

UNIT 1 -- An Introduction to the Operating System	
What is an operating system?	1
Services Provided by the Operating System	3
Illusions	3
Virtual Processors	
Multi-tasking	
Time-slicing	
Virtual memory	
Memory Management Unit (MMU)	
Page fault	
Process	
User Time	
System Time	
Wall Time	
The System Call	5
UNIX Software component layers	5
About man pages	6
The UNIX I/O model	7
Open	
Close	
File Descriptor	
Standard I/O streams	8
stdin(0)	
stdout(1)	
stderr(2)	
Error reporting	9
Global int errno	
Reading and Writing	10
Read	
Write	
Just a bunch of bytes	11
O_APPEND	
The stdio library	12
Buffering and the stdio library	12
Output buffering modes	
UNIT 2 -- An Introduction to the File Systems	
The File System	1
Volume	
What the File System provides	2
Basic Services to the User	
Naming	
Data Storage	
Meta-data	
Free space management	
Flat file systems	2
Metadata	
UNIX Filesystems	3
How UNIX organizes a volume	
Superblock	
Inode	
Inode number	
Metadata	
Inode type	
Directories and inodes	5
Pointer	
Inode Table	
Data Block Area	
Pathnames and Wildcards	7
Absolute paths	
Fully-qualified paths	
Relative paths	
Current working directory	
The NUL terminator (\0) character	
Unbounded number of possible pathnames which refer to the same node.	
Hard Links	7
Link and Unlink	
Directories and link counts	12
Mode / Permissions mask	
Reading directories	13
Symbolic Links	14
Symlink	

Readlink	
Lstat	
Volumes	
The stat system call	17
Metadata fields	
ST_MODE field	
Sticky Bit	
Set-gid bit	
Set-uid bit	
Inode Types	19
The UNIX file permissions model	20
User id	
Group id	
Process	
Superuser or Root	
Access Control Lists	
Mounted Volumes	22
Volume	
Mounted	
Root Filesystem	
Mount point	
Virtual Filesystem	
Pseudo-filesystem	
Network filesystems	
Device number	
Is that filesystem, or file system, or filesystem?	27
A review of filesystem-related system calls seen thus far	27
IMPORTANT	
UNIT 3 -- Processes: fork,exec,exit,wait. I/O redirection	
The UNIX Process	1
Virtual computer	
Process ID (pid)	
Process State	1
The Virtual Address Space of a Process	1
Regions AKA Segments	
Installing a new program with exec	3
Exec(2)	
Environment	
A.out contains...	
Exec system call	4
More in-depth look at exec(2)	
Executing via an interpreter	5
Interpreted script	
The Environment	5
Opaque data	
Environment	
Starting a new process with fork	6
Parent Process	
Child Process	
Parent Process id (ppid) of child	
Fork causes a function to return twice	
Clone	7
Threads	
The file table and file descriptors	7
The struct file	
f_mode	
f_flags	
f_count	
f_pos (cursor)	
close on a fd NULLs out that fd for the calling process ONLY	
Dup and I/O redirection	9
Dup to duplicate fds	
fork and the file descriptor table	10
Typical shell I/O redirection	10
Expected file descriptor environment	11
Possible errors...	
Process termination	11
Signals	
Return code	
Zombie	
User CPU time	

