Why Plan 9 is not dead yet
And
What we can learn from it

Ron Minnich

Advanced Computing Lab

Los Alamos National Lab

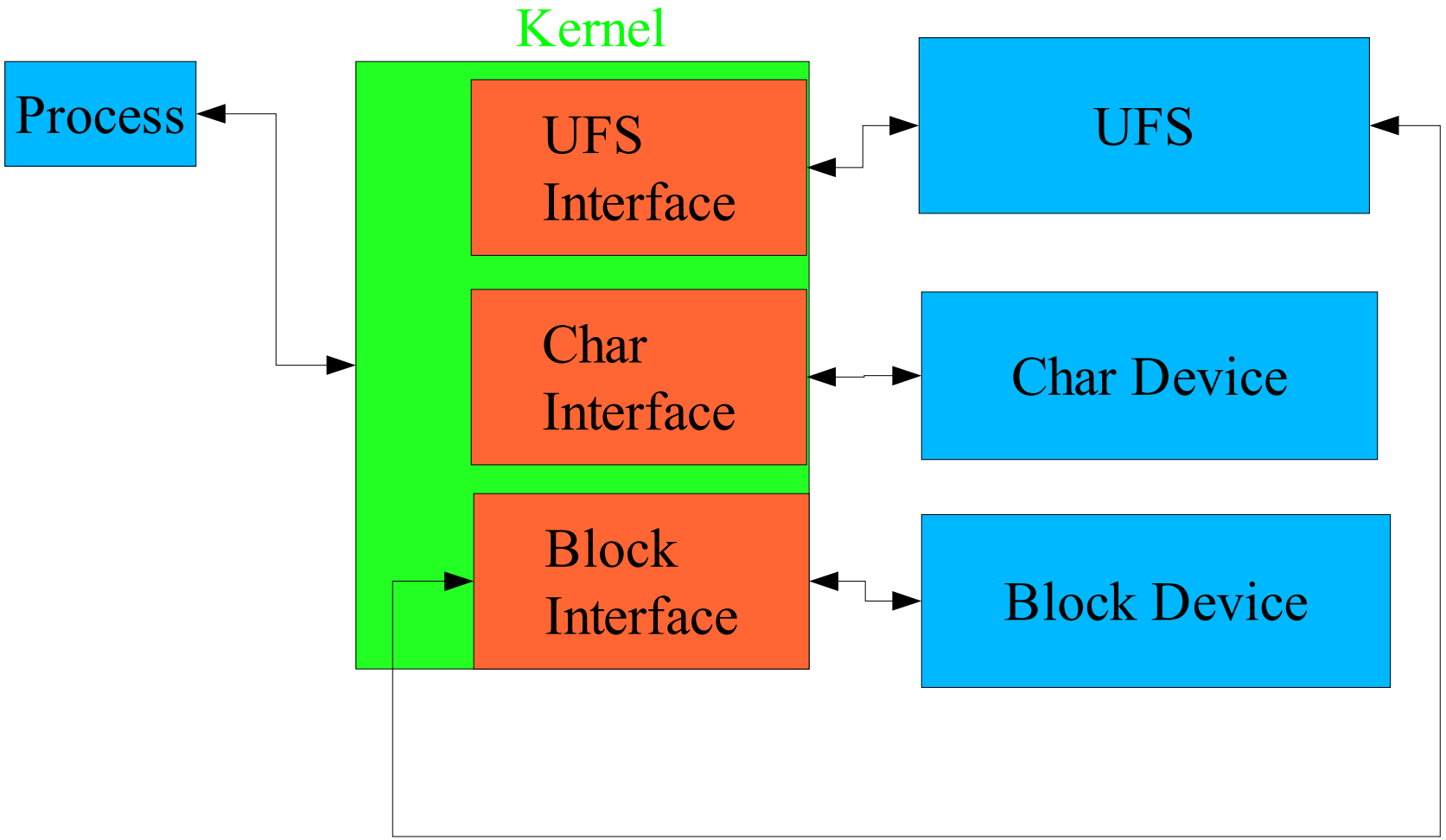
LA-UR-05-4132

Unix

- Simple open/read/write/close with streams of bytes
- Resources are named via a visible path
- This is such an advanced concept that NT still does not do it (try ls \devices sometime)
- It was also extremely controversial

Two guys and a computer

Early Unix


Kernel
Process
UFS Interface
Char Interface
Block Interface
UFS
Char Device
Block Device

