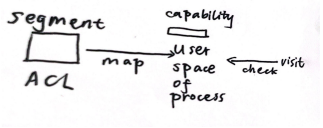
THE HISTORICAL PERSPECTIVE
1 - The Structure of the 'THE ' - Multiprogramming System(THE)
The central abstraction -> sequential processes
Hierarchy -> Layers
Benefit: verity soundness, prove correctness, handle complexity Drawback:x efficiency, loop, hard design
0 Process control, scheduling 1 Memory management, paging 2 Console, message interpreter 3 Peripherals, buffering, stream(virtual device)
4 User level Programs 5 Operator
synchronization: P(-) & V(+)
mutual exclusion, synchronizations
发现太复杂了遂分层
2 - The Nucleus of a Multiprogramming System(Nucleus)
kernel and message buffering(synchronization)
nucleus: scheduling, communications, primitives for manipulating processes. -> flexible design of OSs
internal process: program execution
external process: input/output
round robin scheduling
message buffering: less deadback, high reliability, less efficiency
storage: parent-children allocation
common pool of messages: one process cannot use too many messages.
想搞很多个OS所以抽象出了kernel
3 - TENEX, a Paged Time Sharing System for the PDP-10(TENEX)
real time-sharing system; "virtual machine"->kernel
rich **virtual memory** with sharing and copy-on-write
support for backwards compatibility
virtual memory: BBN pager: virtual -> physical
1)simple memory management, 2)easy sharing
compatibility: rewrite instruction set, and the old one redirects to(trapped) to user space.
纯纯觉得其他OS都不行于是自己开发，弄了虚拟内存
4 - HYDRA: The kernel of a Multiprocessor Operating System
capability-based OS nucleus mechanism && policy
objects(name, type, data, capabilities)

	File A	File B	File C	Printer 1
Alice	RW	RW	RW	OK
Bob	R	R	RW	OK
Carol	RW			
David			RW	OK
Faculty	RW		RW	OK

Capability list
Access Control List

capability rights: kernel rights && auxiliary rights
local namespace(LNS): transitive closure of all capabilities
right amplification called procedure may have more rights than caller (e.g. system call).
ACL: easy-to-use, maintainable, for small system, complex for large system, ↑grain, slow to check
capability: fine-grained,flexible way to right amplification, should control the capabilities cautiously, quick check, easy transfer, link permanent
也是觉得其他OS不行，于是弄了一些抽象(objects, resource, procedure啥的)，把权限整上了
STRUCTURE
5 - Protection(Protection)
use small set of abstractions to unify discussion of protection concepts and mechanisms
why protection: user unpredictability
- domain: communicate by message, strongly rely on the central system

abstractions: objects, domains, access matrix
object:things which need to be protected
domain:the set of entities which have rights to an obj.
觉得现在保护越来越重要，类似于写了个survey of protection，总结了一下重要知识点
6 - Protection and the Control of Information Sharing in Multics(Multics)
segment and virtual memory; shared memory; multiprocessing; hierarchical file systems; online reconfiguration; reference/access monitor; protection systems*; protection domain transitions*; multilevel security policies*
thoughts: permission rather than exclusion, check every access to every object, open source, principle of least privilege, human interface
protection: ACL with capability.
open file -> ACL check, return a file descriptor
file descriptor -> capability for future read

