File Systems with Containers

Tyler Caraza-Harter

Outline

Refresher: Unified File System Layout (and chroot)

Bind Mounts

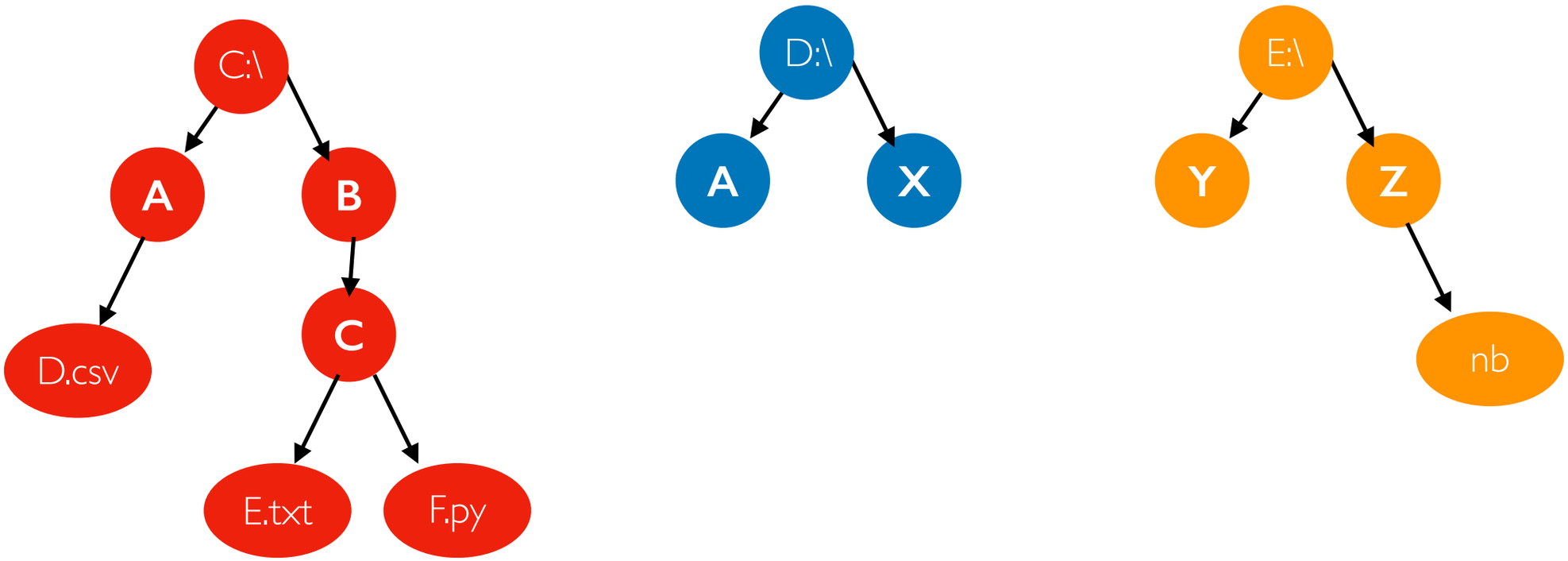
Union/Overlay File Systems

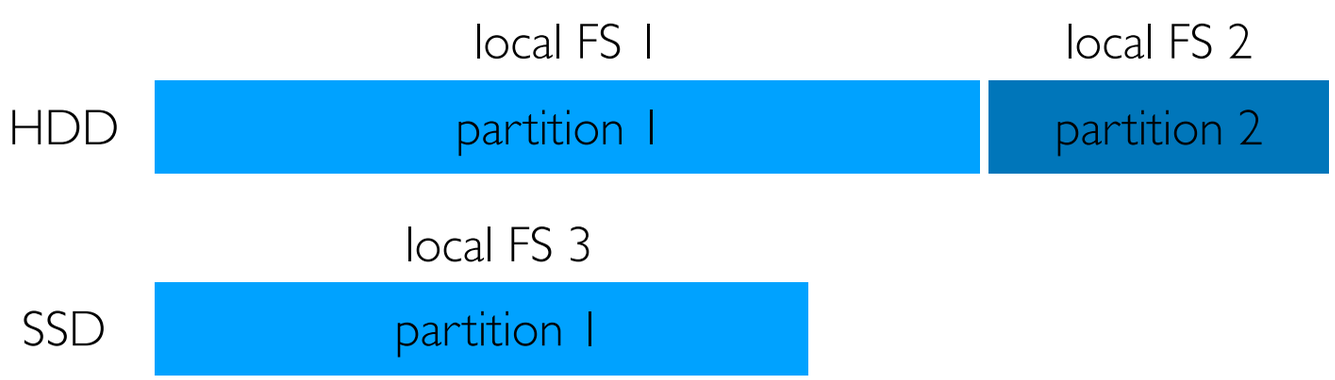
Docker Layers

OpenLambda

Multiple File Systems: Windows Approach

have multiple trees (each is a "drive")





