ICS 2305 SYSTEMS PROGRAMMING

Course description

System Programming overview : Intro and Definitions, Application Vs System Programming, System Software, Operating system, Device Drivers, OS Calls. Window System Programming for Intel386 Architecture: 16 bit Vs 32 bit, Programming, 32 bit Flat memory model , Windows Architecture. Virtual Machine (VM) Basics, System Virtual Machine, Portable Executable Format, Ring O Computer, Linear Executable format, Virtual Device, Device Driver Development. UNIX Unifying Ideas Compilation C Data types, Pointers, etc. Standard I/O, Dynamic memory management, System calls & error handling; Building, Debugging and Directories in Unix, Unix File System Structure; Linux File System Kernel Data Structures; Low-level I/O (Processes, File Descriptors, Opening/Creating/Closing/Deleting Files, Reading and Writing); Intro to Processes (Programming with Processes: IPC, Pipes, FIFO/Named Pipes, Message queues, Sockets, Semaphores and Locks, Signals; Abnormal Termination, Orphaned Processes). Threads, Network File Systems (CVS, NFS v2 spec). Subverting an interface: The SFS Server.

Prerequisites:

BIT 2203: Advanced Programming, ICS 2202: Computer Operating Systems

Course aims

This course aims at providing students a basic understanding of the issues involved in writing system programs, manipulating system processes, system IO, system permissions, files, directories, signals, threads, sockets, terminal, etc. The primary operating system discussed will be Unix (Linux) but Windows will also be discussed. A primary goal of the course then is to reinforce, in a systems programming context, the skills you have already acquired to develop code that is robust.

Learning outcomes

By the end of this course unit, the learner should be able to:

1. Identify the basic components of an operating system, describe their purpose, and explain how they function.

2. Write, compile, debug, and execute C programs that correctly use system interfaces provided by UNIX (or a UNIX-like operating system).
3. List UNIX system calls, and invoke them correctly from within C programs.
4. Write, compile, debug, and execute C programs that create, manage and terminate processes and threads on UNIX.
5. Write, compile, debug, and execute C programs with processes and threads that interact by invoking and catching signals.
6. 6. Write, compile, debug, and execute C programs that use files and I/O on UNIX.
7. Write, compile, debug, and execute C programs that make use of memory management functions.
8. Write distributed applications that communicate across a network.

Instruction methodology

- Lectures,
- Tutorials,
- Practical,
- Presentations,
- Discussions,
- Industrial visits.

Assessment information

The module will be assessed as follows;

- 40% Continuous Assessment (Tests 10%, Assignment 10%, Practical 20%)
- 60% End of Semester Examination.