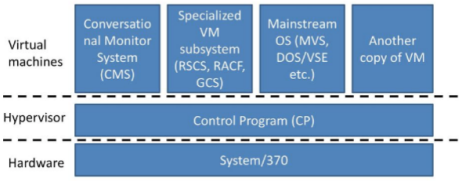


hardware check(MMU)
principal identifier: self, project group, compartment
protection subsystem and protection ring
multics: basic, extendable, securable, kernel codes should be reduced to secure, complex user interface, weak communication, over privileged administrator
multics有关保护的核心概念，更多人用，啥都能干
7 - The UNIX Time-Sharing System(UNIX)
Clean abstractions for files/devices, process management
unifying abstraction: file
no hardware requirement
file exists independently of directories, hard link to file(i-node), soft link to hard link.
no group now(compared to above)
set-user-id -> right amplification; fork; the shell;
UNIX保留了最简洁的必要功能
8 - Plan 9 From Bell Labs
key abstraction again: file
Each node has its own purpose such as file servers, cpu servers, router servers, user terminals.
advantage: cost effective
everything visible in the namespace; Plan9 uses the relative local namespace.
9P: protocol connecting clients with the server, RPCs
third level file storage: WORM(write once read many), snapshots, can slow devices such as tapes.
multithreading: kernel-level threading.
some ideas live: UTF-8, rfork, snapshots
网络开始发展了，上一个团队决定走火入魔让一切都网络化文件化包括CPU，File System等等。
9 - Medusa: An Experiment in Distributed Operating Systems Structure(Medusa)
Provide a solution to the distributed OS structure, previous distributed OS structures are not satisfying.
1) only 1 os in 1 cm2) Oss in all cms3) os cache
1)not reliable, speed slow;2) no room;3) hard design
utility: single OS module abstraction, distributed among computer modules;
activity: a process on a given node for a given utility;
pipes: communication channels among activities.
task force: collections of concurrent activities, activities access files by descriptors, which are stored in descriptor lists.
descriptor list: activity(PDL), task force(SDL), processor(UDL) **robustness and modularity
Medusa认为治疗分布式系统性能的方式还得是换结构不能换设备
10 - Pilot: An Operating System for a Personal Computer(Pilot)
designed issues for personal computer OS.
characteristics: single-user, no share resources, single address space, single language support(Mesa), limited protection, accept hints(e.g. when to page out data), premature TCP network package, defensive protection
file data accessed by mapped to the virtual memory file: permanent/immutable, with distinct uid (filesystem: files; folders; file attributes; operations, metadata; file path, access control, log)
pilot implements Mesa, M depends on Pilot feature
一款全新无隐私的*单用户*个人电脑操作系统
SYNCHRONIZATION
11 - Monitors: An Operating System Structuring Concept(Monitors)
monitors: resources, procedures, lock, queue.
最初是为了在多进程中更好调度操作系统模组资源
12 - Experience with Processes and Monitors in Mesa(Mesa)

Properties	Hoare	Mesa	Java
Monitor lock	All procedures	entry	synchronized
Condition vars (Y/N)	Y	Y	No/yes (one implicit CV per object)
Wait (Y/N)	Y	Y	Y
Signal (Y/N)	Y	Notify & broadcast	Notify & notifyAll
Granularity (Coarse/Fine)	module	Monitor class & monitor record	Monitor/instance sync. blocks
Aborts (Y/N)	?	Y	Y
Nested calls (Y/N)	Y	Y	Y

changes compared to above: program structure, dynamic process creation, dynamic monitor creation(just like a class in C), nested wait, exceptions, scheduling, I/O
in mesa, notify does not directly select one to run, but just a signal that others can run, no instant context switch, and scheduling outside monitors.
java no condition v just for simplicity.
deadlock: in one monitor or between two. priority inversion.
《monitors实际操作系统体验指南与提升方法》
DISTRIBUTION
13 - The Distributed V Kernel and its Performance for Diskless Workstations(V Kernel)
Large file servers, diskless workstations.
Message-based IPC for communications.
vkernel IPC: sync, fixed size kernel buffer, fix size message, reduce queue and buffer, separate data transfer facility for efficient large transfer → inefficiency
一款客户端无disk的分布式系统通信方案
14 - The Sprite Network Operating System(Sprite)
network OS, name and location transparency, automatically process migration(RPC), caching network file system
reason: network, **large memory**->cache, multiprocessors; have transparent namespace: prefix table(dynamic, broadcast);
sprite caching: server-side, client-side, problem: consistency -> version number, caching disabled
另一款分布式系统的方案(大内存，带通信)
15 - Experience with Grapevine: The Growth of a Distributed System(Grapeving)
distributed system for email
message service: deliver, buffer
registration service: registration database, naming, authentication, access control, resource location
message service consults the registration about the location, the registration services communicate with each other; the registration uses message to maintain the distributed registry.
drawback: the connection between servers and low-bandwidth
scale: level of indirection(multiple levels)
distribution and replication -> transparency
典中典的分布式邮件系统方案，分布式中的Multics
16 - Implementing Global Memory Management in a Workstation Cluster(GMM)
global memory management in a cluster of machines connected by a high-speed LAN. Enlarge the memory to minimize the average data reference time.
assumption: network faster than disk for paging.
algorithm: replacing the oldest page in global memory by approximation.
试图提高内存利用率的内存社会主义化逆天方案
OS/ARCHITECTURE INTERACTION
17 - The performance of Microkernel- Based Systems(micro kernel)
a minimal approach to kernel: IPC, virtual memory, thread scheduling.
benefit: easy to secure, reliable, more extensible, the upper implementation can be specialization
no benefit: system calls and more context switches.
monolithic kernel 太繁重了，换一个更小的，可以更好保证安全性与正确性
18 - Exokernel: An Operating System Architecture for Application-Level Resource Management(exokernel)
also for the specialization of the application OS, better performance. Customize the OS by moving everything to user-level. functions:
track resource ownership

