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.
  - descibe its elements or steps.
  - explain its applicability and advantages relative to alternatives.
  - recognize and explain its limitations.
  - explain why it 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.
  - independenly 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.

Subject	Concept	Issue	Approach

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** *OS types/history*	why functionality implmented in OS OS goals	layered/ hierarchical structure **dynamic equilibrium**
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** **serially reusable resources** **partitioned resource** **sharable resource**		subroutines vs system calls **versioned interfaces**
LAB 0 - Linux/C programing			

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	**static and dynamic libraries** process implementation copy-on-write context save/ restore **static libraries** **shared libraries** **dynamically loadable libraries**
Scheduling Algorithms, Mechanisms and Merformance	**execution state model** **metrics: completion time** **metrics: throughput** **metrics: response time** **service level agreement** **process state (model)** **non-preemptive scheduling** **time sharing** **time slice** performance under load	**scheduling goals** **starvation** **convoy** **cost of context switch** optimal time slice end-to-end performance graceful degradation	First In First Out Shortest Job First **priority scheduling** real time **preemptive scheduling** **round-robin** multi-level queues **dynamic equilibrium** **mechanism/ policy separation**
LAB 1A - Processes, terminal I/O and pipes			

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** *pool rebalancing* **common errors**	**address space layout** **stack vs heap allocation** **variable size allocation** **malloc vs sbrk** memory allocation life cycle **coalescing** *free list design* **first fit** **best fit** **worst fit** **next fit** pool/slab allocation diagnostic free lists **garbage collection**
Virtual Memory and Paging	**swap space** **paged address space** **replacement** **Temporal Locality** **Spatial Locality** **Least Recently Used** **working set** **clean/dirty pages**	relocation problem thrashing *paging and segmentation*	base/limit relocation *segment addressing* memory compaction **swapping** **paging MMU** **demand paging** **page fault process** **Belady's Optimal Algorigthm** LRU clock algorithm working set clock algorithm page stealing copy on write *proactive page laundering* *scatter/gather*

LAB 1B - Encrypted Communication			
Threads, Races, and Critical Sections	**thread** thread state **IPC purpose** **non-deterministic execution** **race condition** indeterminate results **critical section**	**thread motivations** **IPC goals**	thread stacks user mode implementations **stream IPC** **message IPC** **shared memory IPC** **mutual exclusion** **atomicity** **spinning**
Mutual Exclusion and Asynchronous Completions	**parallelism scenarios** locking & event completion	correct order **locking goals** **correct mutual exclusion** who to wake up sleep/wakeup races spurrious wakeup	**synchronous requests** **asynchronous completion** **types of locks** interrupt disables spin locks **asynchronous events** waiting lists
Edison Bring-up			
Higher Level Synchronization and Communication	**Semaphores**	bounded buffer problem producer/ consumer problem **costs of contention** **granularity** convoy	**binary semaphores** using semaphores **condition variables** using condition variables semaphore implementation CV implementation **contention reduction** file level locking **advisory locking**
LAB 2A - Races and Serialization			

Deadlocks, Prevention, and Avoidance	Dining Philosophers problem **necessary conditions** livelock	reasonable spinning **deadlock** common errors	compare and swap *monitors* *java synchronization* **where to serialize** **avoidance** **reservations** **prevention** detection and recovery
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
LAB 2B - Contention			

Device I/O and Drivers	*I/O bus* *Device Controller* disk geometry	importance of disks disk performance **random vs sequential I/O** transfer size	polled I/O DMA completion interrupts initiating I/O operations write-back cache write-through cache *chain scheduling* buffered I/O *scatter/gather* *intelligent I/O devices* read/write striping write mirroring parity/erasure coding asynchronous parallel I/O polling for asynchronous completions non-blocking I/O asynchronous event notifications

File Semantics and Representation	**file semantics** **file types and attributes** *data base semantics* *object semantics* *key-value semantics* **file namespace** file name conventions copies vs links symbolic vs hard links *storage volume* disk partitioning	read after write consistancy 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) linked free lists (Unix V5) contiguous allocation bit-map free lists (BSD) *FAT free list* BSD directory entries FAT file descriptors the mount operation file access layers of abstratction *dynamically loadable file systems* *user mode file systems*
LAB 4B - Embedded System Sensors			

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 *how to beat LRU* **delayed writes** **write-back cache** detection and repair journaling and recovery meta-data-journaling copy-on-write *log structured file systems* **checksums and scrubbing** defrragmentation

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 Linux file descriptors (capabilties) **principle of least privilege** Role Based Access Control **Trojan horses** cryptographic hash	challenges *key security*	cryptographic hashes challenge/ response authentication Linux principals Linux authentication Linux setuid **at rest encryption** **symmetric encryption** cryptographic privacy
LAB 3A - File System Interpretation			

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 secure sessions
Remote Data, Synchronization, Security	**local vs cloud client/server model** *distributed file systems*	**remote file system goals**	distributed locks leases distributed transactions **remote file transfer** **remote file access** peer-to-peer security distributed authentication/ authorization
LAB 3B - File System Analysis			
Remote Data Performance and Robustness	**immutability** **reliability** **availability** **stateless server protocols** **idempotent operations** **ACID** **Consistency** **Read-After-Write Consistency** **Close-Open Consistency**	**man in the middle attacks** *graceful degradation* write latency cache consistency	**reliable communication** distributed consensus **replication and recovery** **striping** direct client-server communication *scalable distributed systems* **client-side caching**

MP and Distributed Systems	**classes of distributed systems** **Single System Image** Non-Uniform Memory Architecture *loosely coupled* *tightly copuled* *unit of deployment* *unit of failure/ replacement* cluster membership **Geographic Disaster Recovery** **availability zones** **Software Defined Networking** **Software Defined Storage** Eventual Consistency *BASE semantics* *Data plane* *Control plane* *immutable objects*	APIs -> protocols	**Symmetric Multi-Processor** **Horizontally Scaled** Clustered *heart beating* Active/Active Active/Standby partitioned responsibility *WAN-scale consistency models*
LAB 4C - IOT Security			

Last updated: 5/8/2017