6.033 Review

- Worse is Better: (rarely)
  - { MIT/Stanford philosophy
    New Jersey " "

Worse is better → simplicity of implementation is main concern.
Correctness & consistency for the other one.

(Point the PC back to user mode)

Say a user prog invokes sys. routine. Sys. routine should point PC to the routine.
New Jersey doesn't deal with this (because not simple) while MIT sees it wrong.
Because in high demand, get 90% functionality.

Def. of complex sys: sys is a set of interacting parts collectively have a behavior. Complex → large # of components, many ppl req. to maintain.

Problem: many emergent properties
- Scaling: not same as designing for 100's of for billions.
- ~~reduce~~ if one is wrong, others would be

Hierarchy/layering → reducing connection

herac-25: (every 2 years).

Follows Therac-20 and other machines

Main prob: 2 modes - Electron therapy

X-Ray therapy mode

	E.T	X-Ray (Photon therapy)
Energy	5-25 mEV	25. because X-ray flat tener
Current	X	100X

Where E.T is active & current set to X-ray → dangerous.

Before hardware locks preventing dangerous state.

Therac-25 only relied on software. There were cryptic error msg. malfn comes up, operator has choices → restart
→ Press (P)

Since error msgs were cryptic & operator would press (P) detecting was rough.

Simplified by copy, if happens quickly enough can enter bad state.

Keyboard Handler . After finish type.

↓

Data Entry complete Flag —MEOS (Mode Energy)→ T. Table (wait to adjust TT (takes 8s))

If operator enters D.E. and modifies within 8s, not identified by T.T.

① Reliability vs safety.

if seemed to do nothing wrong doesn't mean safe.

② lack of defensive design. → self checks

Not making cryptic error checks

③ lack of two locks → single pt of failure.

~~① Single pt of failure~~

~~② rep UI~~

-unrealistic risk assessment, company uses fault tree analysis. Unrealistic → prob of software failing → have to test all parts some parts difficult to assign value to.

~~bef~~ Friendly UI vs safe UI.

~~Assigned~~ 0 probability to ~~these~~ Software errors ← Problem

Not about locks being acquired in wrong order. Problem of not having locks

T.T sensors were there. ATT did not responded to Therac 20's complaints but to Therac 25 ← false.

Client/server:

- call funcs on other machine, look like local call.

RPC: How would want to work on RPC:

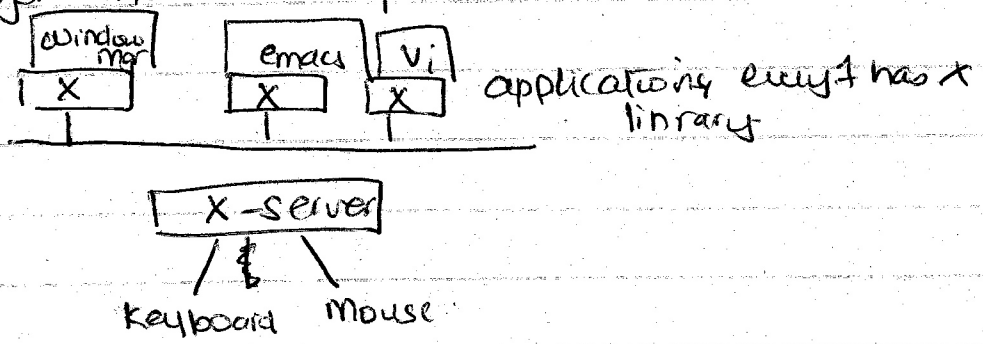
at-~~least~~ least once: Resend packages until get something from client.

At-~~least~~ most once: Send it once, if don't receive don't send more than once. Amazon: don't get credit card more than once, but might not get anything. CI

Exactly once: Hard to do.

X-windows: client/server (just switched).

In general, server is your machine.



Want to draw line/shape on your screen.

2009.

If overlapping windows, move window out of way. Do you send request? Usually not.

Naming in file sys:

- Provides abstraction eg DNS.
- Provides user friendly way.

UNIX file system:

Block layer → File layer → Inode # → Filename → Path → Abs. Path → Symbolic path

File → Inode #

inode

One to one relation.

Hard links v/s Symbolic link.

↓

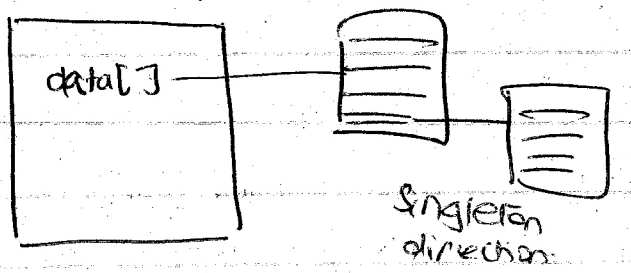
ref to actual inode, Pt. to same inode

No simple way to go from inode to path name.