System CPU time	
Total Amount of CPU time	
Real Time	
wait, wait3, waitpid	
Typical fork/exec flow cycle	15
 UNIT 4 -- Signals and Pipes	
Signals	1
Synchronous signals	
Asynchronous signals	
Sending a signal (generating or posting)	
Delivery	
kill(2)	
kill system call	1
Pre-defined UNIX signal numbers and behaviors	2
Pending signals and signal masks	3
Pending signals array	
Signal mask	
Signal delivery and disposition	3
Desired outcomes of a process on a per-signal-number basis	
Terminate the process	
Coredump	
Ignore	
Stop	
Job control	
Resume	
Handle	
Establishing signal disposition	4
Signal()	
void *handler(int)	
The signal handler	4
setjmp/longjmp	5
Non-local goto	
Masking/Blocking signals	6
Critical regions	
sigprocmask()	
The sigaction interface	8
sa_flags	
sa_mask	
Real-time signals	9
SIGHUP	
real-time signals	
Interrupted System Calls	9
SIGCHLD	10
Signal interactions with fork and exec	10
During a successful exec system call...	
Pipes	11
pipe	
Read-side Write-side	
Pipe properties	11
Flow Control	12
Atomicity	12
Atomically	
Connection Control	12
Broken pipe	
Named Pipes	13
Named pipe (IS_FIFO)	
 UNIT 5 -- Virtual Memory	
Virtual Memory	1
Virtual Addresses	
Virtual addresses translated to physical addresses	
Translation is hardware within MMU	
Virtual Page number	
Physical Page number	
Present bit	
Page Fault	
Physical Page Number	
Protection Bits	
Protection fault	
Dirty Bit	

Accessed bit	
Privileged Bit	
Multi-Level Page Tables	3
Multi-level page tables	
Page Table Entry (PTE)	
Context Switch	
Translations Cache	5
Address Translation Cache (ATC)	
Translation Lookaside Buffer (TLB)	
Key, Value, Hit, Cache miss	
Context Number	
Address Space ID	
The UNIX user-level memory model	7
Break Address	
Demand-Paging	7
Page fault interrupt	
Demand-Paging	
Paging-out and Backing Store	8
Paging-Out	
Backing Store	
Paging-in, minor and major faults	9
Minor Page Fault	
Major Page Fault	
Page / Protection Faults and signals	9
Mmap	10
SIGBUS	
Exec and virtual memory	12
magic number	
Why the MAP_DENYWRITE flag?	13
Fork and copy-on-write	14
Copy On Write (COW)	
Appendix: How an executable file is created	16
 UNIT 7 -- Concurrency and Synchronization	
Synchronization among Concurrent tasks	1
Shared Update Problem: A Simple Race Condition	2
Atomicity	4
Critical Regions and Mutual Exclusion	4
Implementing MUTEX by Disabling Interrupts or Signals	4
Spin Lock Mutex	5
Atomic Instructions	6
Code-based or Data-based?	7
Mutex locking with blocking/yielding	8
The deadlock problem	9
Deadlock Avoidance	9
Deadlock Detection	10
Live Lock	10
Fine vs Coarse Grain Locking	10
Read/write locking	11
Example of reader/writer locks	13
The problem of inconsistent views	13
Seqlocks	14
Semaphores	16
Condition Variables	16
Condition Variable Operations	17
Producer/Consumer problem using condition Variables	18
Some Golden Rules of Condition Variables	20
 UNIT 8 -- Introduction to the Kernel	
What is the Kernel?	1
Description of directories	
Boot-up	2
Process #1 /sbin/init running as uid=0 gid=0	
Re-entering the kernel	4
Synchronous re-entry	
Events Exceptions	
Asynchronous re-entry	
Top half/Bottom half	
Exception Types	4
Fault	
Page Fault	
System Call	