Unix kernel system calls

- Pathname
  - Chdir, chmod, chown, creat,exec,link,mknod,mount,open,stat,umount,unlink
- FD
  - dup,fstat,gtty,stty,seek, read,write,open,close
- Void param or output-only
  - fork,getgid,getpid,getuid,pipe, wait, time, times
- Pid
  - Kill, signal
- Other
  - Break, csw, nice, profil,ptrace, setgid, setuid, stime

So in Unix, "Everything is a file"

- Or so we thought
- As it happened, the model broke down fairly quickly
- In our case, it happened with this:
- (no picture)
- The DEC DAC

The DEC DAC problem

- Problem: you are reading from the DAC (/dev/dac)
- OK, it's a file
- But you want to let a process read from something else (i.e. Filter) and then ... open a file ... oh wait ... IPC has no name
- Recall that kill() was one IPC mechanism, pipes another, both anonymous

We didn't feel so smug at that point

- The “everything is a file” model had broken down fairly quickly
- And for a fairly important use
  - Take a file, apply filter, present to users as file
  - Can't do it!
- And then things got worse ...

Oh, heck

- Did I say everything was named?
- I lied
- Un-named things:
  - Processes
  - File descriptors
- Note that these got fixed later
- What didn't get fixed

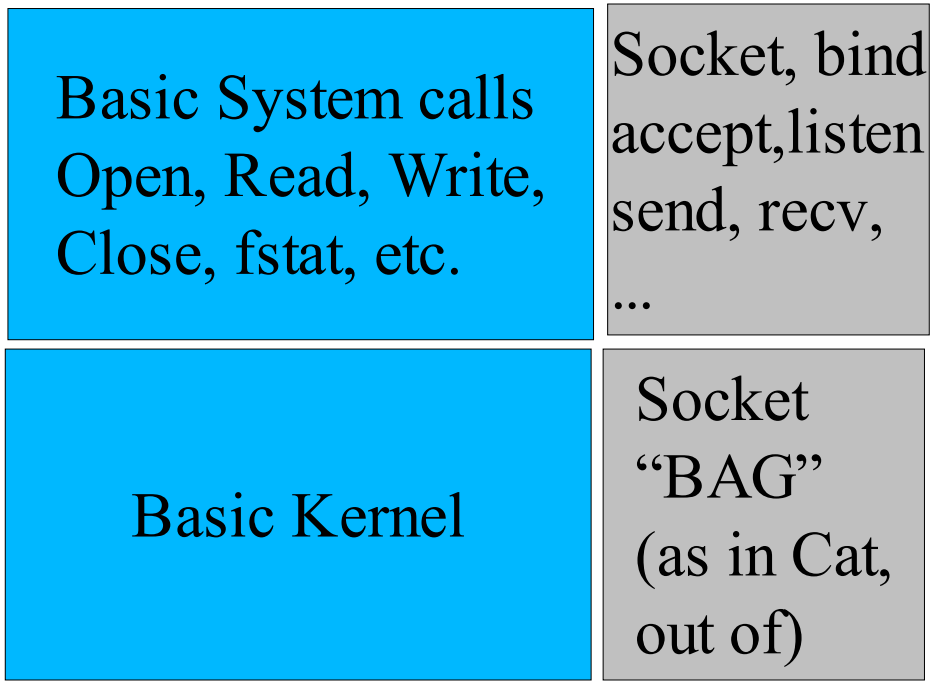
When things started to break

- Unix model had common access method different types of resources
- e.g., “/etc/passwd” and “/dev/dk0” are both path names
- But get to different kinds of things
- So: one interface, multiple resources
- Late 1970s, Unix started to break
- Networking was the cause

