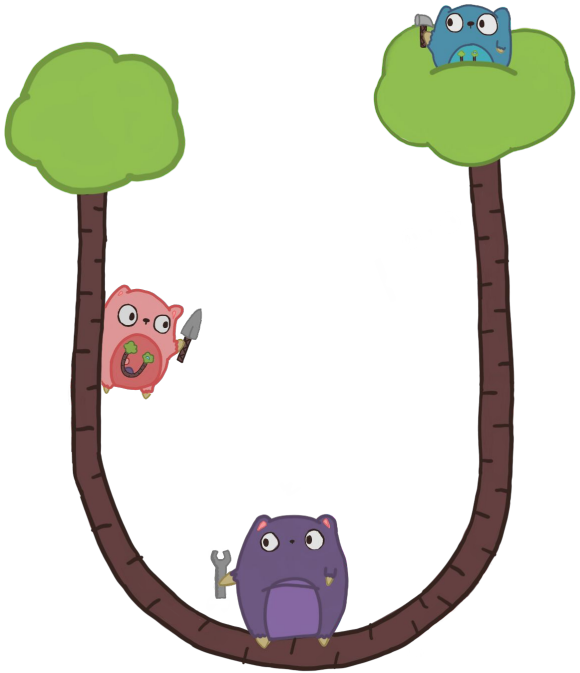
building small stateless network-controlled appliances with coreboot/linuxboot and u-root's cpu command

Ron Minnich, Google

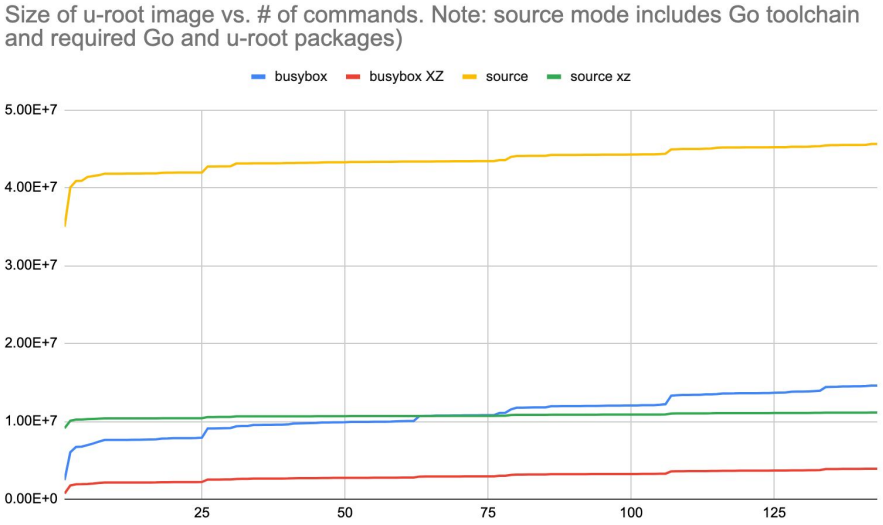
Based on the u-root project (u-root.org)

- Unix commands in Go
- Source-to-source transformation “busybox”
- *Any* command can be in our “busybox”
  - Very different from all other Go “busybox”
- E.g. github.com/nsf/godit (Emacs)
- Works across Linux, BSD, Plan 9
- How can that fit in 4MiB FLASH part?
- “Go binaries are big!”