Interrupt Types	5
Periodic Interval Timer (PIT)	
Inter-processor Interrupt (IPI)	
Important Shared Processor/Kernel Data Structures	5
Interrupt Descriptor Table (IDT)	
Task State Segment (TSS)	
Global Descriptor Table (GDT)	
The special tr register	
The cr3 register	
Interrupt and Exception handling in hardware	6
Associated vector between 0 and 255	
The handler address is the virtual address of the first opcode of the associated handler function within the kernel	
Vectors 0-31 are reserved for processor-generated exceptions	
The five critical registers	
1. the old stack pointer register %esp	
2. the flags/status register %eflags	
3. the program counter register %eip	
4&5. %cs and %ss registers (how code & stack memory accessed)	
The special iret instruction	
Kernel control flow paths	8
What the CPU is doing at any given moment	
Context switch	
Making a System Call	9
The 32-bit X86 API	9
Pass arguments to system calls in registers	
First argument is %ebx	
Second is %ecx	
Three - Six are %edx, %esi, %edi, %ebp	
System call number is passed to the kernel in %eax	
Kernel-mode stack and the thread_info structure	12
The X86-64 API	13
Now, back to our system call, already in progress	14
System Call Return Value	15
Kernel system call hand-off to C	15
Returning from a system call	17
Deferred return from system call	17
Appendix - X86 Architecture and Assembly Language	18
Intel vs UNIX assembly syntax	18
Assembler Directives / Pseudo Opcodes	19
X86 Register Model	19
Segmentation, Special-Purpose and Additional Registers	20
Addressing Modes	20
Function Calling Convention	21
X86-64 Function Calling	23
X86-64 Global Variables	24
Caller/Callee saves	24
X86 General-Purpose Register Summary	25
UNIT 9 -- Task Switch / Sleep and Wakeup	
A Task Switch	1
Context	
Preemptive and Cooperative Multitasking	2
Cooperative Multitasking	
Pre-emptive multitasking	
Periodic Interval Timer interrupt	
Task	
Process/task states	3
States	
State diagram for task state transitions	
INTERRUPTIBLE and NON-INTERRUPTIBLE sleeps	
ZOMBIE task	
Data structures for tracking processes / tasks in Linux kernel	4
struct thread_info	
struct task_struct	
current	
The struct task_struct	4
Sidebar: linked lists	6
Sleep and wakeup	7
Wait Queues	8
Exclusive vs non-exclusive waiting	9
Thundering Herd	

Wait interrupted by signal	10
Wait queue example	11
Waking Up	12
A system call with blocking	14
Sleep/Wakeup Summary	19
The schedule() function	19
Task switch	
Run Queue	
Directly	
Indirectly(lazy)	
An Asynchronous routine must never call schedule directly	
Kernel Mode Pre-emption	20
Kernel pre-emption	
What schedule() does	20
Process and Thread Creation	25
Multiprocessor Considerations	32
Processos Termination	33
 UNIT 10 -- The Kernel's Scheduler	
Scheduling	1
The UNIX model of priority	1
Real-Time Tasks	2
Quantum	2
The Old Linux Scheduler vs the CFS scheduler	2
Ideal and practical latency	3
Weighted timeslice	3
Virtual Runtime	4
Scheduler interactions with fork	5
The Scheduler Tick	5
Interactive Performance	7
Overview of the old Linux scheduler	8
Run Queues	8
Priority formula	10
Quantum	10
Active and Expired	11
Idle	11
scheduler_tick	11
Inserting a task into the run queue at wakeup	12
Scheduler interactions with fork	12
 UNIT 11 -- Signals (Kernel Implementation)	
Signals	1
Signal data structures	1
Generating a signal	3
Following a kill	3
Signal Delivery	6
Setting up the stack frame to handle signal	10
sigsuspend	12
Restarting system calls	12
 UNIT 12 -- Kernel Memory Management	
The Memory Management System	1
The Page Frame Pool	1
Addressing Issues on X86	1
Page Descriptor Table	2
Buddy System	3
Slab Allocator	4
Managing Virtual Address Space	5
Virtual Addressing on the X86	5
Representing the process address space	7
Address space release (munmap)	10
Copy-On-Write and Fork	11
Page Fault Handling	11
Stacks	15
Invalid Addresses and SIGSEGV	15
Invalid addresses in Kernel Mode / The fixup table	16
Benign Page Faults / Paging-in	16
Paging In Logic	17
On-demand creation of Page Tables	19
Paging-in from where?	20
Copy-On-Write (COW)	21
Page Frame Reclamation	22

Reverse Mapping	24
Anonymous Mappings	24
Swap Space and anonymous page-outs	25
File Mappings	26
Resources Exhaustion	29
Thrashing / Swap Token	29
Memory over-commit and the out-of-memory killer	29