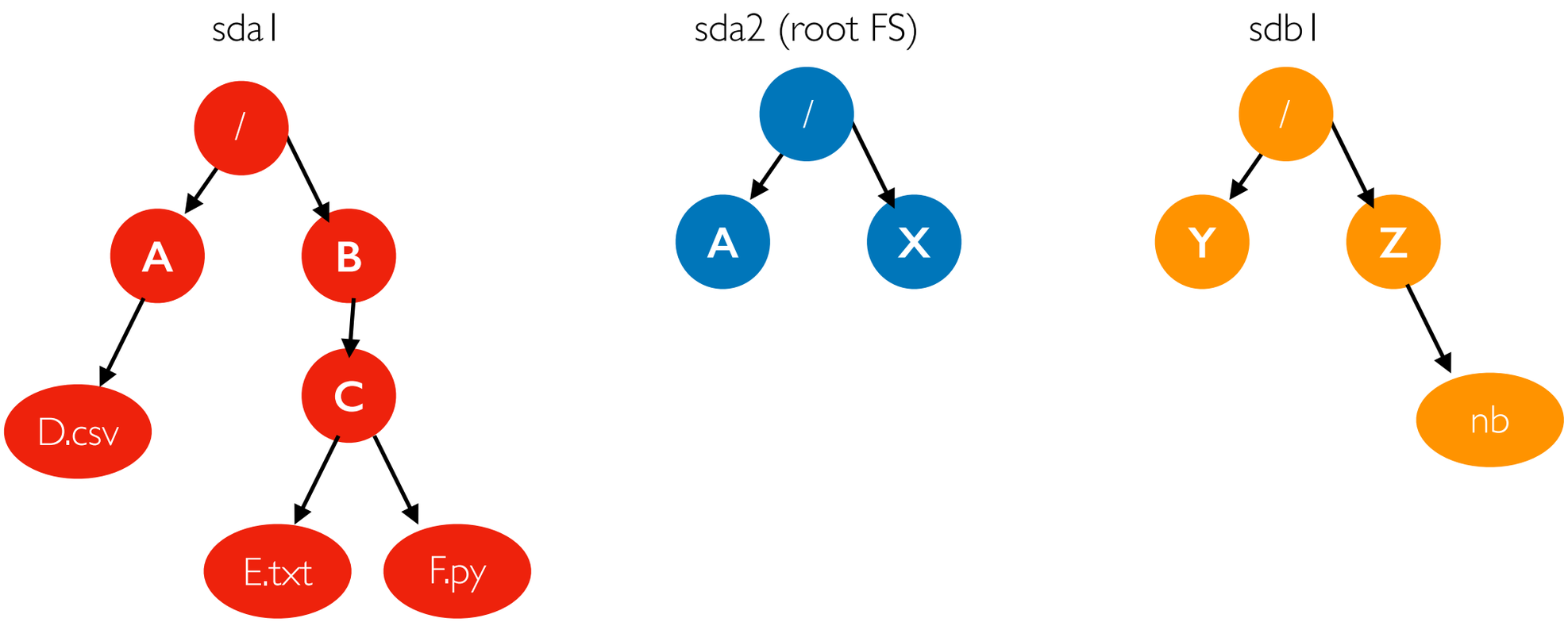
Multiple File Systems: Unix Approach

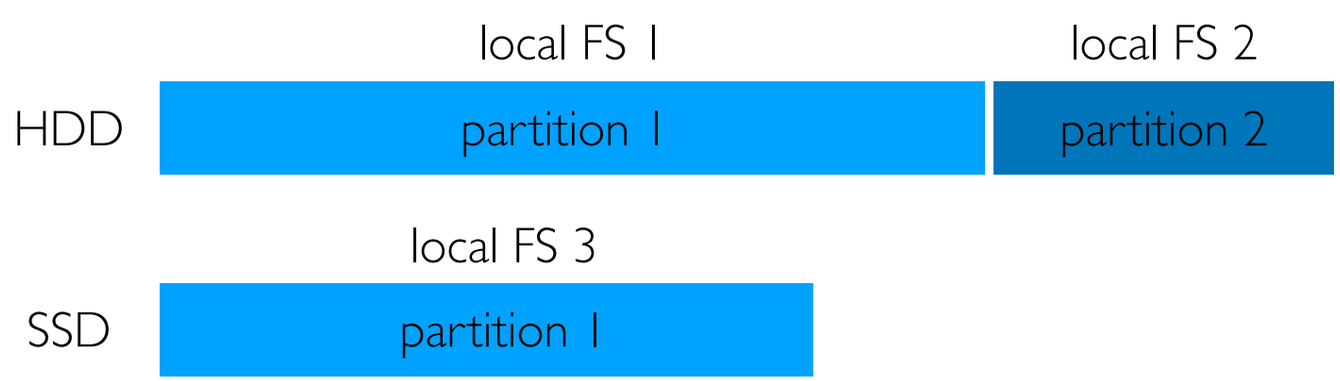
mount file systems over directories of other file systems to make one big tree

https://www.brit.co/fruit-salad-tree/

Multiple File Systems: Unix Approach

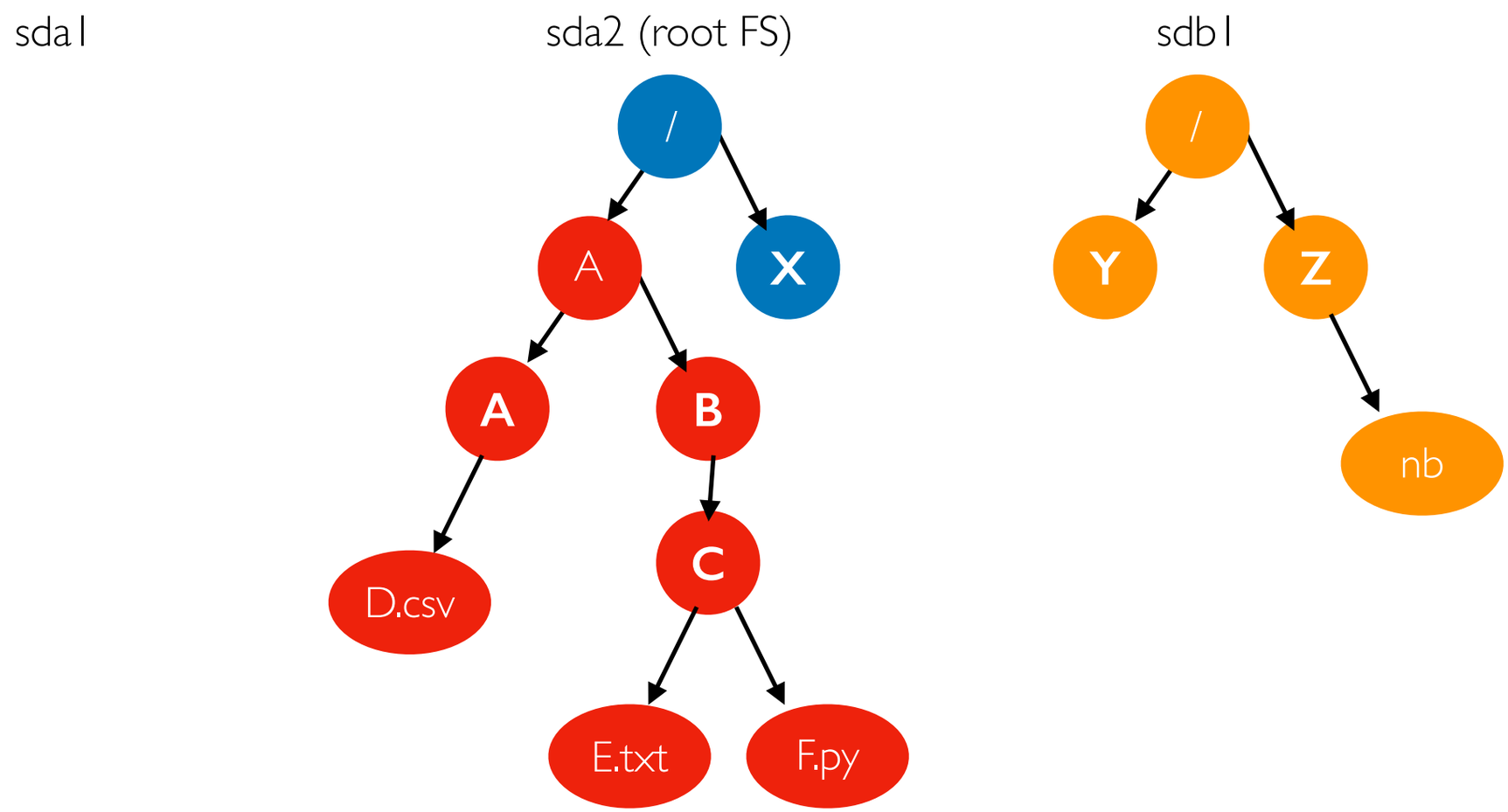
mount file systems over directories of other file systems to make one big tree





Multiple File Systems: Unix Approach

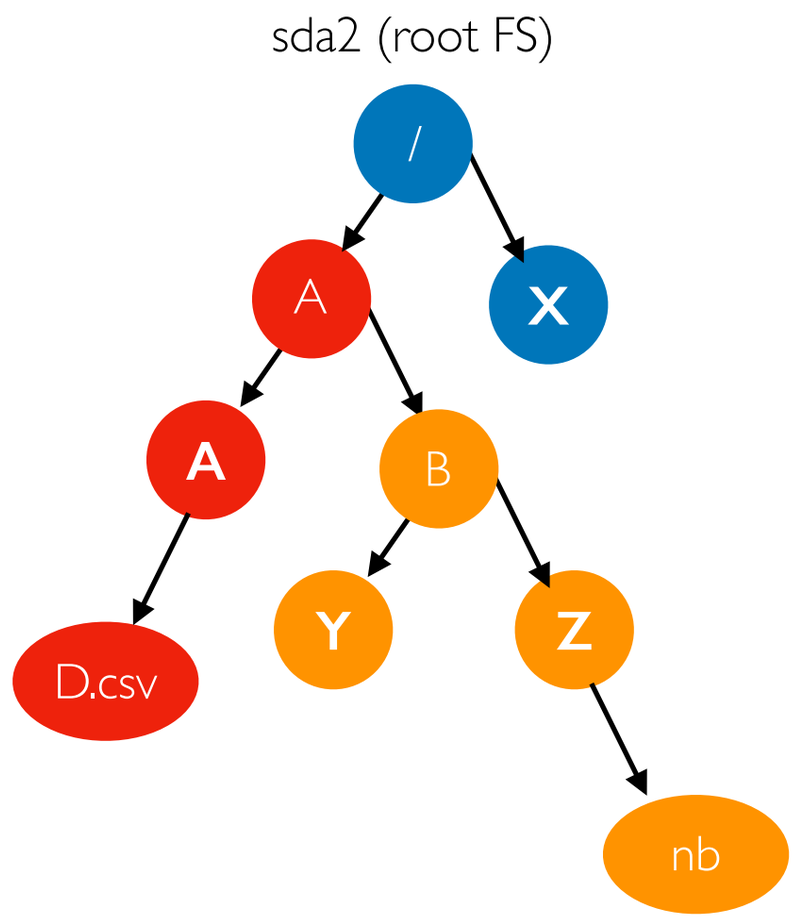
mount file systems over directories of other file systems to make one big tree




```
mount /dev/sda1 /A
```