“Go binaries are big” -- not always

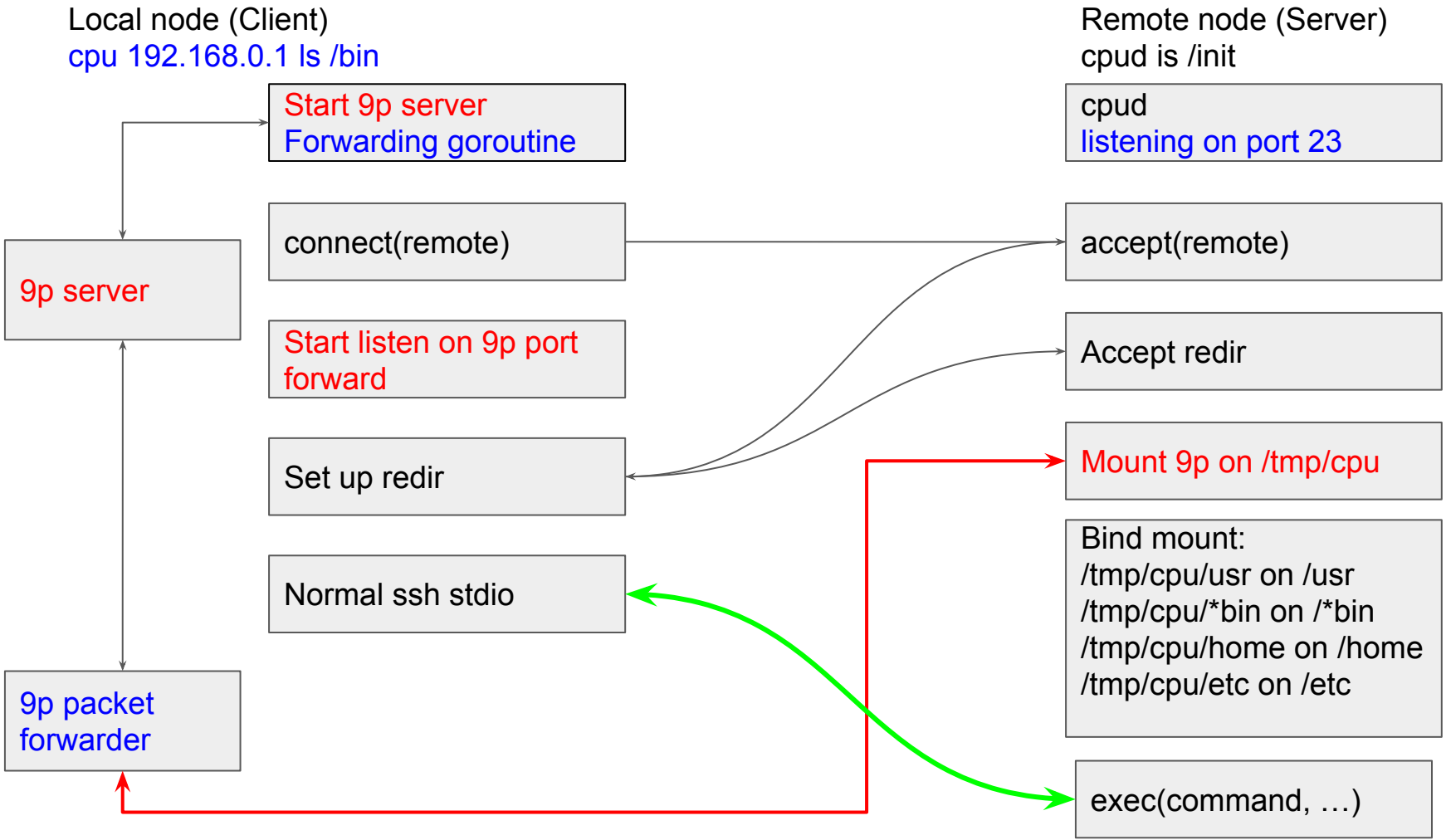
- Note the big first jump
- Flattens out fast
- More commands -> more packages
- Each new command brings less new
  - E.g. hdparm adds 6KiB
- The bottom red line deployed in Google data centers
- Typical install is 3M initramfs
- But why? Why not C?
- Because the most recent C exploit is … today (“sudo exploit”)



Cpu: where ever you go, there your namespace is

- Type a command that looks like an ssh
- Unlike ssh all your local resources (namespace) visible on remote machine
- Commands, files, *everything*
- Looks like NFS home directories/NFS root but:
- The mount name space on remote node is private
- Served by *your* server, not a sysadmin-managed server
- In this demo, linuxboot/cpud is the coreboot payload on the apu2
- Apu2 comes up and only knows how to function as a cpu server
- Plan 9 hackers everywhere rejoice!