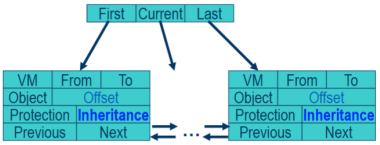
protection(secure bindings, decouple authorization from use)
resource revocation(request->abort)
更更小的，现在exokernel只做资源分配和保护
19 - The Origin of the VM/370 Time-Sharing SYstem(VM370)
motivation: access to full hw, isolation among users.



CP: provide virtual resources to VMs.
two instructions: privileged(only CP, otherwise trapped into CP), problem mode.
single address space for memory management.
RSCS: remote spooling and communication system(network virtualization)
因用户多了想分开用电脑而产生的虚拟机方案
20 - Xen and the Art of Virtualization(XEN)
feature: not influence each other, support different OSs, little performance penalty.
full virtualization drawback: some characteristics of the OS cannot be implemented, should have provided the real hw to guests.
Xen: isolation among VMs, minimal performance overhead, limited modes to guest OSes.
some privileged instructions have to be trapped, which can be better implemented with x86 rings.
memory: x86 hard due to h/w TLB misses and managed page tables. need to change the OS source to make h/w access guest OS page tables directly.
Comparison: no need to modify guest OS from VMWare.
另一个虚拟机方案但半虚拟化从而保存特性+提高性能

VIRTUAL MEMORY

21 - Virtual Memory Management in VAX/VMS(VAX)
VAX: virtual address extension
VMS: virtual memory system
the layout: P0(heap); P1(stack); Sys(OS); Reserved;
OS is now an extension of user address space, directly accessing user code and data, then no TLB flush with system calls.
VM h/w: page table; TLB;
user page table: kernel address space, can be paged out; kernel page table: physical memory space, cannot be paged out.
To improve the performance of paging:
1) local page replacement; 2) page caching; 3) clustering.
algorithm: FIFO -> low overhead but problem such as belady's anomaly: memory ↑ page fault↑
page caching: clean list, dirty list, LRU.
reason for cluster: small page sizes.
一款可以适用于大内存或者小内存的虚拟内存系统
22 - Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures(VM)
mach's difference: 1) support large, sparse virtual address space(use link list); 2) children inherit vm for sharing(cow); 3) memory mapped files; 4) user-level pagers and backing store.



(address map)
resident page table: kernel use to maintain the state of physically resident pages.
copy-on-write: shadow object(modified parts and the pointer to the unmodified parts)
pmap: hardware dependent info cache.
use global FIFO pool rather than resident sets of user program -> easy to tune and can have external pagers
之前的虚拟内存系统都要依赖硬件，做一个不依赖硬件

I/O AND FILE SYSTEM

23 - A Fast File System for UNIX(FFS)
global allocator: across cylinder groups
local allocator: blocks in a cylinder group or cylinder larger block size; data in the same file in cylinder groups; inodes and data close; fragments in a block to reduce waste; replicated superblock; parameterize.
bookkeeping information for each cylinder group.
之前的文件系统太不行了，在文件位置（硬盘维度）以及block/fragment上上强度
24 - The Design and Implementation of a Log-Structured File System(LFS)
assumptions: 1) large main memory -> read cost low, and can used for write buffer; 2) small file access; 3) disk bandwidth increases.
solution: treat disk as a single log for appending, buffer small writes into large writes to utilize the bandwidth.
inode map: also appended to log, but could be optimized by keeping it in memory.
segment cleaner: thread(in a block) with copy combine the age of segment to achieve: free space in cold segments is more valuable than free space in hot segments.
recovery: checkpoint(30s) and roll-forward
version
啥都不管直接往下写降低写成本这个木桶短板
25 - Soft Updates: A Solution to the Metadata Update Problem in File Systems(soft updates)
problem to solve: maintain the consistency after writing; the writing should have orders.
sync writes drawback: large overhead
soft updates: write-back caching, async writes.
granularity: individual entries of inodes and dirs.
break cyclic dependency: write back twice/rollback, turn the block into some safe state before.
一个一个写太慢了用file cache一起写但是要考虑依赖，确保硬盘的一致性，操作系统写硬盘有认证和顺序
26 - The Rio File Cache: Surviving Operating System Crashes(Rio)
assumption: use persistent memory, no battery issues, but enable the memory to survive OS crashes without writing data to disk → warm reboot
goal: performance of main memory with the reliability of disk, write-back performance with write-through reliability.
OS protects the file cache part of the memory by: 1) virtual address read only; 2) writing to physical address disabled and insert a codepatch check into the kernel;
drawback: the metadata update should be ordered in memory.
确保file cache的一致性从而替代write-back，操作系统写内存要有顺序和认证

