

Operating Systems Principles

Learning Objectives

The reading, quizzes, lectures, exams, and labs for this course are all designed around a set of learning objectives. These learning objectives can be divided into a few basic categories:

- **Concept** ... You should be able to:
 - define them and discuss their implications.
 - discuss how they relate to one-another.
 - give examples, recognize instances.
- **Issue** ... You should be able to:
 - discuss the considerations to be weighed.
 - recognize situations where they do and do not apply.
 - interpret them in the context of a particular problem.
 - make and justify decisions based on them.
- **Approach** (mechanism, algorithm, or architecture) ... You should be able to:
 - describe it, and what problem it addresses.
 - describe its elements or steps.
 - explain its applicability and advantages relative to alternatives.
 - recognize and explain its limitations.
 - explain why it is or is not appropriate for a particular problem.
 - describe how it could be applied to a particular problem.
- **Skill** (with an API or operation) ... You should be able to:
 - explain how and why it should be used.
 - independently and correctly use it in practical problems.
 - review, describe, and critique code that uses it.
 - diagnose and correct common problems that occur when using it.

Lect/Lab	Subject	Concept	Issue	Approach	Skill
1	Introduction to Operating Systems	why study OS what is OS mechanism/policy separation interface vs implementation interface contracts modularity/information hiding powerful abstractions appropriate abstraction indirection/deferred binding cohesion opaque encapsulation <i>thinking data structures</i> <i>OS types/history</i>	why functionality implemented in OS OS goals	layered/hierarchical structure dynamic equilibrium	
2	Resources, Services, and Interfaces	basic OS services higher level OS services layers of services private resource APIs and ABIs service protocols upwards compatability objects and operations abstracted resources		subroutines vs system calls versioned interfaces	

		serially reusable resources partitioned resource sharable resource			
P0	LAB 0 - Linux/C programming				
3	Processes, Execution, and State	programs and processes process address space stacks/linkage conventions process state (elements of) process resources user-mode, supervisor-mode traps (exceptions) interrupts (async events) traps (syscalls) limited direct execution	fork vs exec syscalls vs procedure calls	process implementation copy-on-write context save/restore static libraries shared libraries dynamically loadable libraries	process operations signals (exceptions)
4	Scheduling Algorithms, Mechanisms and Merformance	execution state model metrics: completion time metrics: throughput metrics: response time <i>service level agreement</i> process state (model) non-preemptive scheduling time sharing time slice performance under load	scheduling goals starvation convoy cost of context switch optimal time slice <i>end-to-end performance</i> graceful degradation	First In First Out Shortest Job First priority scheduling real time scheduling preemptive scheduling round-robin multi-level feedback queues dynamic equilibrium mechanism/policy separation	
P1A	LAB 1A - Processes, terminal I/O and pipes				file APIs
5	Memory Management, Allocation and Relocation	physical address space virtual address space text segment data segment stack segment shared library segments protected memory sharing	memory management goals internal fragmentation external fragmentation <i>pool rebalancing</i> common errors	address space layout stack vs heap allocation variable size allocation malloc vs sbrk memory allocation life cycle coalescing <i>free list design</i> first fit <i>best fit</i> <i>worst fit</i> next fit pool/slab allocation <i>diagnostic free lists</i> garbage collection	
6	Virtual Memory and Paging	swap space paged address space page table entry page replacement Temporal Locality Spatial Locality Least Recently Used working set clean/dirty pages	relocation problem thrashing <i>paging and segmentation</i>	base/limit relocation <i>segment addressing</i> memory compaction swapping paging MMU translation look-aside buffer demand paging page fault process Belady's Optimal Algorighm LRU clock algorithm working set clock algorithm page stealing <i>copy on write</i> <i>proactive page laundering</i>	

P1B	LAB 1B - Compressed Communication				
7	Threads, Races, and Critical Sections	thread thread state IPC purpose non-deterministic execution race condition indeterminate results critical section	thread motivations IPC goals flow control	thread stacks user mode implementations stream IPC message IPC shared memory IPC synchronous vs asynchronous IPC mutual exclusion atomicity spinning	
8	Mutual Exclusion and Asynchronous Completions	parallelism scenarios locking & event completion	correct order locking goals correct mutual exclusion who to wake up sleep/wakeup races spurious wakeup	synchronous requests asynchronous completion types of locks interrupt disables spin locks asynchronous events waiting lists	
P4A	Embedded System Bring-up				
9	Higher Level Synchronization and Communication	Semaphores	bounded buffer problem producer/consumer problem costs of contention granularity <i>convoy</i>	binary semaphores using semaphores condition variables using condition variables semaphore implementation CV implementation contention reduction file level locking advisory locking	
P2A	LAB 2A - Races and Serialization				thread API recognize critical sections pthread_mut operations pthread_cond operations protect critica section
10	Deadlocks, Prevention, and Avoidance	Dining Philosophers problem necessary conditions livelock	reasonable spinning deadlock common synchronization errors	compare and swap <i>monitors</i> <i>java synchronization</i> where to serialize avoidance reservations prevention detection and recovery	
11	Performance	performance metrics	goals and challenges performance principles typical causes of performance problems common measurement errors	load generation traces/logs internal instrumentation end-to-end measurement	load characterization analysis techniques presentation techniques
P2B	LAB 2B - Contention				malloc APIs profiling
12	Device I/O and	I/O bus	importance of disks	polled I/O	