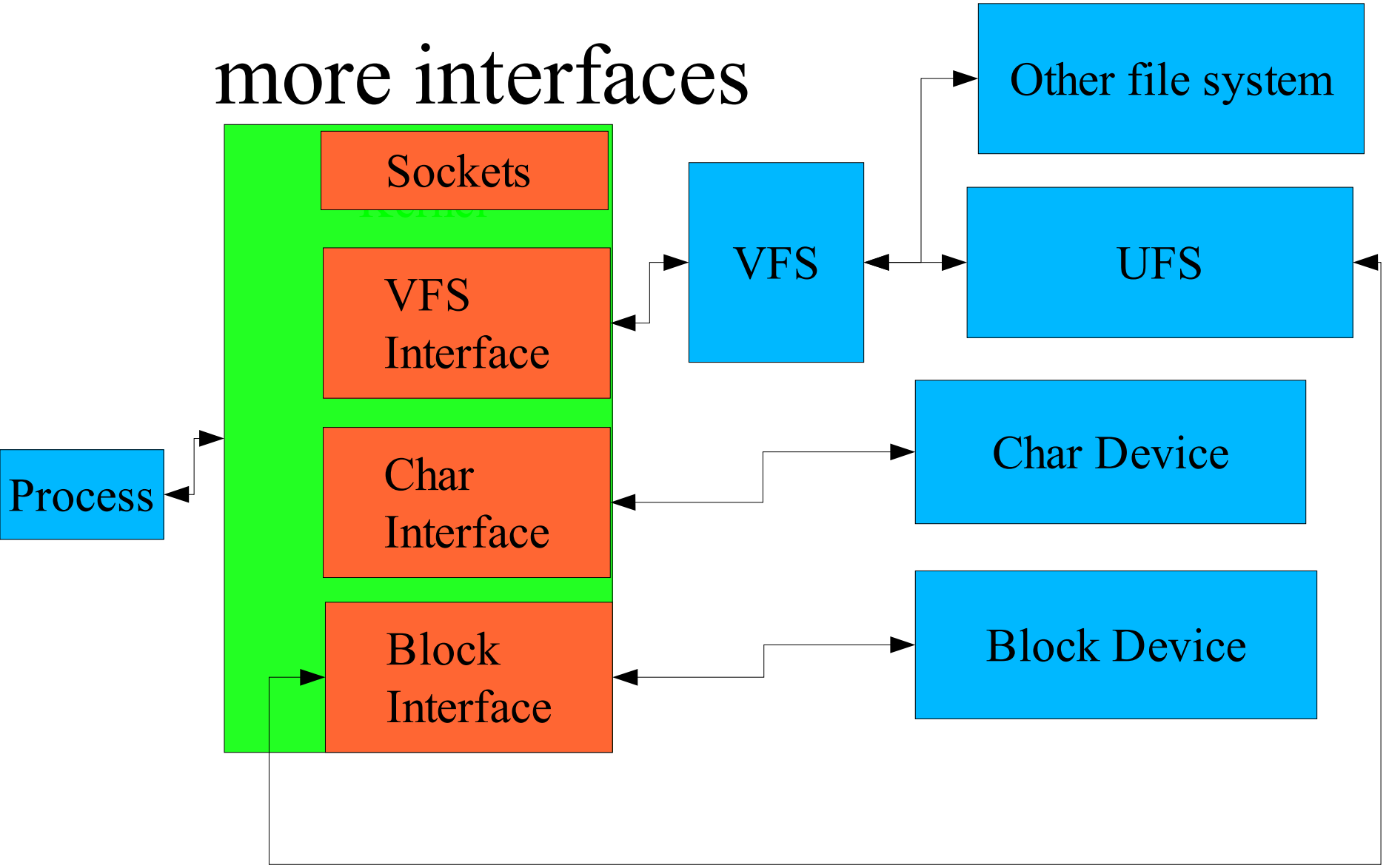
Networking added to Unix

- Initially, people tried stuff like:
  - "/dev/tcp/harv" (RAND RFC 618)
  - One interface, different resource
  - Failed, for various reasons
  - Unix had no way (architecturally) at the time to switch out to protocol streams from file names
- So binary names for sockets were created (BBN? Still not sure)
- Which broke the model big time

What Unix started to change to (1978)