Multiple File Systems: Unix Approach

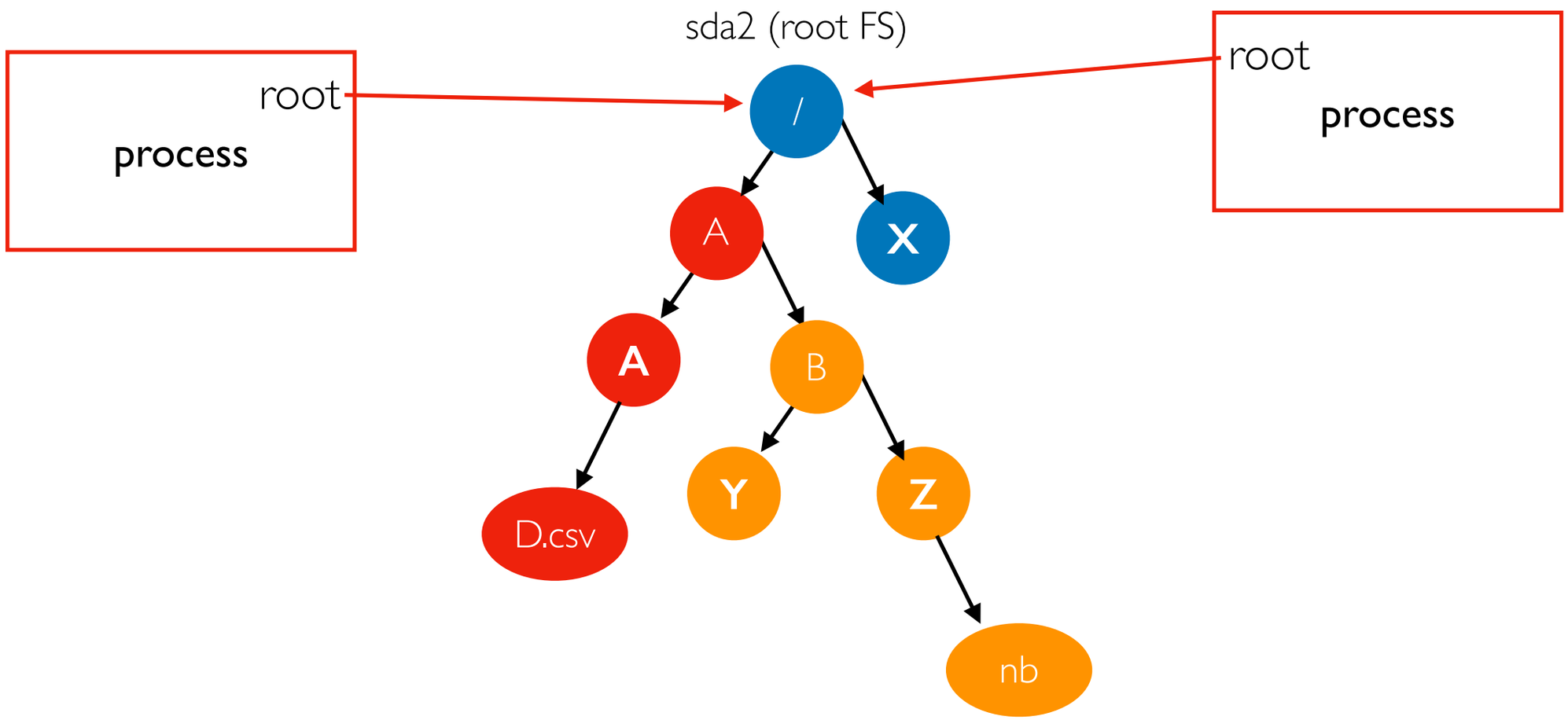
mount file systems over directories of other file systems to make one big tree



Note: each Docker container has it's own root file system and mount namespace

```
mount /dev/sda1 /A
mount /dev/sdb1 /A/B
```

Process's Root

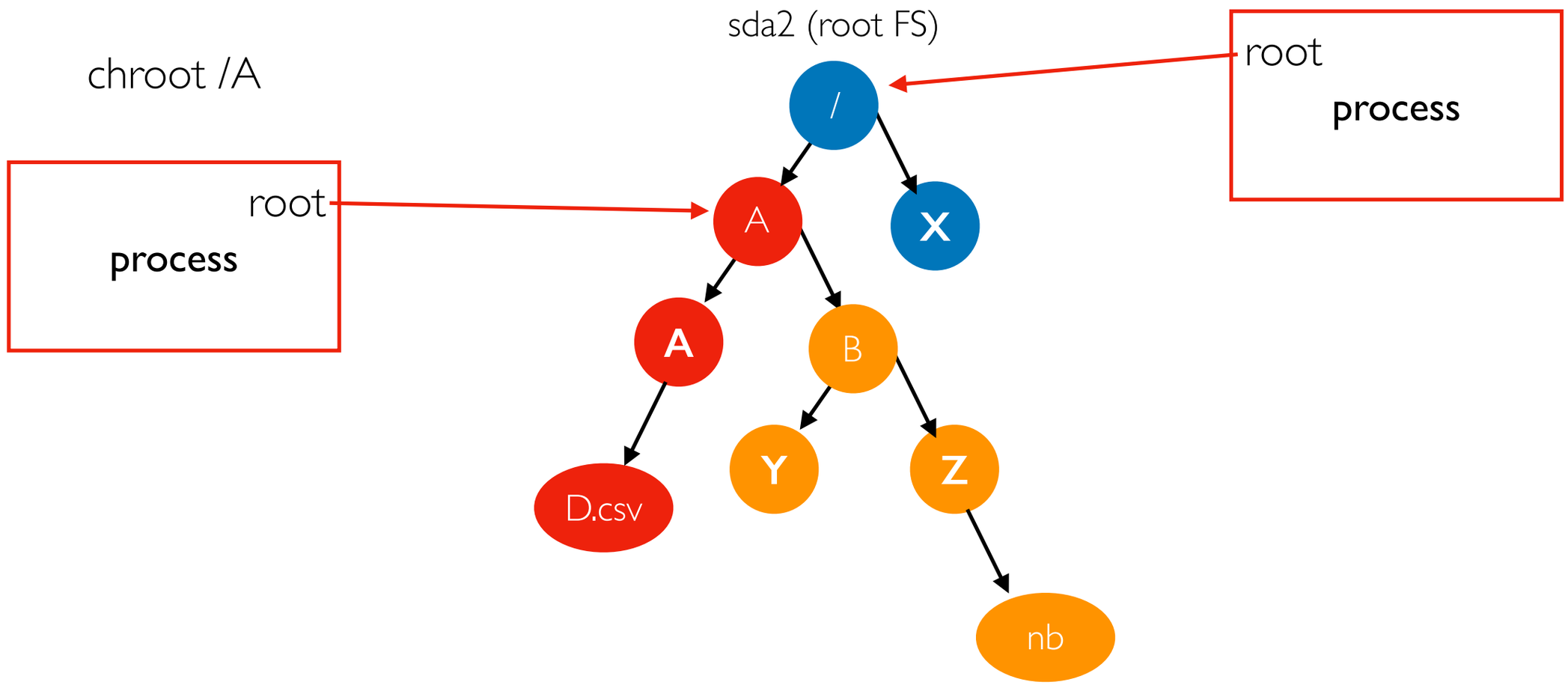


each process (and its descendants) uses a node in the unified FS as its root (usually the root of the unified tree)

