L9e Virtual Machine Intro

① Benefits of Virtualization
- Multiple Secure Environment

A system VM provides a sandbox that isolates one system environment from other environments.
- Failure Isolation

Virtualization helps isolate the effects of a failure to the VM where the failure occurred.
- Better System Utilization

A virtualized system can be (dynamically or statically) reconfigured for changing needs.
- Mixed-OS Environment

A single hardware platform can support multiple operating systems concurrently.

② Virtualization

A virtualized system (or subsystem) is a mapping of its interface, and all resources visible through that interface, to the interface and resources of a real system. Virtualization involves the construction of an isomorphism that maps a virtual guest system to a real host system.

③ Virtualization Properties
- Isolation: Fault Isolation, Software Isolation, Performance Isolation.
- Encapsulation: All VM state can be captured into a file, Complexity is proportional to virtual HW model and independent of guest.
- Interposition: All guest actions go through the virtualizing software which can inspect, modify, and deny operations.

④ Virtual Machine Monitor

A VM is implemented by adding a layer of software to a real machine so as to support the desired VM's architecture. This layer of software is often referred to as virtual machine monitor (VMM). VMMs are often implemented as a co-designed firmware-software layer, referred to as the hypervisor.

⑤ Types of Virtual Machines
- Process VM: Capable of supporting an individual process.
- System VM: Provides a complete system environment. Supports an OS with potentially many types of processes.

⑥ Partitioning

Using partitioning, multiple applications can simultaneously exploit the available resources of the system. in-space: physical partitioning. in-time: logical partitioning.

⑦ Virtualizing Devices

The technique that is used to virtualize an I/O device depends on whether the device is shared and, if so, the ways in which it can be shared. The common categories of devices are: Dedicated devices, Partitioned devices, Shared devices, Spooled devices.

L10 Memory Management

① Address Space

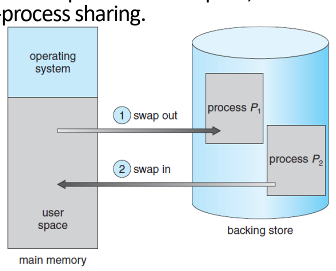
Definition: Set of accessible addresses and the state associated with them.

② Important Aspects of Memory Multiplexing

Protection, Controlled overlap, Translation.

③ Base and Bound

Translation happens at execution. Issues: Fragmentation problem over time, Missing support for sparse address space, Hard to do inter-process sharing.



Multiple separate segments: each segment is given region of contiguous memory.

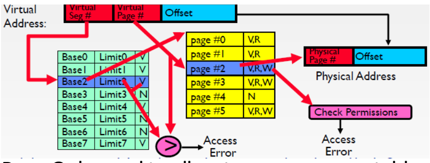
Fragmentation: wasted space
- External: free gaps between allocated chunks
- Internal: don't need all memory within allocated chunks

④ Page Sharing
- The "kernel region" of every process has the same page table entries.
- Different processes running same binary
- User-level system libraries (execute only)
- Shared-memory segments between different processes

⑤ Security Measures
- Address Space Randomization
- Kernel address space isolation

Context switch
- Needs to be switched: Page table pointer and limit.
- Protection: Translation (per process) and dual-mode.

⑥ Multi-Level Translation



Pros: Only need to allocate as many page table entries as we need for application, Easy memory allocation, Easy Sharing.

Cons: One pointer per page, Page tables need to be contiguous, Two lookups per reference.

L10e I/O Intro

① Linux: | pipe

② POSIX I/O

POSIX: Portable Operating System Interface

Design Patterns/Concepts: Uniformity - everything is a file, Open before use, Byte-oriented, Kernel buffered reads, Kernel buffered writes, Explicit close.

③ File System Abstraction

Files live in hierarchical namespace of filenames.

④ C High-Level File API – Streams.

Three predefined streams: stdin, stdout, stderr.

⑤ C Low Level: Standard Descriptors

int fsync (int filedes) – wait for i/o to finish
void sync (void) – wait for ALL to finish
int pipe(int fileds[2]);
// Writes to fileds[1] read from fileds[0]
int dup2(int old, int new);
int dup(int old);

Operations: open, read, write, create, close

Operations specific to terminals, devices, networking; Duplicating descriptors; Pipes: bi-directional channel; File Locking; Memory-mapping files; Asynchronous I/O.

⑥ Streams vs. File Descriptors

Stream are buffered in user memory (sleep won't work). Operations on file descriptors are visible immediately (sleep will work).