Questions about how look up.

2008 — good set of questions (Qs/6).

UNIX V6



Can't store large files. Scheme in 2008 pitfalls.

try to bump ~~file~~ size to 64 bit.

What happens if change block addresses to 4 bytes, how much increase filesize & look at 2008 & to gain idea of math expected.

Lec 5, 6, 7 (UNIX, OS, Eraser)

Name

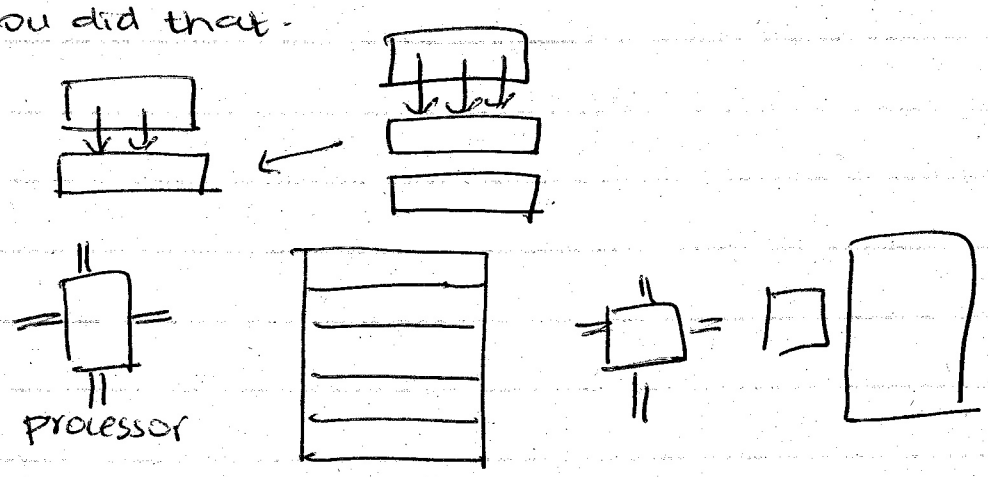
OS structure: ~~Abstract devices~~

an OS
- Abstract devices.
- run many prog. together/multi program

Techniques
- Virtualization
- abstraction

2 layers communicate in certain defined interface.

Virtualization put 3rd layer w/o telling upper layer that you did that.



read 0x01

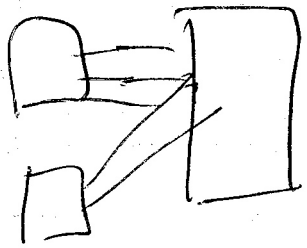
Virtual memory → put something in btw.

Talk.

Abstraction: take lower layer, change lang. you use

Virtual memory.

Can have many processors share same memory.



No matter how messed up your code is never writing to what other is writing

- VMem
- Threads
- Communication
- ~~Stream~~ link

Another Virtualization:

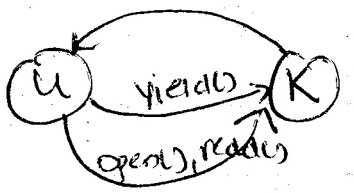
Threads:

Virtualizing processor

Basic abstraction. For threads:

-yield(): Stop current thread go to next one

yield() is kind of OS procedure called system call. It goes in kernel. If prog. your own sys. call won't be able to. ~~Can~~ Since your bit will be in user mode not kernel, can't write your own Sys. call.



Why want to control how go from U to K?

Securing, and reliability. Simplify the interface.

Very limited way of going from U to K.

nstead of complete freedom, gives limited freedom to user

tate where user ~~has~~ should not touch but Kernel has to touch.

e.g. in virtual memory, each processor knows where its mapping table is. To do that looks at that table. Has to be changed but done in restricted way. That is can at a Kernel can do, but user can not.

ommunication b/w processors:

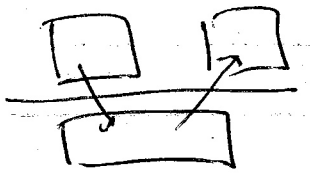
PA	PB
X=0	X=0

X defined in context of the program. Talking about diff. X both of them.

But what if want to communicate?

SHARED BUFFER → OS simulate the network links.

↳ Bounded Buffer: when 2 prog want to communicate, make sys. call SEND(), RECEIVE(). Don't know when to send or receive.



When call send(), Kernel takes care of it. Keeps an array. When somebody sends msg, put in queue, receive → take out of queue.

