(Lab 4) Systems Architecture & Basics of C Programming

CS2013 Systems Programming

Department of CSE, IIT Palakkad

Aug 21, 2025

Quiz 4 (15 minutes, **Do not copy the question**)

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

   ```
   $ mkdir data
   ```

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

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

3. A `bc` command is running in your terminal session. To ...

   a. ...suspend the execution - press `Ctrl`+ ____ **(1)**
   b. ...resume the suspended process - type ____, press enter **(2)**
   c. ...stop the execution - press `Ctrl` + ____ **(3)**

   Clearly fill in the blanks (1), (2) and (3) appropriately.

Plan

- Deeper understanding of Git
- Architecture, Operating System and User programs
- Basics of C programming
- Making Git more friendly

Understanding Git better

- Demo (live)

Understanding Git better

- Demo (live)
- Demo to understand git commands

Writing shell scripts

- Demo for guessing game
  - Writing a small game

Writing shell scripts

- Demo for guessing game
  - Writing a small game
- Cool features in shell script
  - Reading input
  - Hash bangs
  - Quiet execution !
  - Functions
  - Alias

Systems Architecture - Assembly

- Assembly
  - language in which computer speaks - zeros and ones
  - looks like
    - `add R10 R8 R12` (MIPS, RISC)
    - `CMOVcc EAX r` (Intel x86, CISC)

Systems Architecture - Assembly

- Assembly
  - language in which computer speaks - zeros and ones
  - looks like
    - `add R10 R8 R12` (MIPS, RISC)
    - `CMOVcc EAX r` (Intel x86, CISC)
- Useful in . . .
  - very restricted hardware
  - vulnerability analysis
  - critical parts of software written in assembly

Systems Architecture - Assembly

- Assembly
  - language in which computer speaks - zeros and ones
  - looks like
    - `add R10 R8 R12` (MIPS, RISC)
    - `CMOVcc EAX r` (Intel x86, CISC)
- Useful in . . .
  - very restricted hardware
  - vulnerability analysis
  - critical parts of software written in assembly
- Downside - full understanding of the underlying hardware to write program

Why use an Operating system ?

- Purpose of an OS
  - drives hardware + provides a comfortable env for user
  - allows fair allocation and management of resources
  - offers stability, security and isolation
- Privileged and unprivileged modes
- System calls and why do we need them

System Design (Arch + OS + Userprogs)

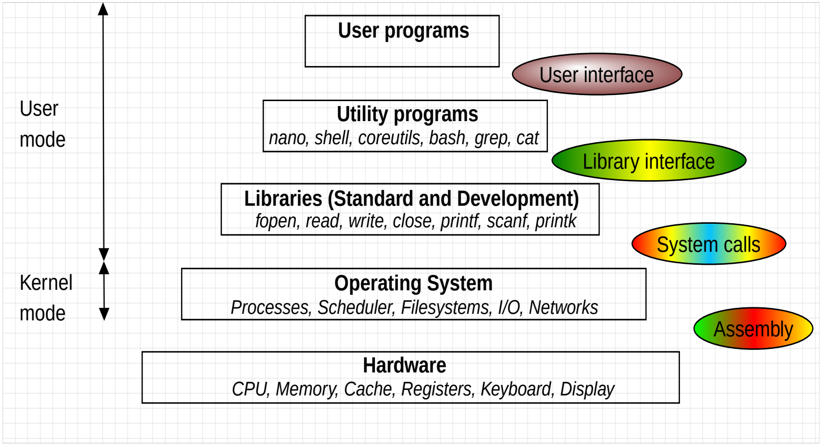
User programs
User interface
User mode
Utility programs
nano, shell, coreutils, bash, grep, cat
Library interface
Libraries (Standard and Development)
fopen, read, write, close, printf, scanf, printk
System calls
Kernel mode
Operating System
Processes, Scheduler, Filesystems, I/O, Networks
Assembly
Hardware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

Figure: Systems Design

A subset of OS features

- Address space isolation
- Paging, page tables and address translation
- Managing shared resources - I/O, Memory, Disk, Network. How ?
- . . . by providing a system call interface
- . . . by providing a device interface

Next

Basics of C Programming

Need for types

- Demo shell script

Need for types

- Demo shell script
- One person's constant is another person's variable !
- Computers don't understand types. Why have types ?

Need for types

- Demo shell script
- One person's constant is another person's variable !
- Computers don't understand types. Why have types ?
- Low level - computer sees only 0s and 1s. How to interpret data ? Types does this job.

Need for types

- Demo shell script
- One person's constant is another person's variable !
- Computers don't understand types. Why have types ?
- Low level - computer sees only 0s and 1s. How to interpret data ? Types does this job.
- Fix type of a variable = Fix domain of a variable (in Math)
- Can help catch lots of errors

The C programming language

- Designed - Dennis Ritchie and Ken Thompson (1973).
- Avoid directly writing assembly.
- Lacks many features of OOP languages (like in Python/C++/Java)
- Lacks high-level features (strings, dictionary, lists, sets)
- Want a feature ? Implement it from scratch !

The C programming language

- Designed - Dennis Ritchie and Ken Thompson (1973).
- Avoid directly writing assembly.
- Lacks many features of OOP languages (like in Python/C++/Java)
- Lacks high-level features (strings, dictionary, lists, sets)
- Want a feature ? Implement it from scratch !
- Closest to the hardware. Hence runs really fast !
- Linux and Mac kernel – written in C. Provides a C API (Application Programming Interface)
- Question: How to run a C program ?

Lab Exercise

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`
  - does a git pull

Exercise 1 (Game)

- Go to: `http://10.129.4.1/cs2013/lab04/`
- Download `oh-my-git-linux.zip` to `Downloads` and unzip the file to `Downloads`.
- Do `$ chmod +x Downloads/oh-my-git-linux/oh-my-git` (To be done in home directory).
- Do `$ ./Downloads/oh-my-git-linux/oh-my-git`

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)