chroot

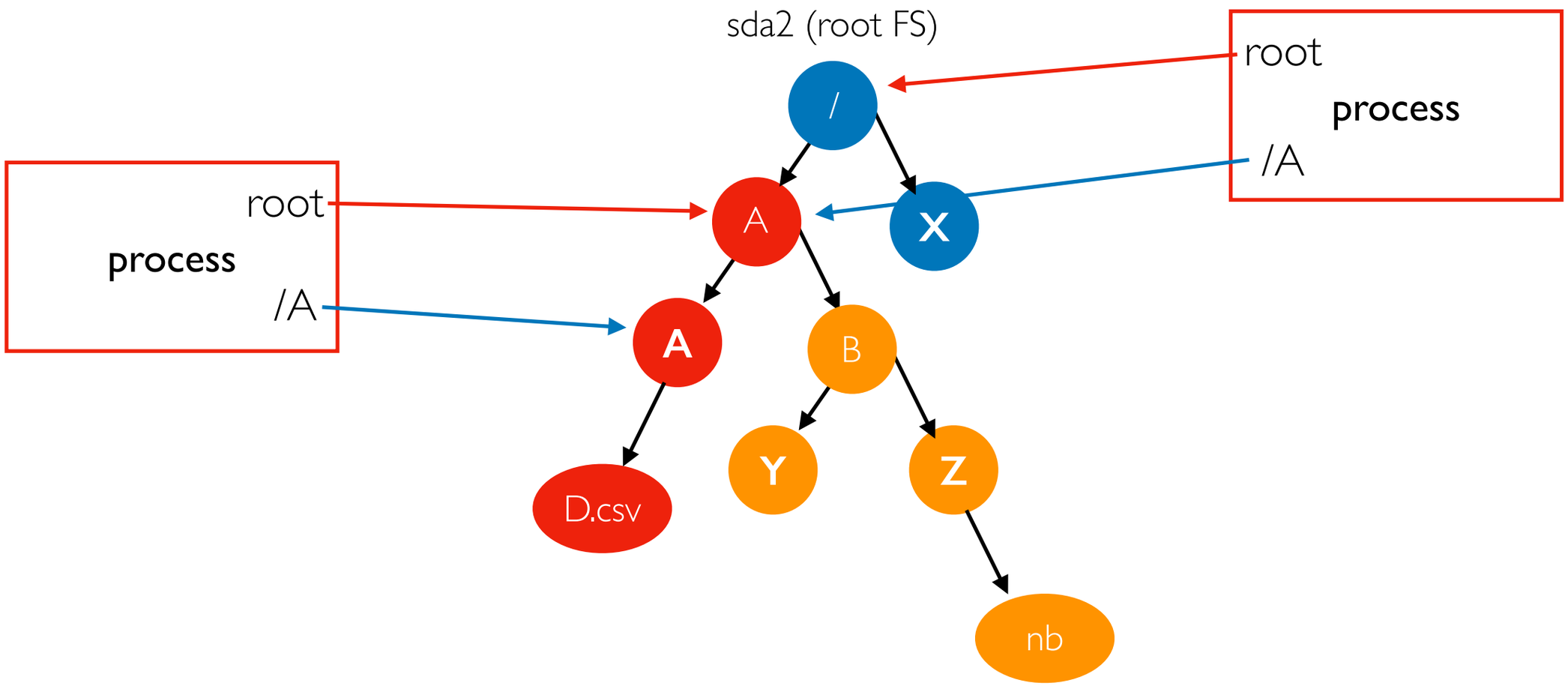
- `chroot("PATH") // change by path`
- `fchdir(fd); chroot(".") // change by FD`

careful, if an FD pointing to a dir stays open,
the process can use it to escape later!



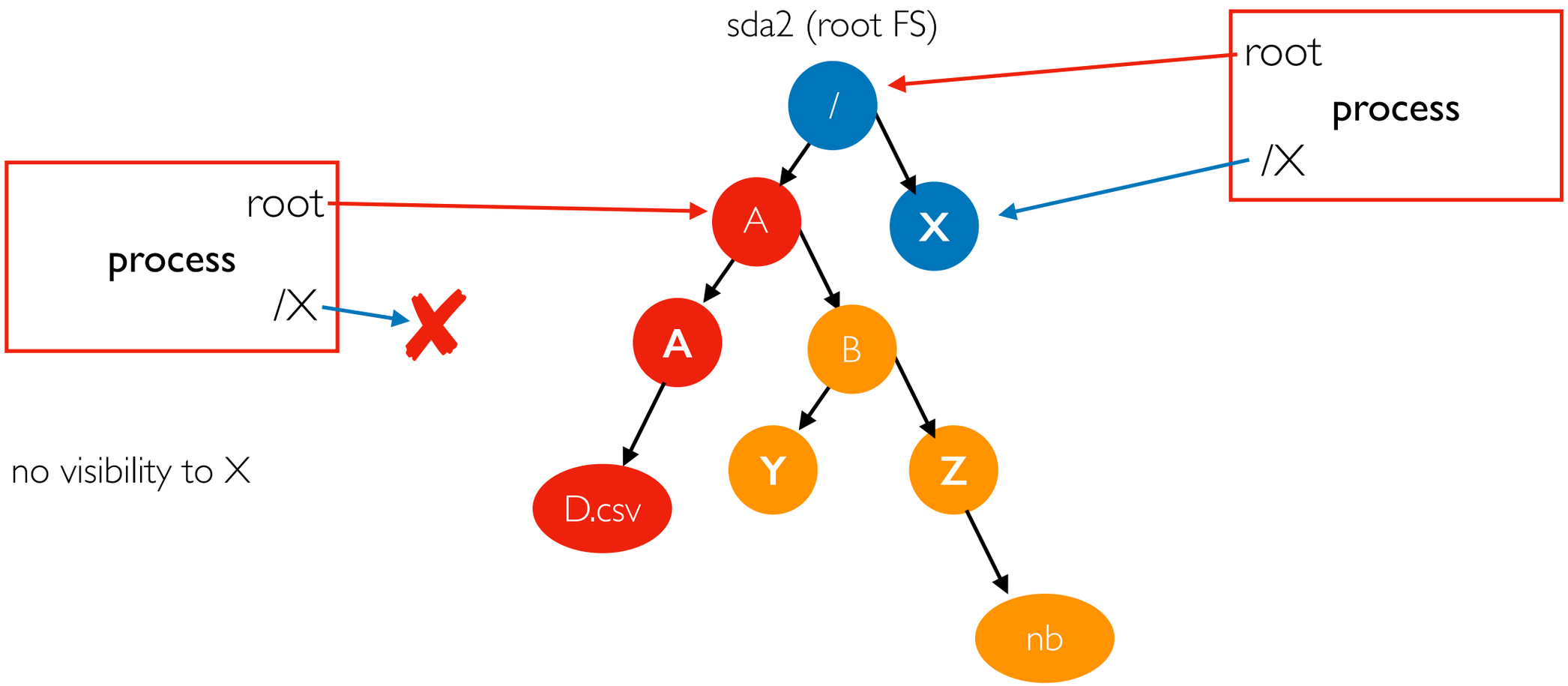
a process can change it's root to a non-root in the unified tree