SCHEDULING

27 - Scheduler Activations: Effective Kernel Support for the User-level Management of Parallelism(Scheduler Activations)
goal: functionality && performance
user-threads: 1) improve performance and flexibility; 2) OS unaware of user-level threads
kernel-threads: 1) high overhead of system call, context switch, too general to add code complexility; 2) OS aware of kernel threads, integrating events and scheduling
scheduler activation: virtual processor, can be modified by communication between user and kernel.
user-level: control over the scheduling.
N threads: M processors model
way to protect critical sections: the recovery, the application can choose to context switch when told of the preemption or blocked.
upcall: OS->user; downcall: user->OS
一款全新集成user-thread和kernel-thread优点thread
28 - Lottery Scheduling: Flexible Proportional-Share Resource Management(Lottery Scheduling)
a randomized resource allocation mechanism for CPU with relative priorities.
benefit: 1) like scheduler-flex, well with the relative rates, multi-user servers, dynamic workstations (adaptively give CPU to interactive jobs), soft real-time applications.
ticket transfer: priority inversion; inflation/deflation: create or remote tickets to manage them; currencies: hierarchy scheduling; compensation tickets;
issue: can't express response time differently from share.
那个彩票调度方法，带随机概率防饿死

SMARTPHONES

29 - Cells: A Virtual Mobile Smartphone Architecture(Cells)
constraints of other methods: 1) resource limited; 2)response time is important; 3) device virtualization not good; 4) existing paravirtualization requires modifications on OS.
VP: a set of processes, but in their specific domains
solution: container virtualization(namespace is shared, the OS interface used by all virtual machines) && hardware virtualization.
foregroundVP && background VP. VPs can be set to not use specific devices.
手机上也需要多用户且分离，但资源有限所以，感觉不像完全的虚拟机，更像是某种container
30 - TaintDroid: An Information-Flow Tracking System for the Realtime Privacy Monitoring on Smartphones
taintdroid: an extension to the Android mobile-platform that tracks the flow of privacy sensitive data through third-party applications.
implicit: control flow
explicit: data flow → propagation rules.
granularities: variables, messages, methods, files
tradeoff: accuracy vs performance
小心隐私泄露.jpg

BIG DATA

31 - The Google File System(GFS)
assumptions: failures are the norm; big files; appending writes; anatomical write append without synchronization overhead.
master server:metadata(namespace, access control, mapping from files to chunks, locations of chunk servers) consistency management, garbage collect, migrate chunks, communicate with heartbeat
chunk server: (primary server, lease->decide order) large file of 64MB.version number,checksum,replicated
APIs: open, delete, read, write, snapshot, append(at least once, possibly inconsistent)
drawback:single master, bad latency, bad performance for small or many files, relaxed consistency.
对于大数据现状提出了一些对于分布式文件存储系统的新要求与新适应
32 - Bigtable: A Distributed Storage System for Structured Data
distributed multi-level map, fault-tolerant, persistent, scalable {row:string, column:string, time:int64}->str
columns->column family; rows->tablet->tablet server
centralized -> single master server for metadata
大数据分布式存储的类数据库表格方案
33 - MapReduce: simplified data processing on large clusters(Map Reduce)
parallel programming model for large clusters
heartbeat msg between the workers and master
fault tolerance: reschedule task when almost done
drawback: inefficient for multi-pass algorithm, no efficient primitives for data sharing.
大数据处理的时候为了利用集群资源把数据处理分为了输入和输出，中间结果可以给集群用来优化调度
34 - Finding a needle in Haystack: Facebook's photo storage(haystack)
assumptions: write once, read many. never modified, barely deleted.
haystack: high throughput, low latency, fault tolerant, cost effective, simple metadata
one superblock, multiple needles, x photos in a file
drawback: no album level abstraction, no privacy protection, photo not totally deleted.
商业竞争导致技术变革现状