Demo hell


Local node (Client)
cpu 192.168.0.1 ls /bin
Remote node (Server)
cpud is /init
Start 9p server
Forwarding goroutine
cpud
listening on port 23
9p server
connect(remote)
accept(remote)
Start listen on 9p port forward
Accept redir
Set up redir
Mount 9p on /tmp/cpu
Bind mount:
/tmp/cpu/usr on /usr
/tmp/cpu/*bin on /*bin
/tmp/cpu/home on /home
/tmp/cpu/etc on /etc
Normal ssh stdio
9p packet forwarder
exec(command, …)

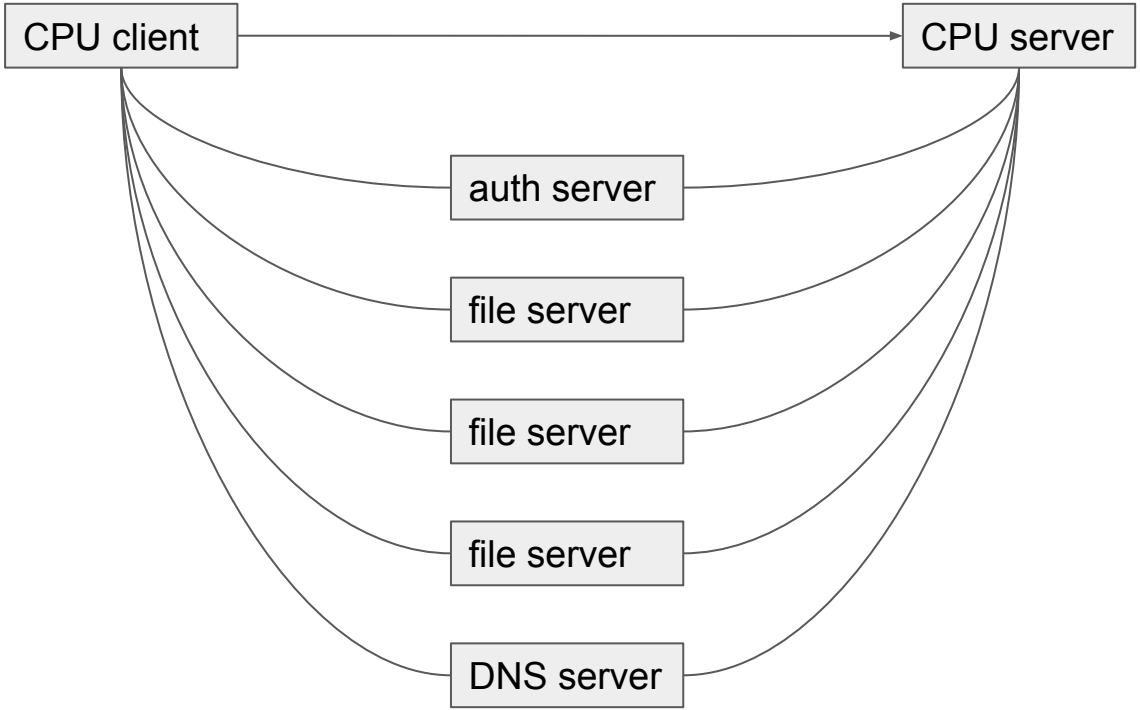
There’s a bit of a trick in that mount step

- On Plan 9, cpu forwards client *name space* in the environment
- Name space describes the set of mounts
  - Mounts are not privileged in Plan 9 and are process-private (and inherited)
  - You may have seen CLONE_NEWNS in man 2 clone
  - Al Viro (Plan 9 fan) brought that in to Linux ca 1999
- One element of the name space is *factotum*, connection to auth server
- This allows the new process on cpu server to authenticate to other servers
- Other elements describe mounts of file servers

Typical plan 9 namespace

- bind /root /root
- mount -aC '#s/boot' /root
- bind / /
- ...
- mount -a '#s/factotum' /mnt
- ...
- mount -a '#s/cs' /net
- bind -a /bin/disk /bin
- mount -a '#s/dns' /net
- cd /usr/harvey