chroot



the target of a path depends on your root

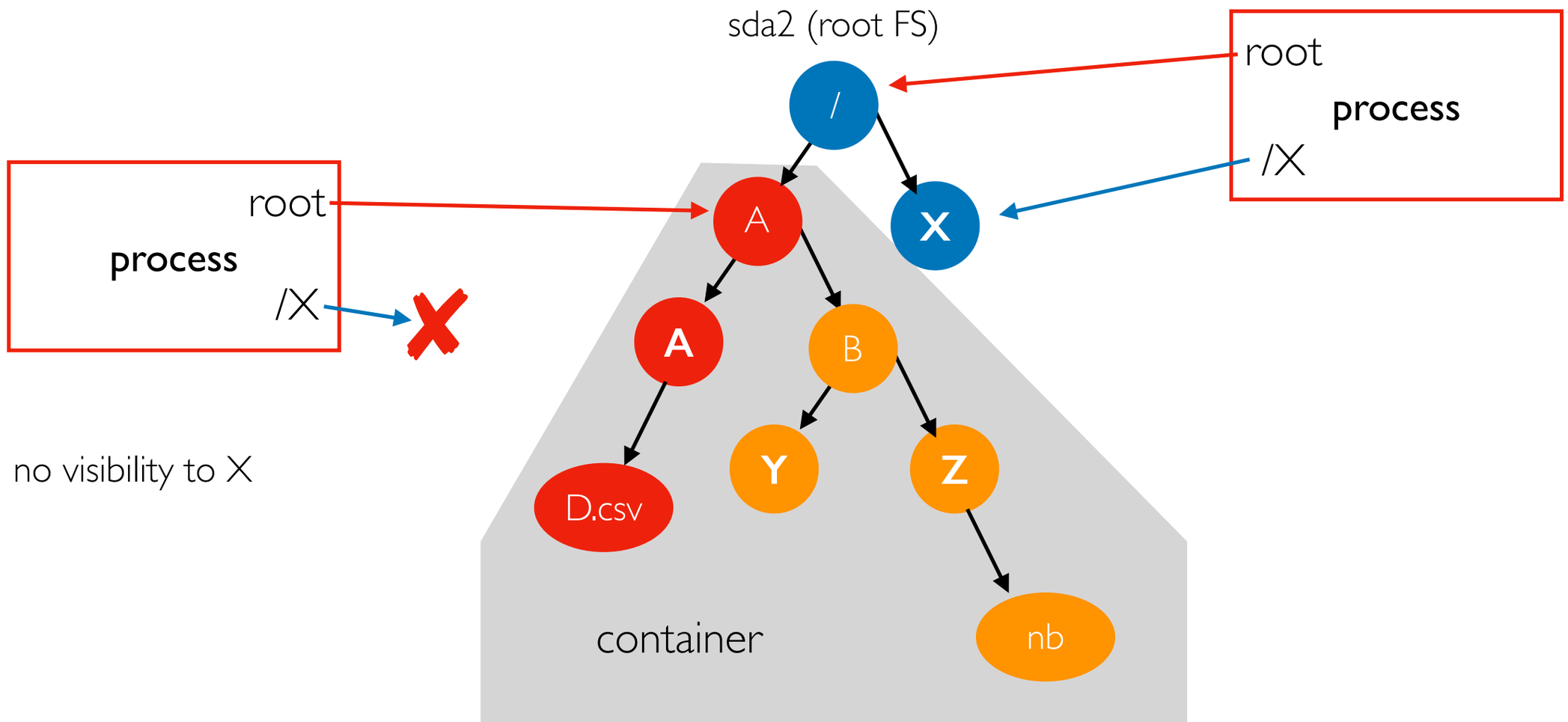
chroot



no visibility to X

useful for security

Containers



General container strategy: using a variety of techniques, fill a directory with everything the container needs (e.g., Ubuntu files, other dependencies, executables); then make that the root for the processes of the new container.

Outline

Refresher: Unified File System Layout (and chroot)

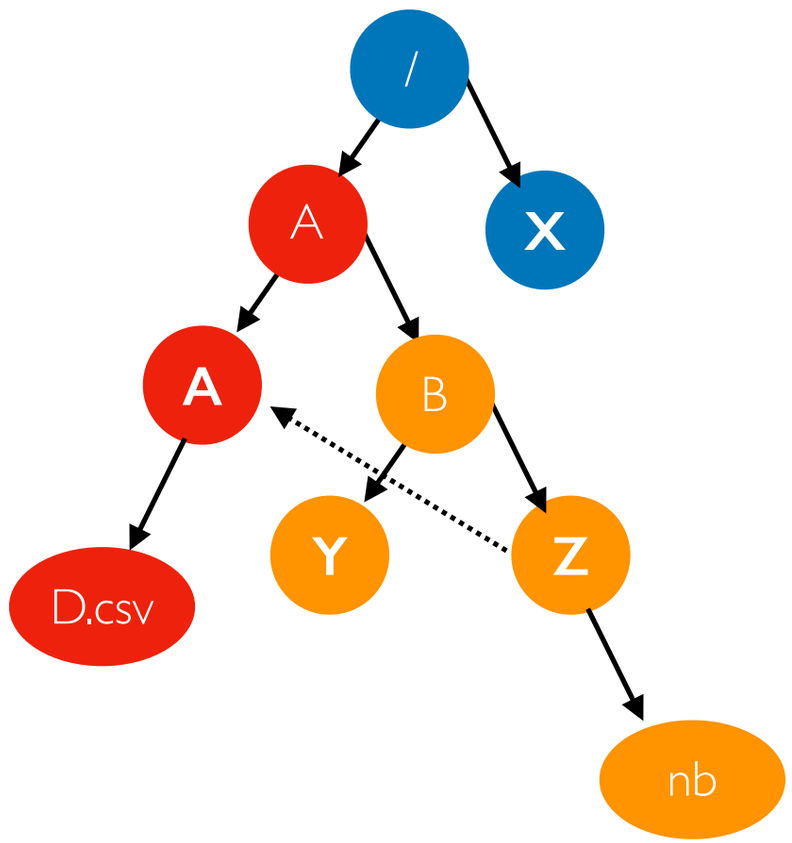
Bind Mounts

Union/Overlay File Systems

Docker Layers

OpenLambda

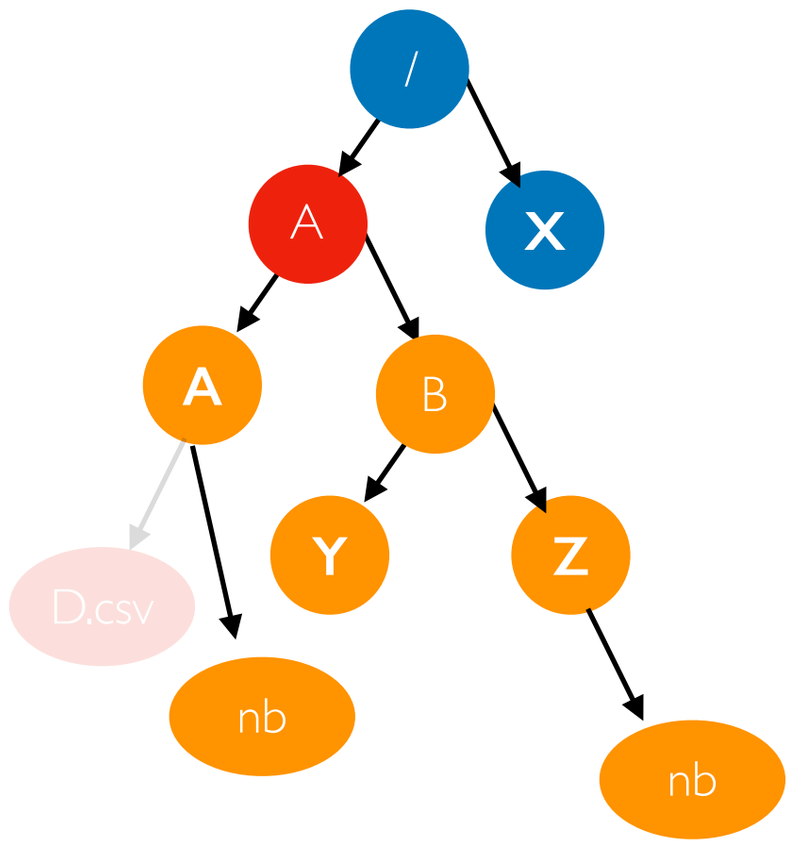
Bind Mounts




```
mount --bind /A/B/Z /A/A
```