Later ...
more interfaces



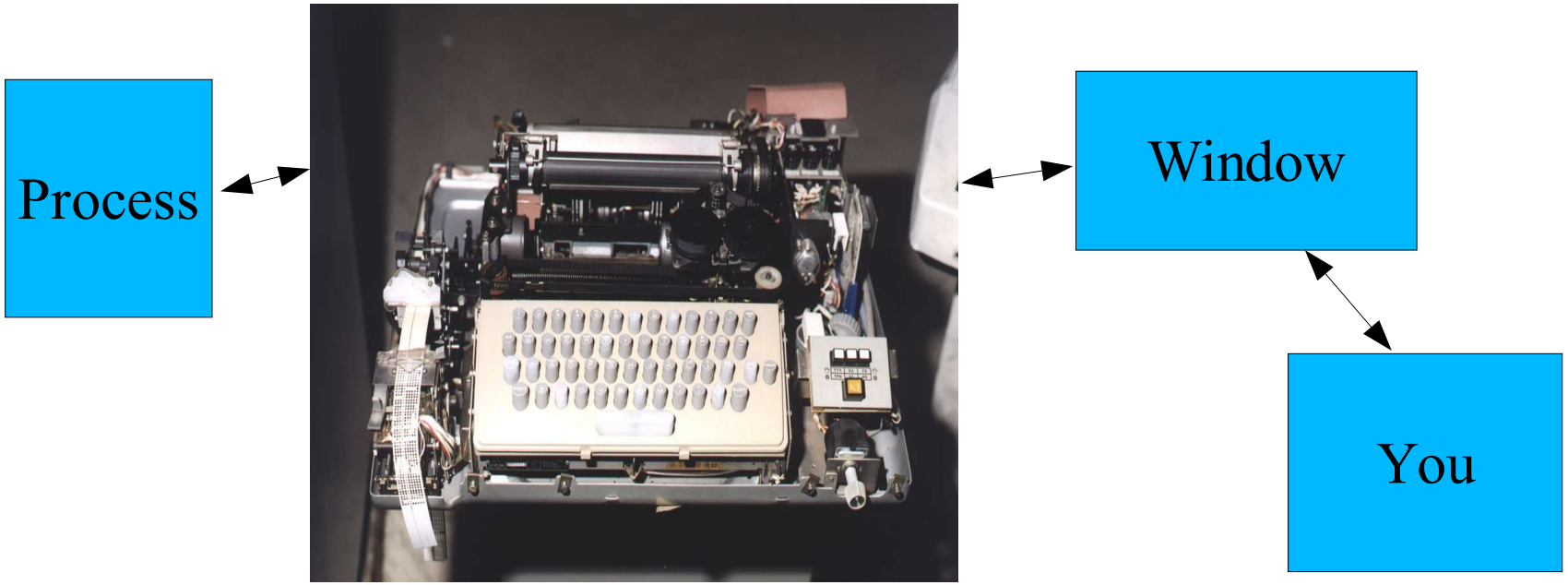
So look what's happened

- Incompatible new subsystem
- Multiple interfaces, multiple resources
  - Duplicate functions for similar operations
  - Basic consistency of the kernel started to fall apart
- Breaks pathname model completely
  - “This is such an advanced concept that NT still does not do it (try ls \devices sometime)”
  - Oh, wait, Unix doesn't either! There are all kinds of resources that have no pathname now

Then to now

- Over the years, legacy mechanisms that should have gone away did not
- The basic input model is still a tty
- Don't believe me? Type 'stty' at an xterm and tell me why a window has a baud rate
- Also, tell me why I can't edit the text in an xterm window

Xterm window



- Wow, how advanced
- 1,000,000 times the performance and there's still an ASR-33 in the middle ...

