Systems Software

Jay Prakash

jai.cse.iitkgp@gmail.com

Lecture Dates and Schedule

- Jan – 6,8,9,13,15,16,20,22,23,27,29-30(IS)
- Feb – 3,5,6,10,12,13,17,19,20,24,26,27
- Mar – 2,4-5(IS),8-13(Brk),16,18,19,23,25,26,30
- Apr – 1,2(H),6(H),8,9,13,15,16,20,22,23

- Lectures - Mon–10 AM, Wed–9 AM, Thu–11 AM
  - Makeup (if needed) – Weekdays – 8 AM / Saturday

- Lab – From coming week, policy to be announced shortly.

Key Points

- Pedagogy – Theory, algorithms, mathematics, programming.
- Nature of QP – Short theory, concept visualization, problem solving, program execution and understanding.
- Grade (Relative) components
  - Class Tests / Assignments (max 3) / Reading – 15%
  - In-Sem exams – 35 marks each (1 hour)
    - First In-Sem - 15%
    - Second In-Sem – 20%
  - Final exam – 100 marks (3 hours) - 50% (Complete syllabus)
  - !Attendance (Handy in case of border-line or F grades)
- Google Classroom / Moodle – Course Materials
  - Register yourself after you receive the course link.

Why this course?

Systems software are the core of every system (e.g., Computer - OS) – It is magic, unknown, frustrating, and/or scary to most people.

Goals for Today

- Why take this course on Systems Software?
  - Almost everywhere with evolving technologies
  - See next slide
- What is Systems Software?
- Issues in Software Systems Design
  - Design Principles

Functionality comes with complexity?

- Every piece of computer hardware different
  - Different CPU - Pentium, PowerPC, ARM, MIPS
  - Different amounts of memory, disk, ...
  - Different types of devices
    - Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - Cable, DSL, Wireless, Firewalls,...

Key Questions

- Questions:
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?

The OS paradigm

Two main functions:

Manage physical resources:

- It drives various devices
  - Eg: CPU, memory, disks, networks, displays, cameras, etc
- Efficiently, reliably, tolerating and masking failures, etc

Provide an execution environment to the applications running on the computer (programs like Word)

- Provide virtual resources and interfaces
  - Eg: files, directories, users, threads, processes, etc
- Simplify programming through high-level abstractions
- Provide users with a stable environment, mask failures

System design is Complex

- OS are a class of exceptionally complex systems
  - They are large, parallel, very expensive, not understood
    - OS implementation : years, large groups of people
  - Complex systems are the most interesting:
    - Internet, air traffic control, weather forecast, etc.
- How to deal with this complexity?
  - Abstractions and layering
  - Goal: systems trusted with sensitive data and critical roles

What are the issues in OS / System Design?

- Structure: how is an operating system organized ?
- Sharing: how are resources shared among users ?
- Naming: how are resources named by users or programs ?
- Protection: how is one user/program protected from another ?
- Security: how to authenticate, control access, secure privacy ?
- Performance: why is it so slow ?
- Reliability and fault tolerance: how do we deal with failures ?
- Extensibility: how do we add new features ?

What are the issues in OS / System Design? ...cont'd

- Communication: how can we exchange information ?
- Concurrency: how are parallel activities created and controlled ?
- Scale, growth: what happens as demands or resources increase ?
- Persistence: how can data outlast processes that created them
- Compatibility: can we ever do anything new ?
- Distribution: accessing the world of information
- Accounting: who pays bills, and how to control resource usage

Know your Student

20 minutes

1. What you thought you would be studying as a part of this course?
2. What are your perceptions of the course based on its name?
3. What you know about systems design and implementation?
4. What you know about Operating Systems?
5. Having known C programming, do you know how a compiler works?
6. Have you written assembly language programs? When, How?
7. Do you like problem solving? Why? Can a CS course of this type be made more interesting through mathematics? Give one example.