this could optionally be read only at /A/A

Bind Mounts

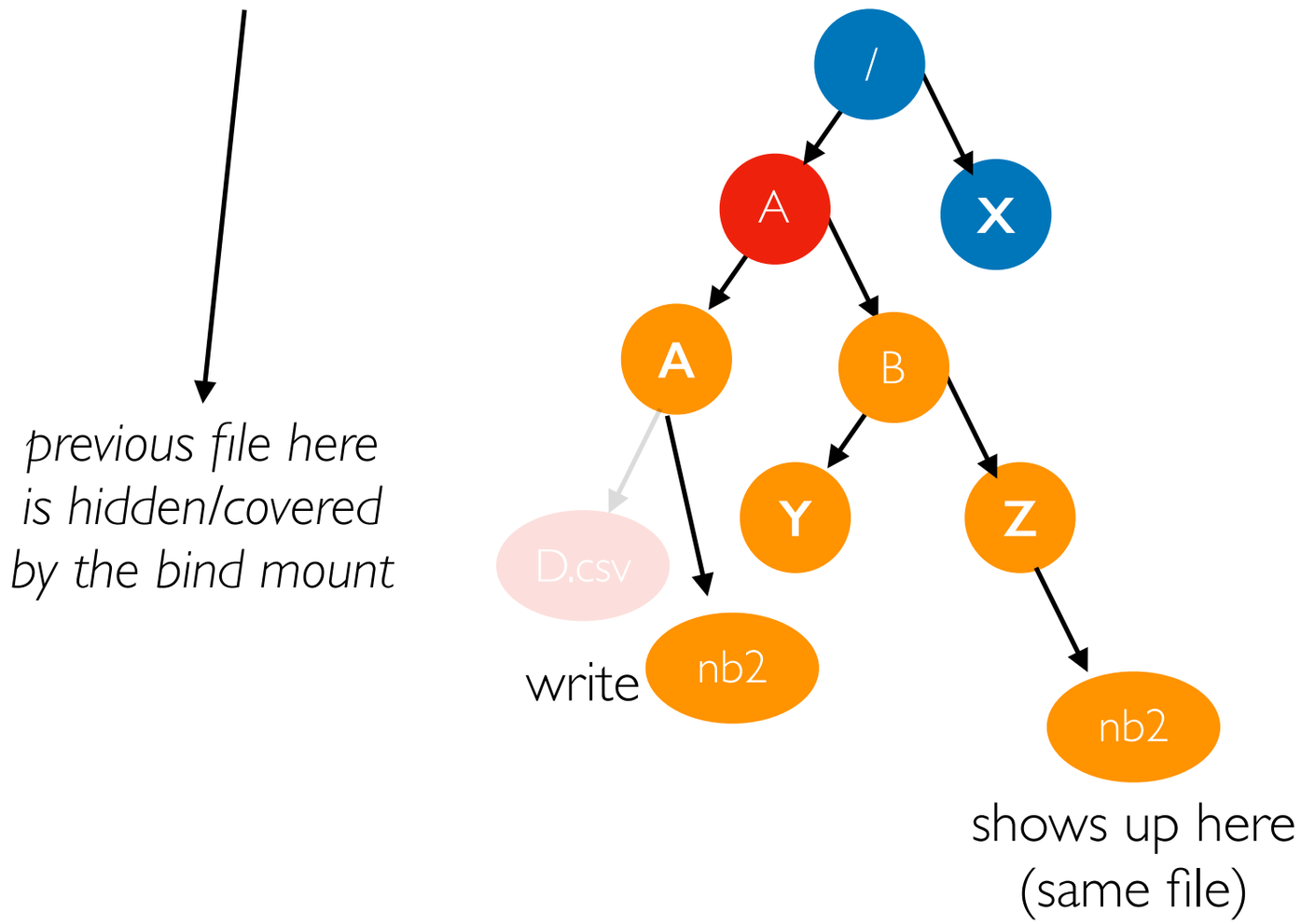



```
mount --bind /A/B/Z /A/A
```

this could optionally be read only at /A/A

Bind Mounts

union file systems are a slower but more flexible alternative that gives a "merged" view of multiple directories



this could optionally be read only at /A/A

Outline

Refresher: Unified File System Layout (and chroot)

Bind Mounts

Union/Overlay File Systems

Docker Layers

OpenLambda

Slacker

Background

- internship work done for Tintri (FAST paper, U.S. Patent 10,430,378)
- Docker used AUFS (another union FS) by default at the time
- overlayfs is similar (current default)

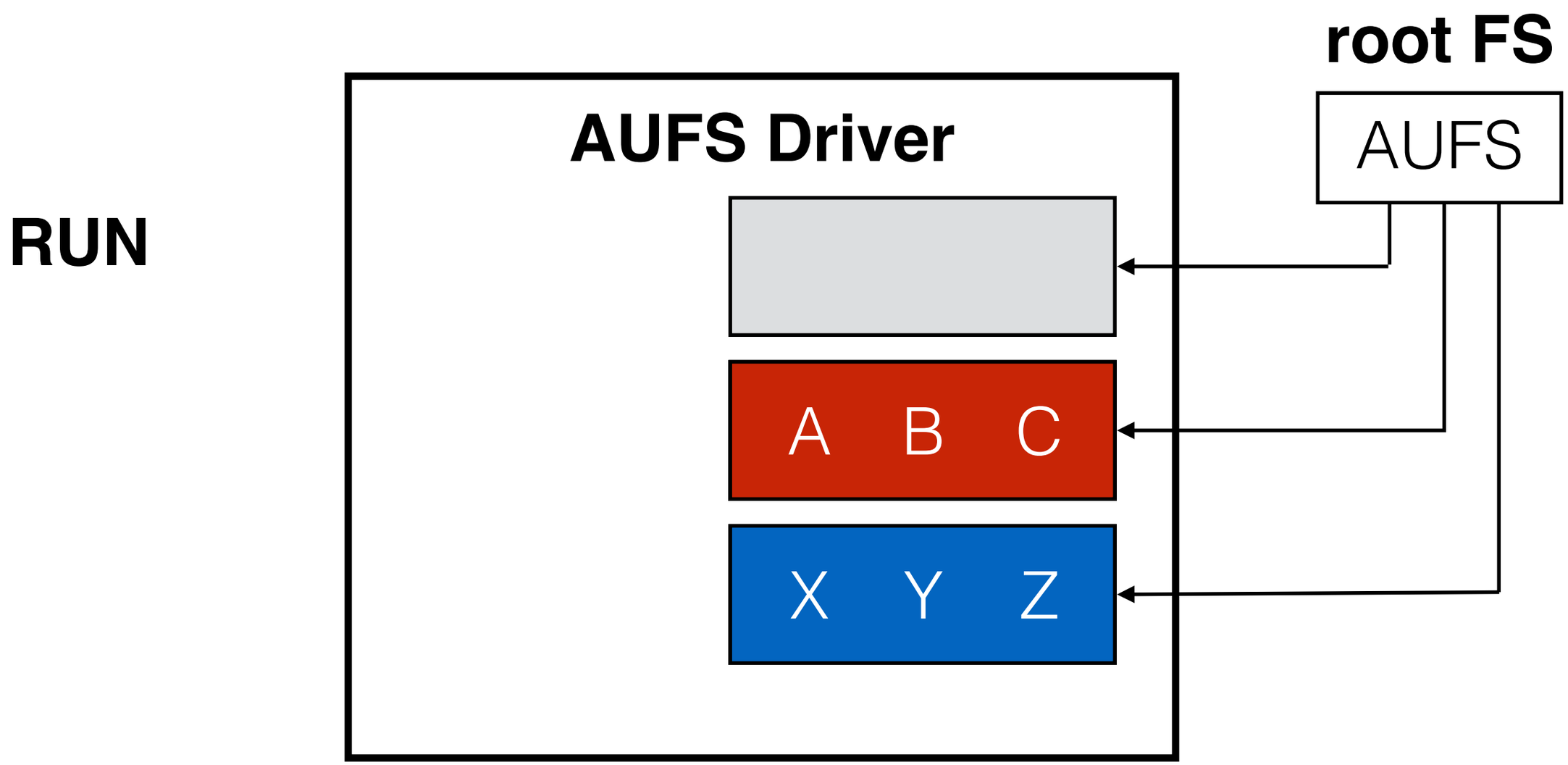
Slacker: Fast Distribution with Lazy Docker Containers