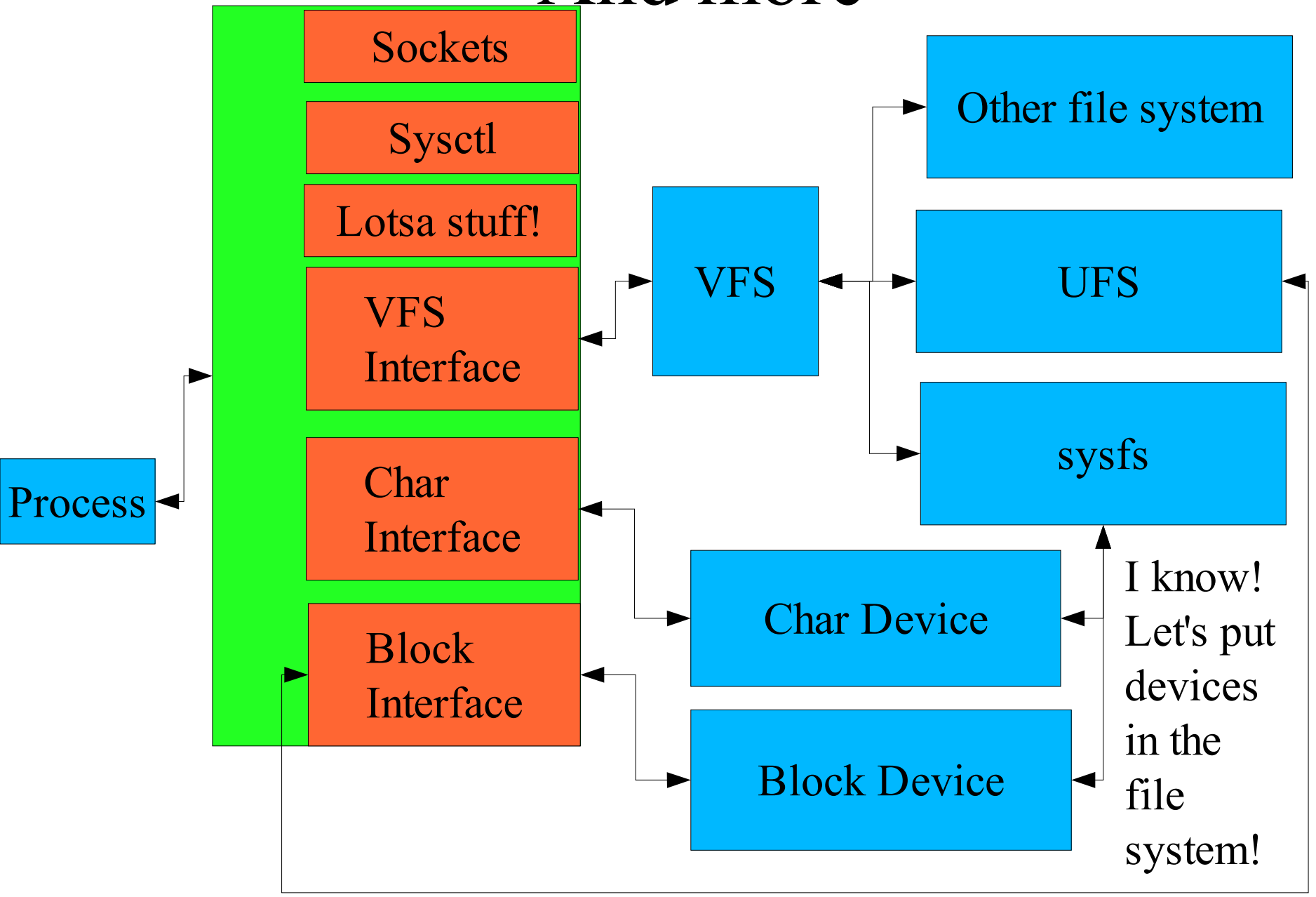
Where we are

- 300 system calls and counting in Linux
- Most are different types of entry points to different subsytems, continuing the trend started by sockets
- Kernel layering gets more and more complex, revision by revision

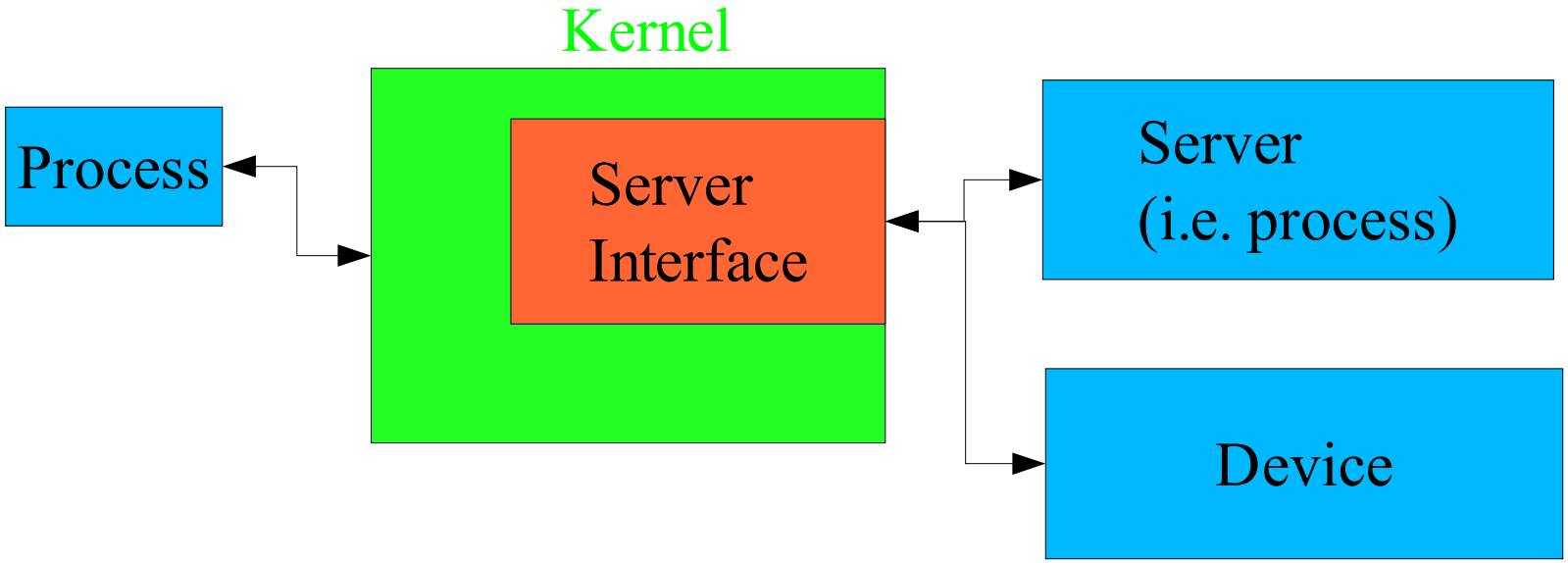
What went wrong?