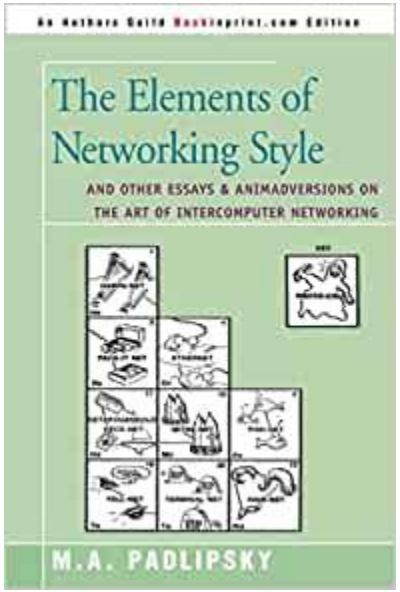
Plan 9 recreation of name spaces



Not an option on Linux!

- No widely accepted auth, naming infra
  - We tried to set this up on the 9grid in the 2000s but proved impossible
  - Ipfs, perkeep are trying but not really there; upspin has it, but no users
  - Gdrive comes very, very close but … not quite there
- Linux/Unix users have moved from network file system (1990s) to local!
  - “NFS” largely a way to backup and share files
  - This is a really strange backwards evolution driven by laptops among other things
  - Chromebooks changed the dynamic a bit, but are moving back to local (“offline docs”)
- How did we get here?

Resource sharing vs. remote access

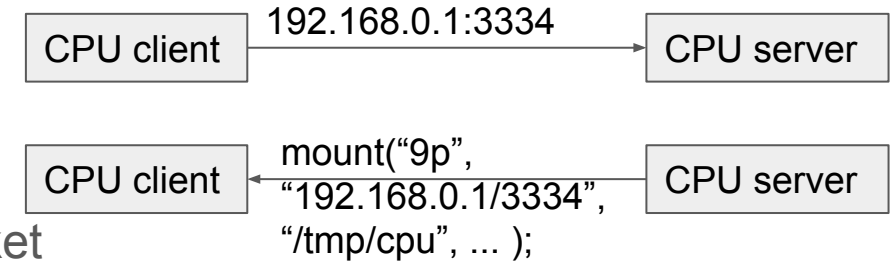


Sharing name spaces on Linux

- But my laptop/workstation are not file servers!
- Comms issue: what transport?
- What to do?
  Only option is to have the cpu client become a 9p server
- Fortunately, this is “easy”
- Chris Koch wrote a beautiful Go 9p server for Gvisor
- We have had an sshd in u-root for several years
- Just a simple matter of programming
- How do we plumb it up?

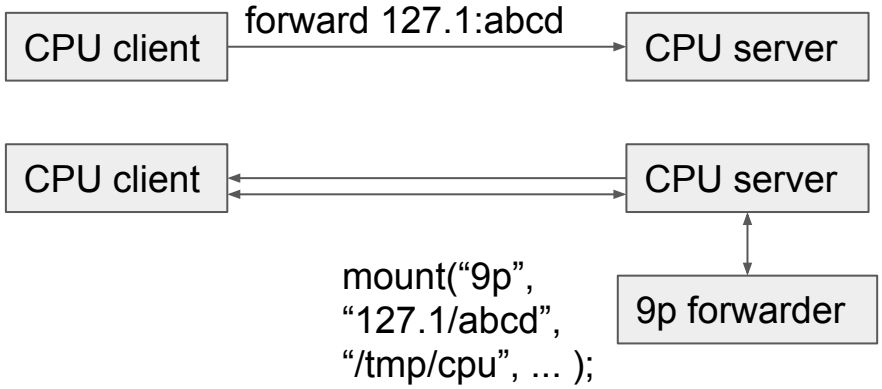
V1: cpu client exports 9p service on socket