Why Buffer in User-space? Avoid system call overhead; Functionality, Simplifies operating system.

⑦ Device Drivers

Device-specific code in the kernel that interacts directly with the device hardware.

Typically divided into two pieces: Top half: accessed in call path from system calls; Bottom half: run as interrupt routine.

⑧ File abstraction works for inter-processes communication (local or Internet)

L11 Caching, TLB

① Page Table Discussion

Pros: Simple memory allocation, Easy to share

Cons: Too big table.

② Page Table Entry (PTE)

Pointer to next-level page table or to actual page + Permission bits: valid, read-only, read-write, write only.

How to use:
- Demand Paging: Keep only active pages in memory, Place others on disk and mark their PTEs invalid.
- Copy on Write: UNIX fork gives copy of parent address space to child, Address spaces disconnected after child created. To do this cheaply, make copy of parent's page tables (point at same memory); mark entries in both sets of page tables as read-only; page fault on write creates two copies.
- Zero Fill On Demand: New data pages must carry no information (say be zeroed). Mark PTEs as invalid; page fault on use gets zeroed page.

③ How are segments used

One set of global segments (GDT) for everyone, different set of local segments (LDT) for every process.

④ Inverted Page Table (P29)

Forward Page Tables: Size of page table is at least as large as amount of virtual memory allocated to processes.

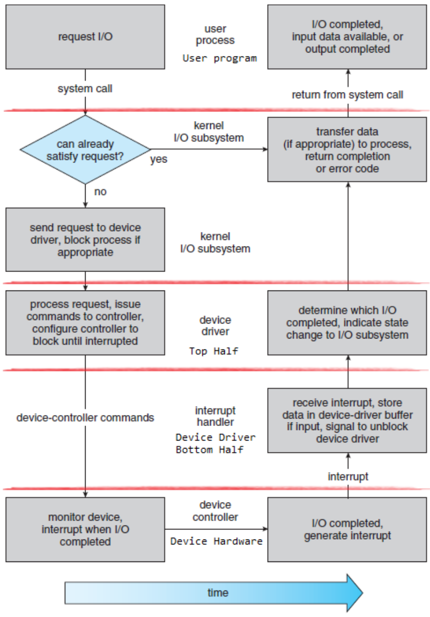
Inverted Page Table: Size directly related to amount of physical memory. Cons: Complexity of managing hash chains: Often in hardware! Poor cache locality of page table.

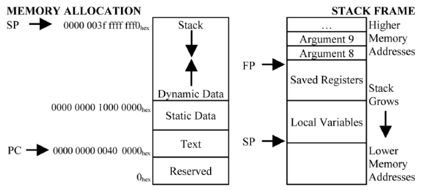
⑤ Comparison
- Simple Segmentation. Pros: Fast context switching: Segment mapping maintained by CPU. Cons: External fragmentation;
- Paging (single-level page). Pros: No external fragmentation, fast easy allocation. Cons: Large table size ~ virtual memory, Internal fragmentation.
- Paged segmentation & Two-level pages. Pros: Table size ~ # of pages in virtual memory, fast easy allocation. Cons: Multiple memory references per page access.
- Inverted Table. Pros: Table size ~ # of pages in physical memory. Cons: Hash function more complex, No cache locality of page table.

⑥ Memory Management Unit (MMU)

The processor requests READ Virtual-Address to memory system, through the MMU to the cache (to the memory).

On every reference (I-fetch, Load, Store) MMU read (multiple levels of) page table entries to get physical frame or FAULT.





SIZE PREFIXES AND SYMBOLS

SIZE	PREFIX	SYMBOL	SIZE	PREFIX	SYMBOL
10^3	Kilo-	K	2^{10}	Kibi-	Ki
10^6	Mega-	M	2^{20}	Mebi-	Mi
10^9	Giga-	G	2^{30}	Gibi-	Gi
10^{12}	Tera-	T	2^{40}	Tebi-	Ti
10^{15}	Peta-	P	2^{50}	Pebi-	Pi
10^{18}	Exa-	E	2^{60}	Exbi-	Ei
10^{21}	Zetta-	Z	2^{70}	Zebi-	Zi
10^{24}	Yotta-	Y	2^{80}	Yobi-	Yi
10^{-3}	milli-	m	10^{-15}	femto-	f
10^{-6}	micro-	μ	10^{-18}	atto-	a
10^{-9}	nano-	n	10^{-21}	zepto-	z
10^{-12}	pico-	p	10^{-24}	yocto-	y