- Unix had the active process model
- And the passive data model
- But it lacked an active data model
  - Farber suggested this in 1978 and we all thought it made no sense
  - He was right, as usual
- It also lacked the richness of structure to allow naming resources
- And some things were just wrong

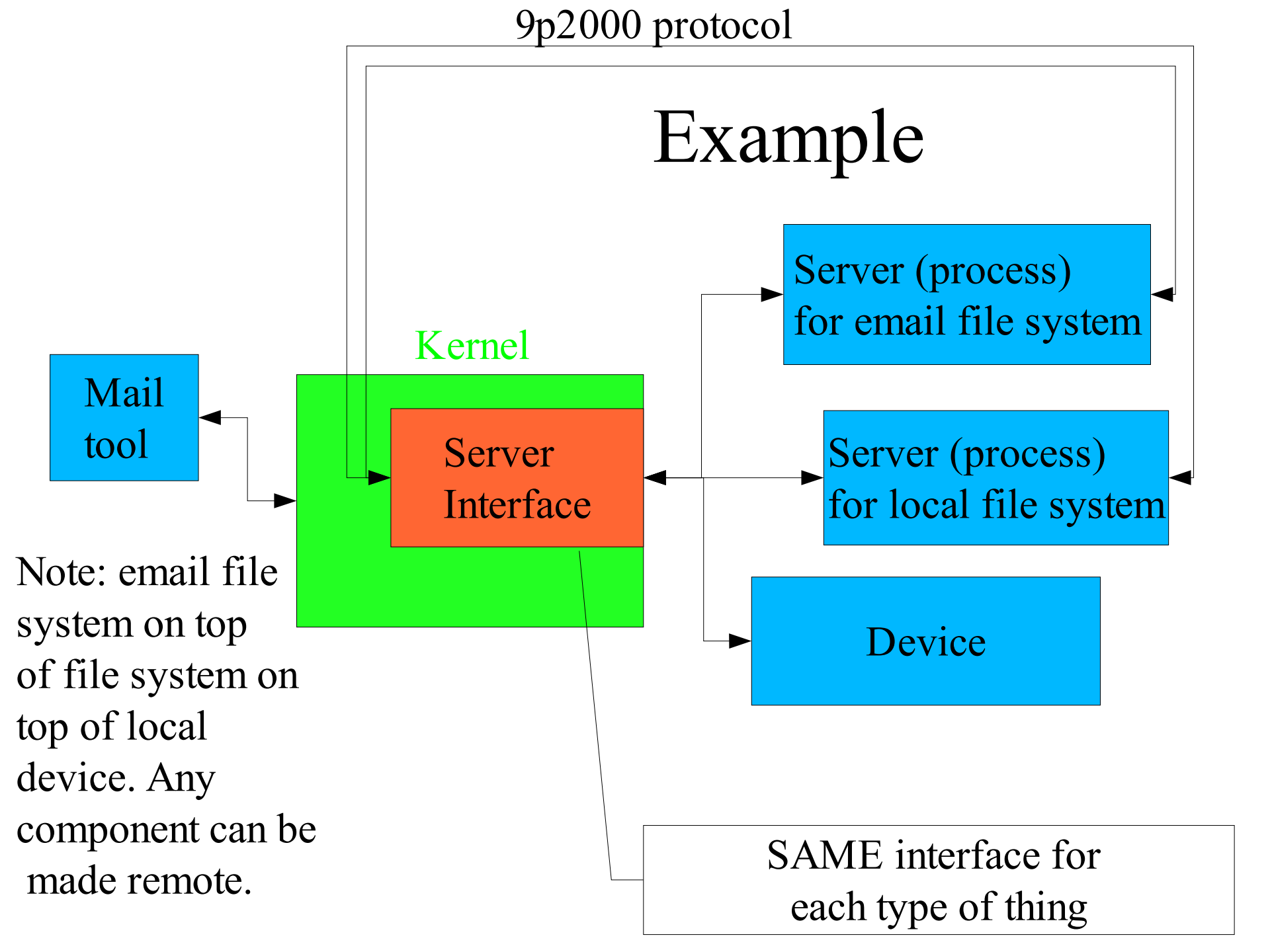
And more


Sockets
Sysctl
Lotsa stuff!
VFS Interface
Char Interface
Block Interface
Process
VFS
Other file system
UFS
sysfs
Char Device
Block Device
I know! Let's put devices in the file system!