- Client gets 9p server socket
- Communicates that host:port to cpu server
- Server mounts /tmp/cpu via that socket
- Problem: do we make it a priv port?
- Problem: what if “bad guy” jumps in?
- Only allow one mount?
- Doesn’t help much: converts perpetual hole into race condition
- This worked but was not enough



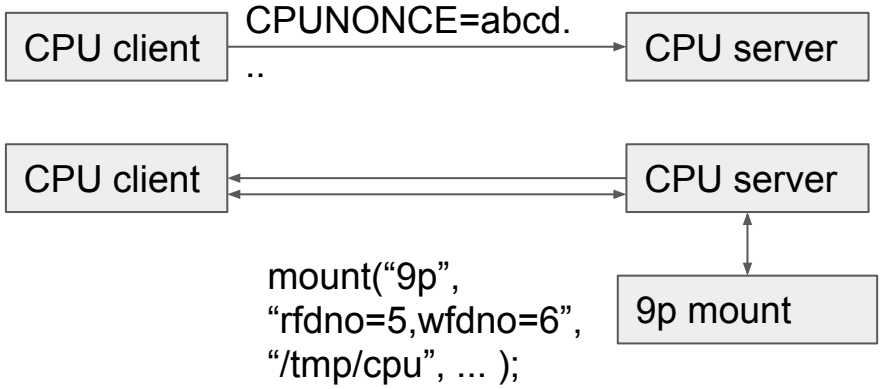
V2: mux 9p and ssh traffic, cpu server does mount

- Client sets up remote port forward via sshd
- Server kicks off 9p forwarder goroutine
- goroutine mounts /tmp/cpu via that local socket
- Forwards 9p RPCs over ssh
- Better: not exposing client port, just local port
- Really want that to be more private



V3: mux 9p and ssh traffic, mount on fd

- Client sets up remote port forward
- Sends CPUNONCE as 32-byte string
- 9p goroutine creates a socketpair
- Once mounted writes CPUNONCE value back over socket
- 9p server in cpu client only accepts this once
- And it *must see the nonce* first
- Based on how factotum works



But how do we trust a cpud in the first place?

- For now, we just do
- But system transparency can provide more assurance
  - https://www.system-transparency.org/
- The idea being that we can have assurance that the entire stack up to and including cpud is measured
- Very similar to what chromeos does with its stack
- Or our ideas on attestation
- Ssh trust issues apply to cpu

Some cool things about Go

- One ssh package did port forward, another did key unwrapping
- Needed both
- Would have been a nightmare in C

```
// ossh can unpack password-protected private keys.

ossh "golang.org/x/crypto/ssh"

"github.com/gliderlabs/ssh"
```

- Refer to ossh or ssh
- I know it’s just namespacing, but it’s more convenient than C++ for me

Other details

- I’ve built NERF images that only have a cpud
  - From HPC days: one daemon, one kernel, in flash
- Next step is to create such images coupled with system transparency
- Minimal bare metal image
- That we can mistrust less
- Provides all resources to the programs running
- This is how HPC systems have been structured for a long time
- Also has similarities to facebook’s approach
- And where I wish most people’s stacks would go

Cpud mounts 9p on /tmp/cpu

- Cpud unshares its mount points
- Does a private mount of tmpfs on /tmp
- Hence, the /tmp/cpu mount is always private
- Is that all we need?
- Can we say PATH=/tmp/cpu/bin:blah HOME=/tmp/cpu/.. Etc?
- Sadly, not
- Too many built-in assumptions in glibc/various distros
- In addition to the mount, need bind mounts

Final namespace

none on /tmp type tmpfs (rw,relatime)

127.0.0.1 on /tmp/cpu type 9p
(rw,nosuid,nodev,relatime,sync,dirsync,uname=rminnich,access=client,msize=65536,trans=fd,rfd=9,wfd=9

127.0.0.1 on /lib type 9p (...)

127.0.0.1 on /lib64 type 9p (...)

127.0.0.1 on /usr type 9p (...)

127.0.0.1 on /bin type 9p (...)

127.0.0.1 on /etc type 9p (...)

127.0.0.1 on /home type 9p (...)

