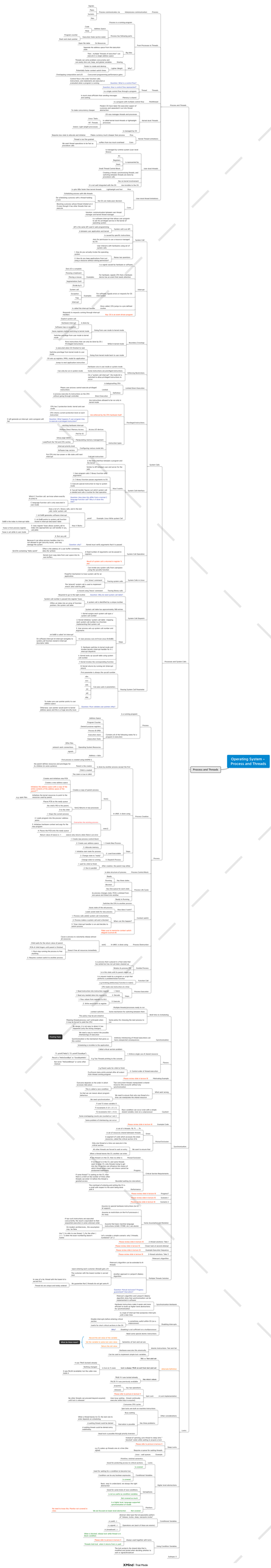
Operating System – Process and Threads
Process and Threads
Process and Threads
Process
Signals
Pipes
Sockets
files
Process communication via
Interprocess communication
Process
Process in a coming program
Code
Heap
Address Space
Program counter
Stack and stack pointer
Process has following parts
Execution State (active state)
Open file table
Fs Resources
From Processes to Threads
Separate fs address space from file descriptor
Threads share address space
Fast, multiple "threads of execution" can execute in a single address space
Big data
Threads can write problem concurrently and
Sharing
Threads share open files and other resources
Easier to create and destroy
Lighter weight
Why?
Potentially faster context switch times
Overlapping computation and I/O
Concurrent programming performance gain
Contract flow is the order function calls, returns and other control transfers are made or evaluated when a program is running
Question: What is a control flow?
Question: How is control flow represented?
Thread
Threads
Is a single control flow through a program, represented by a stack
Is stack more efficient than sending messages and sorting
Memory is shared
Pthreads
Pthreads-OS-like lower-like executing aspect of pthreads and exposed to run-on-top-of-pthreads
Process and Threads
To make concurrency cheaper
OS kicks manage threads and processes.
Linux: Tasks
NT: Threads
Solaris: Light-weight processes
Is implemented in user-level threads or lightweight processes
Is managed by OS
Pthreads
Pthreads-contexts much cheaper than process
Pros
Kernel Thread
Kernel Thread Limitations
Requires less state to allocate and initialize
Thread is less fat than process
Cons
Suffers from too much overhead
Is managed by runtime system (user-level library)
Is represented by
User level threads
User level threads
User-level thread
Registers
Stack
Heap
Small Thread Control Block
Creating a thread, synchronizing threads, and scheduling thread execution are done in procedure calls
Has no kernel involvement
Is run well integrated with the OS
Are invisible to the OS
Pros
Is only little faster than kernel threads
Lightweight and fast
User-level thread limitations
Scheduling process with one threads
Blocking system call blocks all threads
Cons
Be scheduling a process with a thread holding a lock
Pre-OS can make poor decision
Kernel-level threads
Kernel-level thread manage and kernel thread manager
Kernel-level thread
Is a software interrupt driven user program
Is a software interrupt driven kernel operating system
API is the same off-card or web programming
System call is an API
Is between user application and kernel
Is called by specific instructions
Asks for permission to run a resource managed by kernel
System Call
Use directly with hardware using all of system calls.
1. How do you actually invoke the operating system
2. How do you keep applications from just using a resource without asking permission?
Ribbon has questions
Is a signal raised by hardware or software
Divd 10 is complete
Preceding a keyboard
Pending a mouse
Network packet
Hardware Interrupt
Exceptions
Trap
Fault
Abort
Interrupt
Hardware signal
Kernel fault
Software trap
Exceptions
Fault
Trap
Abort
Interrupt
Fix software signals errors or requests for OS intervention
Fix software
Fix software
Linux called CPU jumps to a pre-defined handler
Is called the interrupt handler
Responds to requests coming from interrupt controller
Yes, it is an event driven program
Yes, it is an event driven program
Exception handler
Exception handler
Interrupt handler
Interrupt handler
Interrupt