Problem → Concurrency issues.

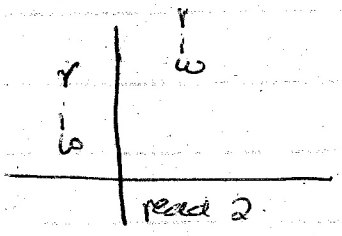
When someone calls Kernel, another Kernel code may be attempting it at same time.
What could go wrong? (Race condition).
Why race cond. a problem?

Races:

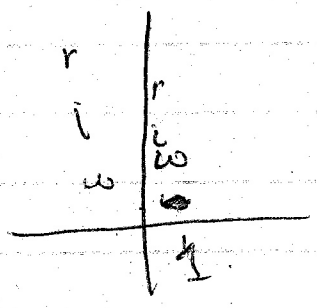
x=0 | x++ | x++

↑ abstraction breaks internally ~~read x~~

x++ → { read x r0
inc r0
write r0 x



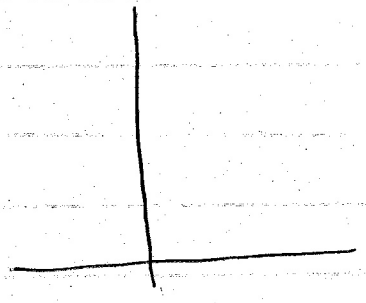
Not all races are bugs, but most of them are.



w x 1 | r x
w y 1 | r y

One operation

x	y	
0	0	{ could be diff. value.
0	1	
1	0	
1	1	



When ppl reading thing → NOT a problem.
Problems occurs when at least 1 is writing.

when you write → change state.

Soln: ① Use locks. (not only way of dealing it).
mechanism to control interleavings. For us, they are abstractions.

lock(1)	~~test~~
x=1	read x
y=1	read y.
unlock(1)	

what happens? not solved problem.

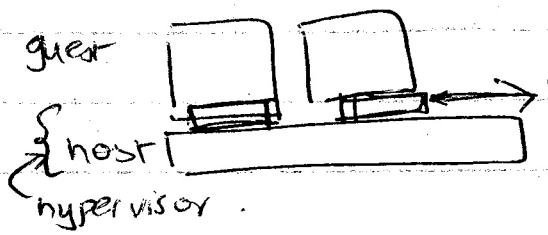
lock(1)	lock(1)	2 possibilities.
x=1	read x	read 0's or
y=1	read y	read 1's.
unlock(1)	unlock(1).	

Simon's Paper: (2005).

- Use hierarchies.
  - Hierarchy → collection of components (organized by strength of interaction) (see paper)
  - Evolve naturally because stable system.

Virtual machines:

1. Terminology:



Can have diff instances of guest OS running on underlying OS

Underlined OS

VMM: responsible for all virtualization.
Takes care of CPU virtualization & memory virtualization.

CPU Virtualization:

making it seem like sharing single processor among different processes simultaneously.

(1) Trap & emulate:

(VMM + host = hypervisor)

take all guest code run on processor.

Caveat: run in user mode

Not prob. for applications. But for guest kernel mode → ~~wat~~ has to run in kernel code. Results in traps.

At this pt. VMM comes & emulates it.

It doesn't work in x86 because of some instructions like popf.

fails silently.

when run popf in user mode (doesn't change flag, but doesn't trap)

in kernel mode (changes flags / ~~interrupt~~ update interrupt flag

sets eflags, interrupt)

The signalling mechanism is not present.

2. Binary Translation:

① Rewrite problem instructions like "popf"

② All others copy them identically. (IDENT)

③ Rewrite control flow instructions

Translated code lives in Translated Cache that belongs (region of memory.)
to VMM.

TC → only guest kernel mode instructions are translated.

This is what processor executes.

Control flow e.g. jmp

```
xor
popf
jmp 24
```

24 "x"

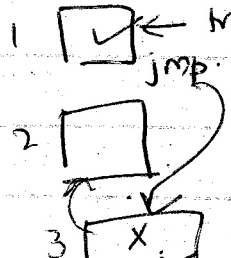
In previous step rewriting

xor ✓

popf = popf translated to 3 lines.

now line 24 doesn't correspond to x

Thus you write jmp to point to correct address. Why dynamically?



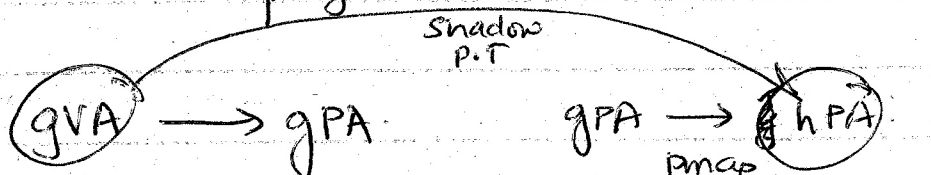
You translate 1, 3, 2.