A better model



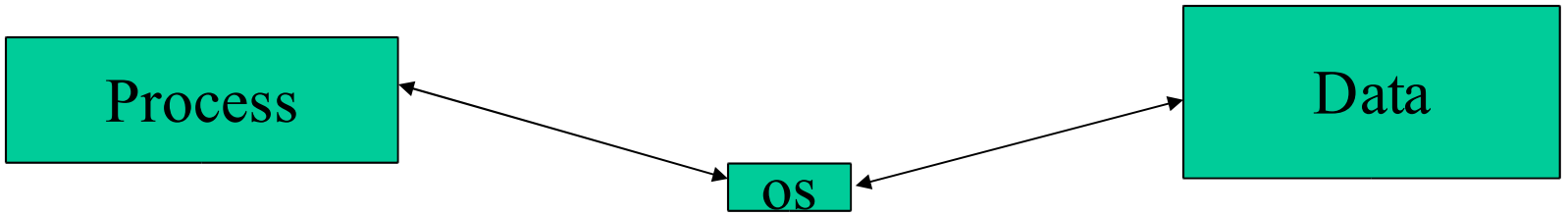
With a common server interface, location of services is no longer important. The differentiation of char/block is archaic. Processes no longer distinguish servers and devices. Devices no longer have numbers.


9p2000 protocol
Example
Server (process)
for email file system
Kernel
Mail
tool
Server
Interface
Server (process)
for local file system
Device
Note: email file system on top of file system on top of local device. Any component can be made remote.
SAME interface for
each type of thing

The key idea

- Unix Kernel is an “I/O multiplexer”
- This newer Kernel is a “server multiplexer”
  - Mediates access from processes to servers
  - Note that servers are often processes
  - Recurrence as in the email example
- And devices, in operation, look just like servers to programs
- Uniform interface to all resources
- Standard protocol (9p2000) to *all* servers

We can have an active data model with named resources



- The data is no longer operated on by the OS
- The OS serves as communications medium for Processes to Data
- The OS does not operate on data directly

Advantages of this model

- Processes talk to servers
- What can be servers?
  - File systems, for one thing
  - Or file systems that aggregate file systems
    - Mirtchovski's ResourceFS (for the 9grid)
  - Or devices such as disks
  - Or Grid schedulers
  - Or graphics devices
  - Or security systems

The boundaries are much less important

- Is the “file system” on my node, or somewhere else?
- In this model, that doesn’t matter so much
  - You can have it in one place, or another
  - So Barney can stop beating me up
- Since the OS provides comms, the comms can be to a local component or via network to a remote component
- Applications see same set of resources

Are there systems which implement this

- Yes, there is a system called Plan 9
  - From Bell Labs
- Just had its fourth version release 2002
- Had its open source release party June 2003
  - DOE helped