Tyler Harter, *University of Wisconsin—Madison*; Brandon Salmon and Rose Liu, *Tintri;*
Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau, *University of Wisconsin—Madison*

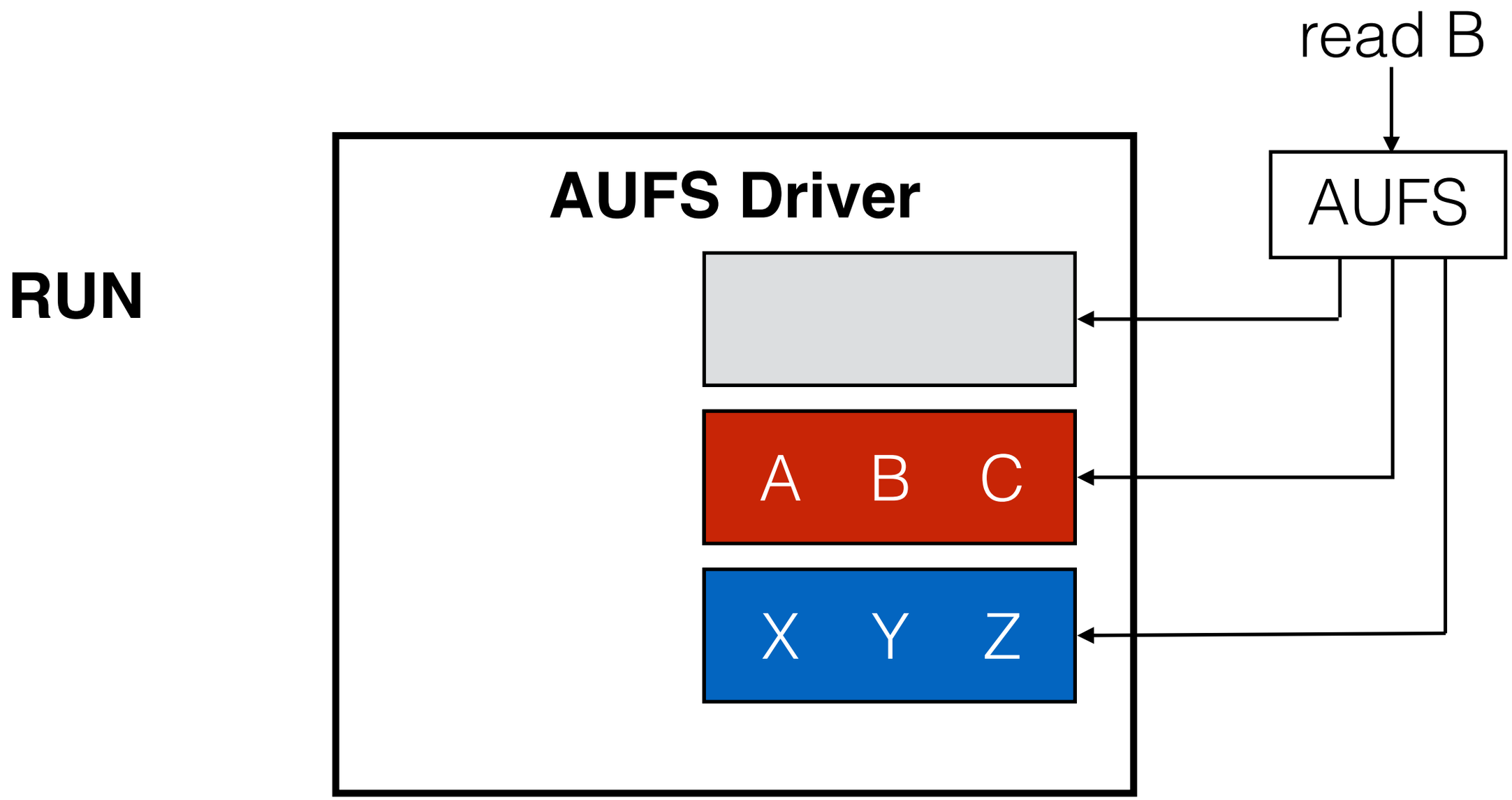
https://www.usenix.org/conference/fast16/technical-sessions/presentation/harter