HW Support: User/Kernel bit + VMM mode bit

2 levels of HW page tables instead of one

Memory Virtualization:

Use shadow page tables.



This mapping is in page table, maintained/managed by guest OS.

managed/maintained by VMM.

Shadow PT created by VMM, actual HW Pg. Tb ptr points to it.

In TLB, shadow pg table will be cached.

= guest phy. page "X"

Pg. Table:

gVA	gPA
1	7
2	3
0	"X"

pmap

gPA	hPA
7	11
6	5
3	4
x	y

Shadow Pg. Table

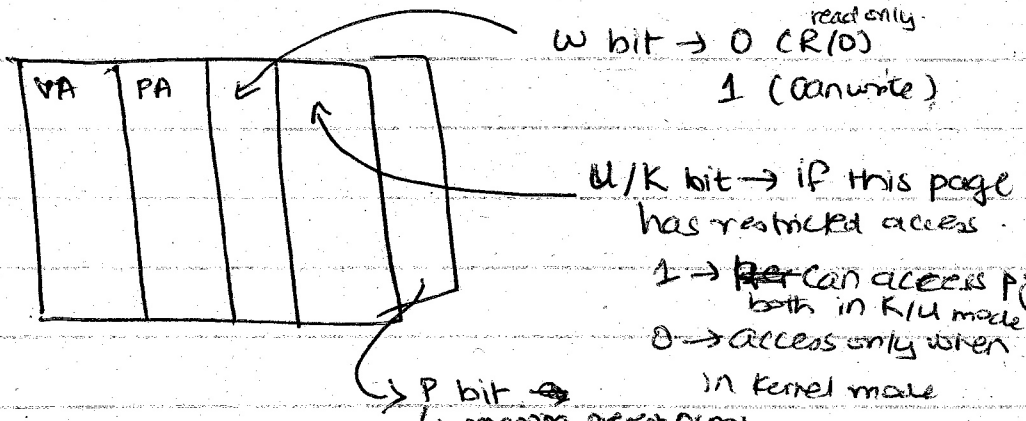
1	11
2	4
0	y

Problem: If guest changes 1–11 to 1–6 mapping in STP also needs to be updated. 1–5.

We use traces for that ← Signalling mechanism for when pages need to be updated. Marks the entry corresponding to the page where page table lies in.

mark 0–y as (R) takes it as a trap.

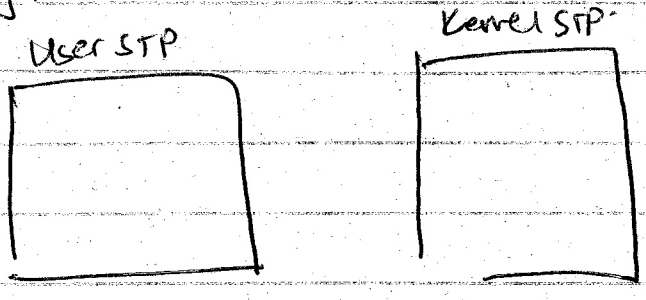
Mark entry for page that contains page table.
i.e. 0 → y. everytime try to write to y which is X which is where there is page table, update it.



Guest OS believes that it has U/K bit.

HW doesn't know abt it. That's why have

2 pg tables



2011 QS.

Guest pg. table = pg. 4

gVA	gPA	P	W	V
0	5	1	1	1
1	9	0	1	1
2	11	1	0	1
3	4	1	1	0
4	7	1	1	0

maintained by guest OS

pmap

gPA	hPA
4	2
5	6
6	4
7	3
11	8

maintained by host OS.

when virtual bit set to user → user mode STP

else kernel STP.

User Shadow PT:

(P → present bit)

gVA	hPA	P	W	U
0	6	1	1	1
1	—	0	—	—
3	—	0	—	—

(no mapping) — for gVA 1

we say not present because user bit set to 0 so when

Kernel SPT

gVA	hPA	P	W	U
2	8	1	0	1
3	2	1	0	1
4	3	1	1	1

lives in host OS, guest kernel is running in user mode w.r.t. to host OS.

can't set to 1 as 3 maps to 4 but 4 is the pg ~~table~~ where pg table exists so using traces set to 0.

~~For~~ Any access is a user access to the host OS. For kernel, user bit is never 0.

In any SPT, user bit is 1.

maintained by guest OS

So who