	Drivers	Device Controller disk geometry	disk performance random vs sequential I/O transfer size	DMA completion interrupts initiating I/O operations write-back cache write-through cache <i>chain scheduling</i> buffered I/O <i>scatter/gather</i> <i>intelligent I/O devices</i> read/write striping write mirroring parity/erasure coding asynchronous parallel I/O polling for asynchronous completions non-blocking I/O asynchronous event notifications	
13	File Semantics and Representation	file semantics file types and attributes data base semantics <i>object semantics</i> <i>key-value semantics</i> file namespace file name conventions copies vs links symbolic vs hard links <i>storage volume</i> disk partitioning	read after write consistency goals of file representation goals of free space representation goals of namespace representation	BSD file system organization FAT file system organization indexed data extents (I-nodes) linked data extents (FAT) <i>linked free lists (Unix V5)</i> contiguous allocation bit-map free lists (BSD) <i>FAT free list</i> BSD directory entries FAT file descriptors the mount operation file access layers of abstraction <i>dynamically loadable file systems</i> <i>user mode file systems</i>	
P4B	LAB 4B - Embedded System Sensors				
14	File Systems Performance and Robustness		disks and file system design block size & internal fragmentation causes of file system damage	read cache write-through general/special purpose caches <i>how to beat LRU</i> delayed writes write-back cache detection and repair journaling and recovery meta-data-journaling copy-on-write file systems log structured file systems checksums and scrubbing	
15	Security, Protection, Authentication, Authorization	confidentiality integrity controlled sharing object, agent, principal mediated access revocability trusted computing platform principles of secure design authentication authorization access control lists Linux file protection (ACLS) capabilities unforgeability	challenges <i>key security</i>	cryptographic hashes challenge/response authentication Linux principals Linux authentication Linux setuid at rest encryption symmetric encryption cryptographic privacy	

		Linux file descriptors (capabilities) principle of least privilege Role Based Access Control Trojan horses cryptographic hash			
P3A	LAB 3A - File System Interpretation				
16	Distributed systems: Goals, Challenges and approaches	goals RPC RPC stub RPC skeleton RESTful interface principles Marshal/unmarshal unforgeable capabilities	challenges Deutsch's 7 Falacies RPC interoperability	RPC tool chain XDR asymmetric encryption digital signatures	
17	Remote Data, Synchronization, Security	local vs cloud client/server model <i>distributed file systems</i>	man in the middle attacks remote file system goals	reliable communication distributed locks leases distributed transactions distributed consensus <i>two phase commit</i> <i>three phase commit</i> remote file transfer remote file access peer-to-peer security work tickets distributed authentication/authorization	
P3B	LAB 3B - File System Analysis				
18	Remote Data Performance and Robustness	immutability reliability availability stateless server protocols idempotent operations ACID Consistency Read-After-Write Consistency Close-Open Consistency classes of distributed systems	<i>graceful degradation</i> write latency cache consistency APIs -> protocols	replication and recovery distributed data striping direct client-server communication <i>scalable distributed systems</i> client-side caching	
19	MP and Distributed Systems	Single System Image Non-Uniform Memory Architecture <i>loosely coupled</i> <i>tightly coupled</i> <i>unit of deployment</i> <i>unit of failure/replacement</i> cluster membership Geographic Disaster Recovery availability zones Software Defined Networking Software Defined Storage Eventual Consistency <i>BASE semantics</i> <i>Data plane</i> <i>Control plane</i> <i>immutable objects</i>	<i>CAP Theorem</i>	Symmetric Multi-Processor Horizontally Scaled Clustered <i>heart beating</i> Active/Active Active/Standby partitioned responsibility <i>WAN-scale consistency models</i>	

P4C	LAB 4C - IOT Security				
-----	-----------------------	--	--	--	--

Last updated: 4/23/2018