Uttarakhand Technical University, Dehradun
New Scheme of Examination as per AICTE Flexible Curricula
Computer Science and Engineering, VIII-Semester
CS 801 Advanced Operating Systems

Course Objectives:

- Have an understanding of high-level OS kernel structure.
- Gained insight into hardware-software interactions for compute and I/O.
- Have practical skills in system tracing and performance analysis.
- Have been exposed to research ideas in system structure and behaviour.
- Have learned how to write systems-style performance evaluations.

- **Learning Outcomes:**
  1. Outline the potential benefits of distributed systems.
  2. Summarize the major security issues associated with distributed systems along with the range of techniques available for increasing system security.
  3. Apply standard design principles in the construction of these systems.
  4. Select appropriate approaches for building a range of distributed systems, including some that employ middleware

Course Content:

UNIT I:

Overview of UNIX system calls. The anatomy of a system call and x86 mechanisms for system call implementation. How the MMU/memory translation, segmentation, and hardware traps interact to create kernel–user context separation. What makes virtualization work? The kernel execution and programming context. Live debugging and tracing. Hardware and software support for debugging.

UNIT II:

DTrace: programming, implementation/design, internals. Kprobes and SysTrace: Linux catching up. Linking and loading. Executable and Linkable Format (ELF). Internals of linking and dynamic linking. Internals of effective spinlock implementations on x86. OpenSolaris adaptive mutexes: rationale and implementation optimization. Pre-emptive kernels. Effects of modern memory hierarchies and related optimizations.

UNIT III:

Process and thread kernel data structures, process table traversal, lookup, allocation and management of new structures, /proc internals, optimizations. Virtual File System and the layering of a file system call from API to driver. Object-orientation patterns in kernel code; a review of OO implementation generics (C++ vtables, etc).

UNIT IV:

OpenSolaris and Linux virtual memory and address space structures. Tying top-down and bottom-up object and memory page lookups with the actual x86 page translation and segmentation. How file operations, I/O buffering, and swapping all converged to using the same mechanism. Kmem and Vmem allocators. OO approach to memory allocation. Challenges of multiple CPUs and memory hierarchy. Security: integrity, isolation, mediation, auditing. From MULTICS and MLS to modern UNIX. SELinux type enforcement: design,

implementation, and pragmatics. Kernel hook systems and policies they enable. Trap systems and policies they enable. Tagged architectures and multi-level UNIX.

UNIT V:

ZFS overview. Open Solaris boot environments and snapshots. Open Solaris and UNIX System V system administration pragmatics: service startup, dependencies, management, system updates. Overview of the kernel network stack implementation. Path of a packet through a kernel. Berkeley Packet Filter architecture. Linux Netfilter architecture.

Topics for Programs:

1. Getting Started with Kernel Tracing - I/O
2. Kernel Implications of IPC
3. Micro-Architectural Implications of IPC
4. The TCP State Machine
5. TCP Latency and Bandwidth

Text Books:

1. Jean Bacon, Concurrent Systems, Addison – Wesley, 1998.
2. William Stallings, Operating Systems, Prentice Hall, 1995.
3. Andrew S. Tanenbaum and Maarten van Steen. "Distributed Systems: Principles andParadigms", Prentice Hall, 2nd Edition, 2007.
4. Silberschatz, Galvin, and Gagne, Operating System Concepts Essentials, 9th Edition.