Computer System Programming

1. **Course number and name:** 020PSYES5 Computer System Programming

2. **Credits and contact hours:** 4 ECTS credits, 35 contact hours (lectures + labs)

3. **Instructor's or course coordinator's name:** Maroun Chamoun

4. **Text book**:
    a. **Other supplemental materials:**
    Handouts posted on the Web.

5. **Specific course information**
    a. **Catalog description**:
    Program development and object code structure - UNIX introduction - UNIX file systems and low-level I/O - Signals and signal handlers - I/O redirection and pipes - Process management - System V IPC (Inter-process communication) and synchronization: semaphores, message queues, shared memory- Networking and Berkeley Socket Programming.

    b. **Prerequisites:** 020POOES1 Object-Oriented Programming, 020SSEES3 Operating Systems

    c. **Required**: Elective for CCE students; required for CCE software engineering option students

6. **Specific goals for the course**
    The goal of this course is to provide an in-depth introduction to a systems programming, system programming language(s) and application of those language(s) to systems level problems.

    a. **Specific outcomes of instruction:**
        - Become familiar with the basic tools used to develop software in C on the Unix platform.
        - Exhibit proficiency in the use of the C programming language to design and code systems-related programs.
        - Design and implement programs making direct use of operating system facilities to perform low-level file I/O and directory manipulations.
        - Use a modern OS API and the concepts of process creation, synchronization & communication to solve concurrent programming problems.
        - Become familiar with socket programming using the Berkeley socket API

b. KPI addressed by the course:

KPI	a2	e3	k2
Covered	x	x	x
Assessed	x	x	x

7. Topics and approximate lecture hours:

- Introduction: Fundamental concepts, Historical Introduction to UNIX, System calls and library functions, Error handling, System data types, Tools and Compilation: C compiler, compiling & linking, creating and using static libraries, creating and using shared libraries, introduction to make (3 lectures)
- Lab: Best practices in C programming (1:15 lab hours)
- File I/O: I/O system calls versus stdio, File descriptors, I/O system calls: open(), close(), read(), write(), Seeking to a file offset: lseek(); file holes, Atomicity and race conditions, Relationship between file descriptors and open files, Duplicating file descriptors, File control: fcntl(), Open file status flags, Other file I/O APIs, File attributes, Retrieving file information: stat(), Changing file attributes, Directories and links, Hard and soft (symbolic) links, Directories: Current working directory, System calls and library functions for working with directories and links, Working with pathnames, Scanning directories; walking directory trees (3 lectures)
- Lab: Files (1:15 lab hours)
- Processes: Process ID and Parent process ID, Command-line arguments, Environment list, Process groups, sessions, and job control, Process creation and termination: Process creation: fork(), File descriptors and fork(), Process termination: exit() and _exit(), Exit handlers, Monitoring child processes: wait(), waitpid(), waitid(), Executing programs: execve(), exec() library functions, File descriptors and exec() (3 lectures)
- Signals: Signal types and default actions, Setting signal dispositions, Signal handlers, Signal sets, Blocking signals (the signal mask); pending signals, Sending signals, Designing signal handlers (2 lectures)
- Lab: Processes and signals (1:15 lab hours)
- Pipes and FIFOs: Creating and using pipes, Using pipes to connect filters, FIFOs, Semantics of I/O on pipes and FIFOs (2 lectures)
- Lab: Communication using pipes (1:15 lab hours)
- System V IPC: Introduction to IPC, semaphores: Semaphore operations, synchronizing access to a shared resource, shared memory: using shared memory objects, synchronizing access to shared memory, message queues: Message queue attributes, sending and receiving messages (3 lectures)
- Lab: Communication using message queues (1:15 lab hours)
- Lab: Communication using shared memory (1:15 lab hours)
- Lab: synchronizing access to a shared resource using semaphores (1:15 lab hours)
- BSD sockets: Introduction to sockets, Socket types and domains, Sockets system calls, Stream sockets, Datagram sockets, UNIX Domain sockets: Stream and datagram sockets in the UNIX domain, Socket permissions, Creating a socket pair: socketpair(), Internet domain sockets: TCP/IP fundamentals, Protocols and layers: IP, UDP, and TCP, IP addresses, Port numbers, Internet socket addresses, Data representation issues,

Client-server example, Concurrent versus iterative server design, recv() and send() system calls, Socket options (3 lectures)

- Lab: Communication using Datagram sockets (1:15 lab hours)
- Lab: Communication using Stream sockets (1:15 lab hours)