⑦ Cache (P61)
A repository for copies that can be accessed more quickly than the original.
Average Memory Access Time (AMAT)
= (Hit Rate x Hit Time) + (Miss Rate x Miss Time)
Temporal Locality, Spatial Locality.
Misses: Compulsory, Capacity, Conflict, Coherence.
Organizations: Direct Mapped: single block per set; Set associative: more than one block per set; Fully associative: all entries equivalent.
Cache line: Tag + Index + Block (offset)
Write through: The information is written to both the block in the cache and to the block in the lower-level memory; Write back: The information is written only to the block in the Cache.
⑧ Translation Look-Aside Buffer (TLB)
Placed before cache.
Thrashing: continuous conflicts between accesses.
Conflict misses expensive: Fully Associative.
When Context Switch: Invalidate TLB, or Include ProcessID in TLB.
TLB Consistency: If translation tables change (evict and swap), must invalidate TLB entry.
⑨ Page Fault
Not an interrupt, synchronous to instruction execution (Trap).

L12 Demand Paging, General I/O
① Uses of Demand Paging
Extend the stack, Extend the heap, Process Fork, Exec, MMAP to explicitly share region.
② Cache size needs to fit working set size.
③ Effective Access Time
EAT = Hit Rate x Hit Time + Miss Rate x Miss Time = Hit Time + Miss Rate x Miss Penalty
④ Misses in Page Cache
Compulsory Misses, Capacity Misses, Conflict Misses, Policy Misses (bad eviction policy).
⑤ Thrashing
A process is busy swapping pages in and out with little or no actual progress (low CPU utilization).
⑥ I/O Subsystem
Devices have many different speeds.
Goal: Provide Uniform Interfaces, Despite Wide Range of Different Devices.
Devices: Block Devices, Character Devices, Network Devices.
Timing: Blocking Interface: “Wait”; Non-blocking Interface: “Don’t Wait”; Asynchronous Interface: “Tell Me Later”.
⑦ Transferring Data to/from Controller
- Programmed I/O: Each byte transferred via processor in/out or load/store. Pro: Simple hardware, easy to program, Con: Consumes processor cycles proportional to data size.
- Direct Memory Access: Give controller access to memory bus. Ask it to transfer data blocks to/from memory directly.
⑧ I/O Device Notifying the OS
- I/O Interrupt: Device generates an interrupt whenever it needs service. Pro: handles unpredictable events well. Con: interrupts relatively high overhead.
- Polling: OS periodically checks a device-specific status register. Pro: low overhead. Con: may waste many cycles on polling if infrequent or unpredictable I/O operations.
- Actual devices combine both polling and interrupts

L13 File Systems
① Logical Directory Organization
Goals: Efficiency - locating a file quickly; Naming - convenient to users; Grouping
② Hard links, soft links (P19)
Links (hard links) make directory a DAG, not just a tree. Softlinks (aliases) are another name for an entry.
③ File: permanent storage
Contains: Data (Blocks on disk), Metadata (Attributes - Owner, size, last opened, … Access rights (R, W, X, Owner, Group, Other, …)).

④ File Allocation Table (FAT) (P70)
In FAT, directory is a file containing <file_name: file_number> mappings. File attributes are kept in directory. Each directory is a linked list of entries. Root directory is at block 2 (no 0 or 1).
Security Holes: FAT has no access rights; FAT has no header in the file blocks (all processes have access of FAT table); Just gives an index into the FAT to read data (Could start in middle of file or access deleted data).
⑤ Inode Structure (P73)
File defined by header, called “inode”.
- Problem 1: When create a file, don’t know how big it will become. Solution: Fast File System (FFS) Allocation and placement policies for BSD 4.2
- Problem 2: Missing blocks due to rotational delay. Solution1: Skip sector positioning (“interleaving”); Solution 2: Read ahead: read next block right after first, even if application hasn’t asked for it yet.
⑥ Directories
– link / unlink (rm): Link existing file to a directory (Not in FAT); Forms a DAG.
When can file be deleted?
– Maintain ref-count of links to the file
– Delete after the last reference is gone
⑦ Links
- Hard link
Sets another directory entry to contain the file number for the file; Creates another name (path) for the file; Each is “first class”.
- Soft link or Symbolic Link or Shortcut
Directory entry contains the path and name of the file; Map one name to another name.
⑧ New Technology File System (NTFS) (P82)
Variable extents not fixed blocks, tiny files data is in header.
Master File Table (MFT)
⑨ File System Caching
Buffer Cache: Memory used to cache kernel resources, including disk blocks and name translations.
Cache Size: adjust boundary dynamically so that the disk access rates for paging and file access are balanced.
Read Ahead Prefetching: fetch sequential blocks early.
Delayed Writes: Writes to files not immediately sent out to disk. Flushed to disk periodically.
- Advantages: 1. Disk scheduler can efficiently order lots of requests; 2. Disk allocation algorithm can be run with correct size value for a file; 3. Some files need never get written to disk! (e.g. temporary scratch files written /tmp often don’t exist for 30 sec)
- Disadvantages: 1. What if system crashes before file has been written out? 2. Worse yet, what if system crashes before a directory file has been written out? (lose pointer to inode!)
⑩ Important “ilities”
Availability: the probability that the system can accept and process requests.
Durability: the ability of a system to recover data despite faults. Doesn’t necessarily imply availability.
Reliability: the ability of a system or component to perform its required functions under stated conditions for a specified period of time.