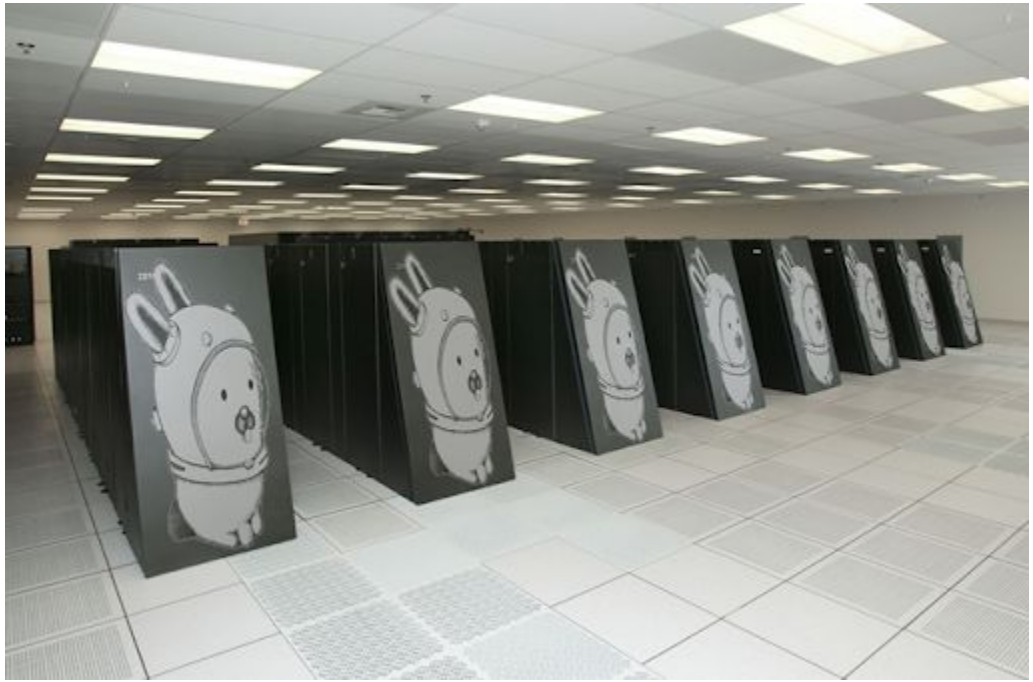
Heterogeneity: connect to ARM cpud from x86

- In Plan 9 world, all binaries for all architectures are present on all machines
- Rooted at, e.g., /amd64, /sparc, /arm, etc.
- Can transmit namespace via namespace switch
  - Once it's implemented :-)
- Default namespace:
  - "/lib:/lib64:/lib32:/usr:/bin:/etc:/home"
- ARM namespace:
  - "/arm64/lib:etc.etc.:/home"
- Also, on machines that have full bin
  - "/home"
- Or serve from a cpio!

Next step: convert programs to packages

- The basic cpu code is right
- Client is 700LOC (includes server)
- Server is 450 LOC
- These need to be converted to packages
- Then the fun part
- Vcpu -- vector cpu
- One cpu client, large number of servers
- Handy for test racks
- Another thing we did in HPC

CPU in HPC



CPU in HPC

- Plan 9 on Blue Gene
- Mail files on a server in Texas somewhere
- Binary on another Plan 9 system
- So:
- cpu bgl
- import themailmachine /m/mail
- run the mail binary provided by NJ file server on the cpu session
- Voila: $100M mail reader
- The best part: we booted plan 9, and mail worked first time we tried it
- As a demo, during a talk, at a DOE conference
- The value of getting the foundation right

What it’s like to use cpu?

- It’s wonderful
- What drove me to finish it up: chipsec
  - Giant wad of python, files, etc.
- You can stop worrying about installing packages on remote systems
- All the commands you have are available to you
- You can forget about distros
- All the distros you have are there for you
- You can forget about needing scp:
  - cpu a flashrom -w somefile etc. etc.

Building systems with cpu images

github.com/linuxboot/mainboards

OSFC slack workspace, u-root channel, see u-root.org

I’m happy to help