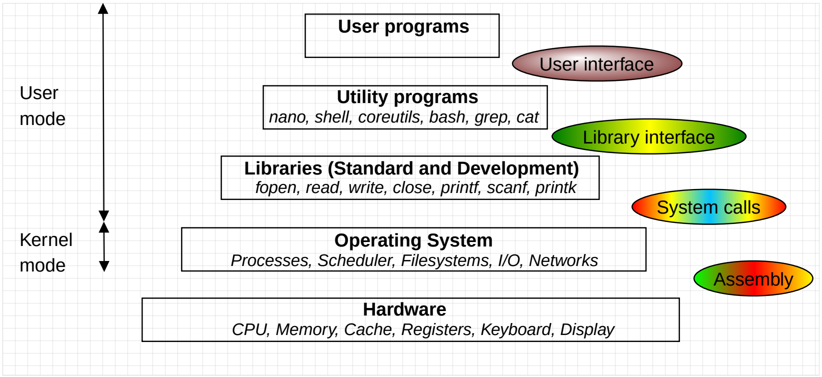
User programs
User interface
User mode
Utility programs
nano, shell, coreutils, bash, grep, cat
Library interface
Libraries (Standard and Development)
fopen, read, write, close, printf, scanf, printk
System calls
Kernel mode
Operating System
Processes, Scheduler, Filesystems, I/O, Networks
Assembly
Hardware
CPU, Memory, Cache, Registers, Keyboard, Display

Figure 1: Systems Design

Linux and C

- Everything is a file, File descriptor
- Libraries (`libc`, `klibc`)
- System calls
  - `open`, `write`, `read`, `...`
- How it is done ? (Linker and Loader)
- C variants of System Calls
  - `fopen`, `printf`, `scanf`

Plan (rest of the session)

Demonstrate C features

Operations and Expressions

- Show demo

Assignment Operator

- The equality operation
- Use and dangers
- Demo

Logical operators

- Given example code

Repetitions

- for loop
- example code in python and C

Conditionals, Repetitions

- if and while
- example code in python and C

C System calls and Library functions

- examples, scanf, printf

Data Type	Format Specifier
Integer	`%d`
long	`%ld`
Float	`%f`
Character	`%c`
Double	`%lf`

Lab Exercise

Questions (Do the following in your repo)

- Do `$ git fetch && git merge`
- Do `$ git switch lab05` to see the `questions.md` and folders created
- Not seeing anything ? Call TA/instructor
- **To push changes: do `$ git push -u origin lab05`**

Class repo (for in-class demo)

- Accessible via
  - `git clone git@gitserver:class_repo`
- To see latest changes, cd to the class_repo and do
  - `git fetch && git merge`
  - This does a git pull

Class has ended

- No more pushes to `gitserver`.
- Complete the exercises during off-lab hours.

Humble Request

Please keep the chairs in position before your leave.
(as a token of respect for our CFET staff)