L14 Advanced File Systems
① Transaction
- An atomic sequence of actions (reads/writes) on a storage system (or database) that takes it from one consistent state to another.
- Typical Structure: Begin a transaction (get transaction id); Do a bunch of updates. If any fail along the way, or any conflicts with other transactions, roll-back; Commit the transaction.
- ACID properties: <u>Atomicity</u>: all actions in the transaction happen, or none happen; <u>Consistency</u>: transactions maintain data integrity (e.g. Balance cannot be negative); <u>Isolation</u>: execution of one transaction is isolated from that of all others; no problems from concurrency; <u>Durability</u>: if a transaction commits, its effects persist despite crashes.
② Difference between “Log Structured” and “Journaled”: In a Log Structured filesystem, data stays in log form; In a Journaled filesystem, Log used for recovery.
Crash During Logging: No commit, discard log entries.
Recovery After Commit: Redo it as usual.
Expensive Journaling: Record modifications to file system data structures, but apply updates to a file's contents directly.
③ Flash Translation Layer (FTL)
Random reads are as fast as sequential reads; Random writes are bad for flash storage.
Node Address Table (NAT): Independent of FTL. Updates to data sorted by predicted write frequency to optimize FLASH management.
Checkpoint (CP): Keeps the file system status.
Segment Information Table (SIT): garbage collection. Per segment information.
Segment Summary Area (SSA): Summary representing the owner information of all blocks in the Main area.
④ Distributed File Systems (DFS)
Virtual File System (VFS): Virtual abstraction similar to local file system.
Four primary object types: superblock object; inode object; dentry object; file object. (no specific directory object, treated as files.)
⑤ Problems and solutions
Simple Distributed File System - Problem: Performance. Caching to reduce network load – Problem: Failure, Cache consistency (What if client removes a file but server crashes before acknowledgement?).
Stateless Protocol: A protocol in which all information required to service a request is included with the request.
Idempotent Operations: repeating an operation multiple times is same as executing it just once. Client: timeout expires without reply, just run the operation again.
Remote Procedure Call Protocol (RPC Protocol). Call procedure on remote machine or in remote domain.
⑥ Network File System (NFS)
Defines an RPC protocol for clients to interact with a file server, Keeps most operations idempotent, Don't buffer writes on server side cache.
Three Layers: UNIX file-system interface, VFS layer, NFS service layer.
Write-through caching. Weak consistency: Client polls server periodically to check for changes. Multiple write: arbitrary result.
⑦ Key-value storage systems (K-V Store)
Distributed Hash Tables (DHT): partition set of key-values across many machines.
Challenges: Scalability, Fault Tolerance, Consistency, Heterogeneity (inconsistency in latency and bandwidth).
⑧ Iterative vs. Recursive
> Recursive Directory Architecture: Have a node maintain the mapping between keys and the machines (nodes) that store the values associated with the keys.
+Faster, as directory server is typically close to storage nodes; +Easier for consistency: directory can enforce an order for all puts and gets; -Directory is a performance bottleneck.
> Iterative Directory Architecture: Return node to requester and let requester contact node.
+More scalable, clients do more work;
-Harder to enforce consistency

HW2
① Advantages of inode to FAT: Fast random access to files. Support for hard links.
② Write- behind policy:
Advantage 1: The disk scheduling algorithm (i.e. SCAN) has more dirty blocks to work with at any one time and can thus do a better job of scheduling the disk arm.
Advantage 2: Temporary files may be written and deleted before data is written to disk.
Disadvantage: File data may be lost if the computer crashes before data is written to disk.