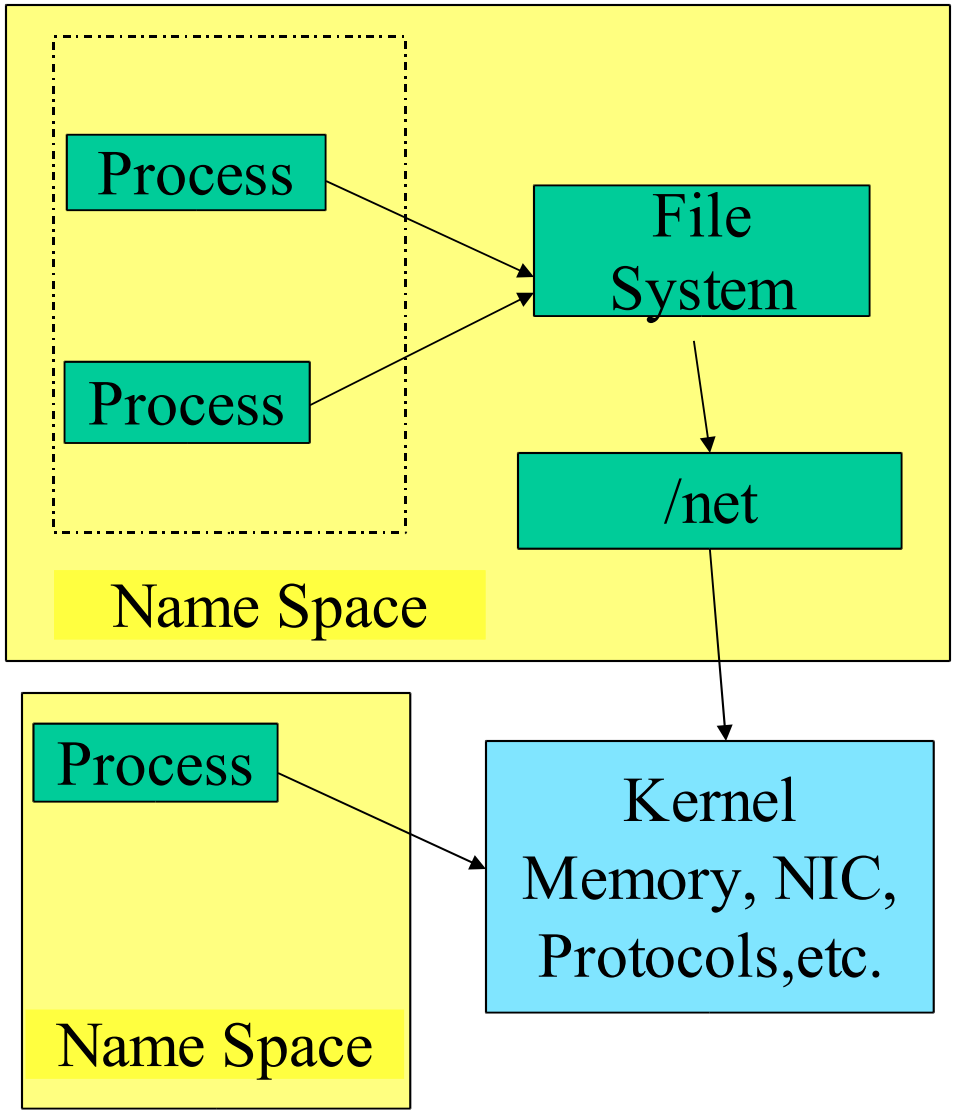
What Plan 9 is

- Not like anything you've seen
  - Not a mini-, micro-, nano-, or other fad kernel
- Core OS is fixed-configuration set of "devices"
  - Means "anything that *has* to be in the OS"
  - E.g. Memory, Net hardware, etc.
- Everything else is a "Server"
  - File systems, windowing systems, etc.

(Some) Plan 9 attributes

- Small kernel with fixed architecture
  - Good for HPC nodes
  - In-kernel devices (e.g. Memory, NIC, TCP)
  - Out-of-kernel services (e.g. TCP, file systems)
- Yep, you really can put servers for same function in or out of kernel
- Devices can be prototyped as servers first
  - This would be one way to implement, e.g., Portals
- Real-time scheduler
- Simple and fast

Plan 9 structure



- Processes attach *servers* as needed
- Attaches are inherited
- Not visible outside the group
- In this example one group has attached remote files
- Other group only needs IPC so it has no other services

Plan 9 system calls (< ½ as many as BG/L "LWK")

- Path
  - Bind (NOT NETWORKING), chdir, exec, mount, open, create, remote, unmount, exec, remove,
  - notify (signal but with arbitrary-length byte string)
- FD
  - close,dup, fsession, fauth, fstat, seek, stat, wstat (i.e.chmod), pread, pwrite, fd2path
- Other
  - Rfork, alarm, exits, noted, rendezvous,
  - Segattach, segdetach, segfree, segflush

Discussion

- How disks look
  - Note lack of need for loopback mount
- Fossil
  - What was I doing yesterday
- Venti
  - Never lose a file again
- Where did getpid, kill and ptrace go?
  - Simple: replaced by file I/O to /proc files

Conclusion

- Can an OS preserve the original simplicity of Unix
  - Most things are paths; common access model for different resources
- ... and yet attach the current capabilities of Unix?
  - Networking, Network file services, graphics
- Plan 9 indicates the answer is “yes”
- Except Plan 9 does far more than Unix

More conclusion

- So where were we in 1978?
  - In a pretty good place, but going the wrong way
- Where are we
  - At the end of a long road, which may be ending (or not)
- Where can we go?
  - Find a different road
- But it won't be easy ...

Demo of CPU node.

“You wouldn't share needles,

so

Why would you share a compute node?