https://www.usenix.org/system/files/conference/fast16/fast16-papers-harter.pdf

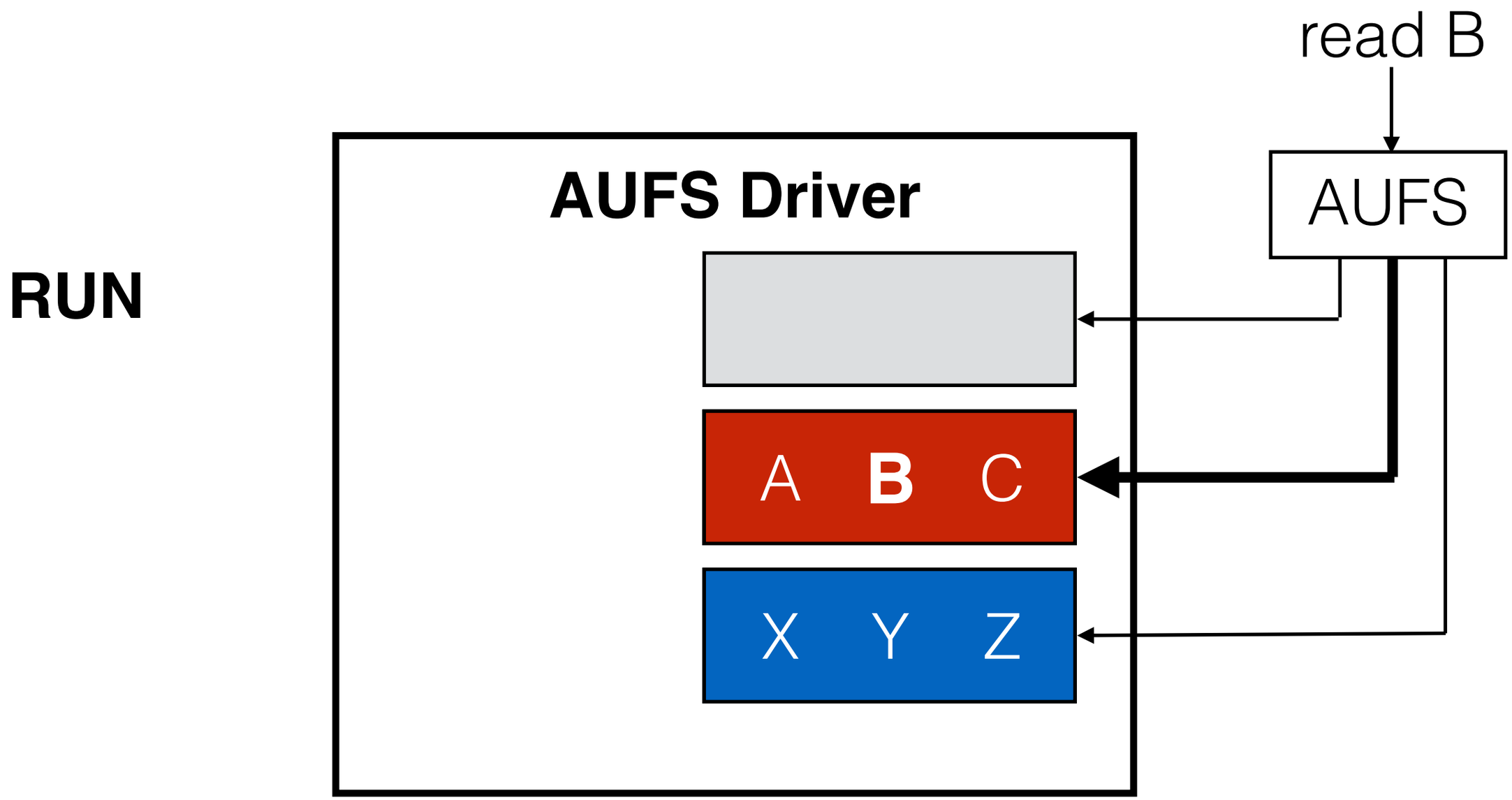
Another Union File System



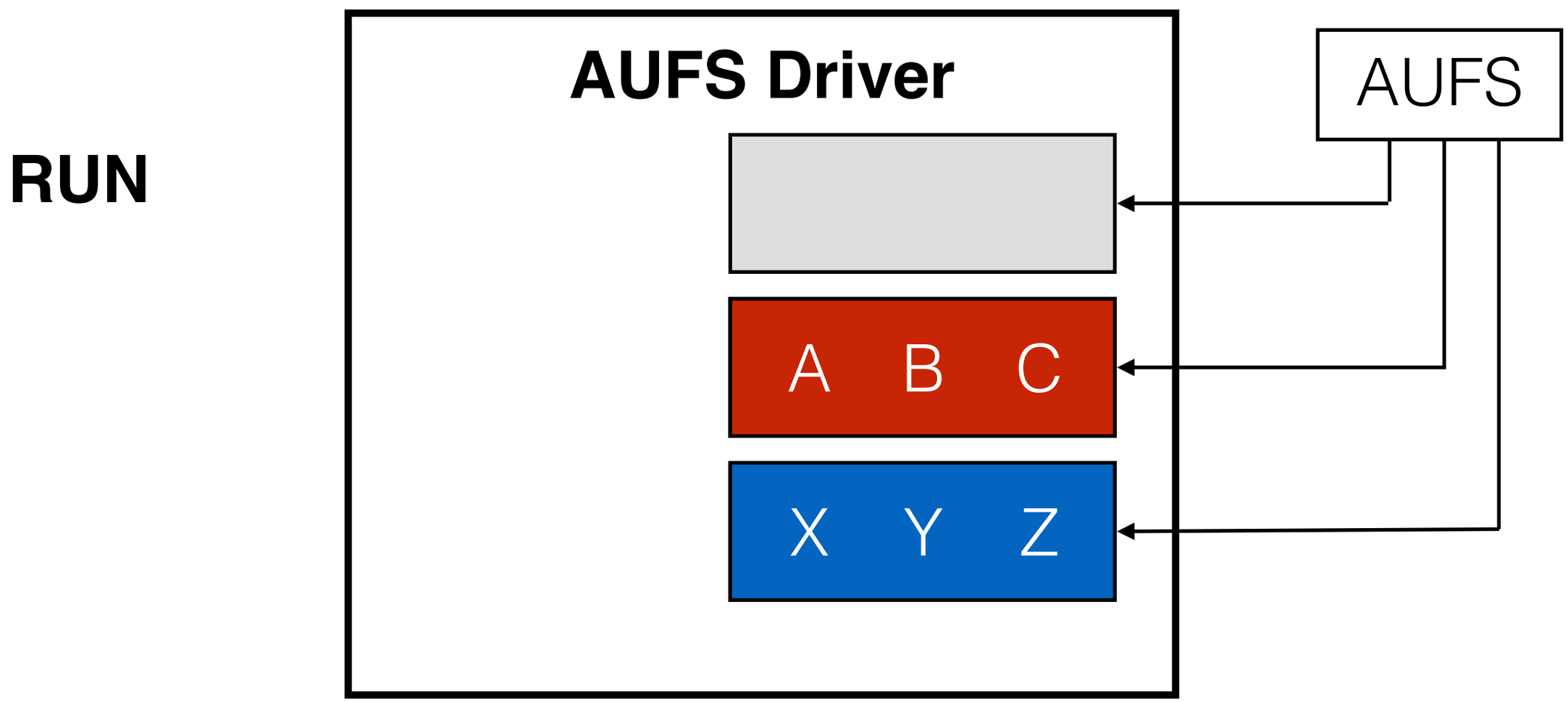
Another Union File System



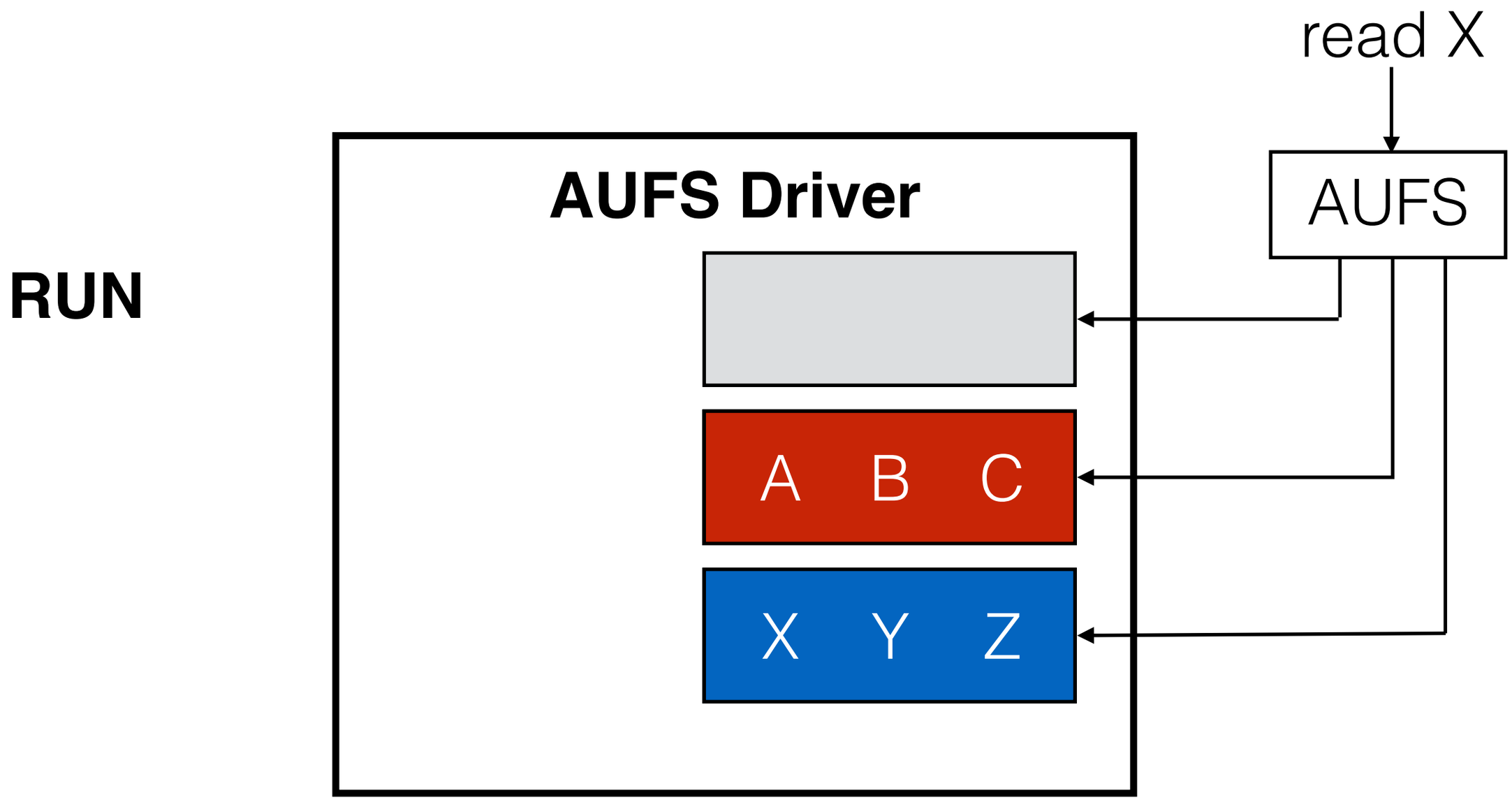
Another Union File System



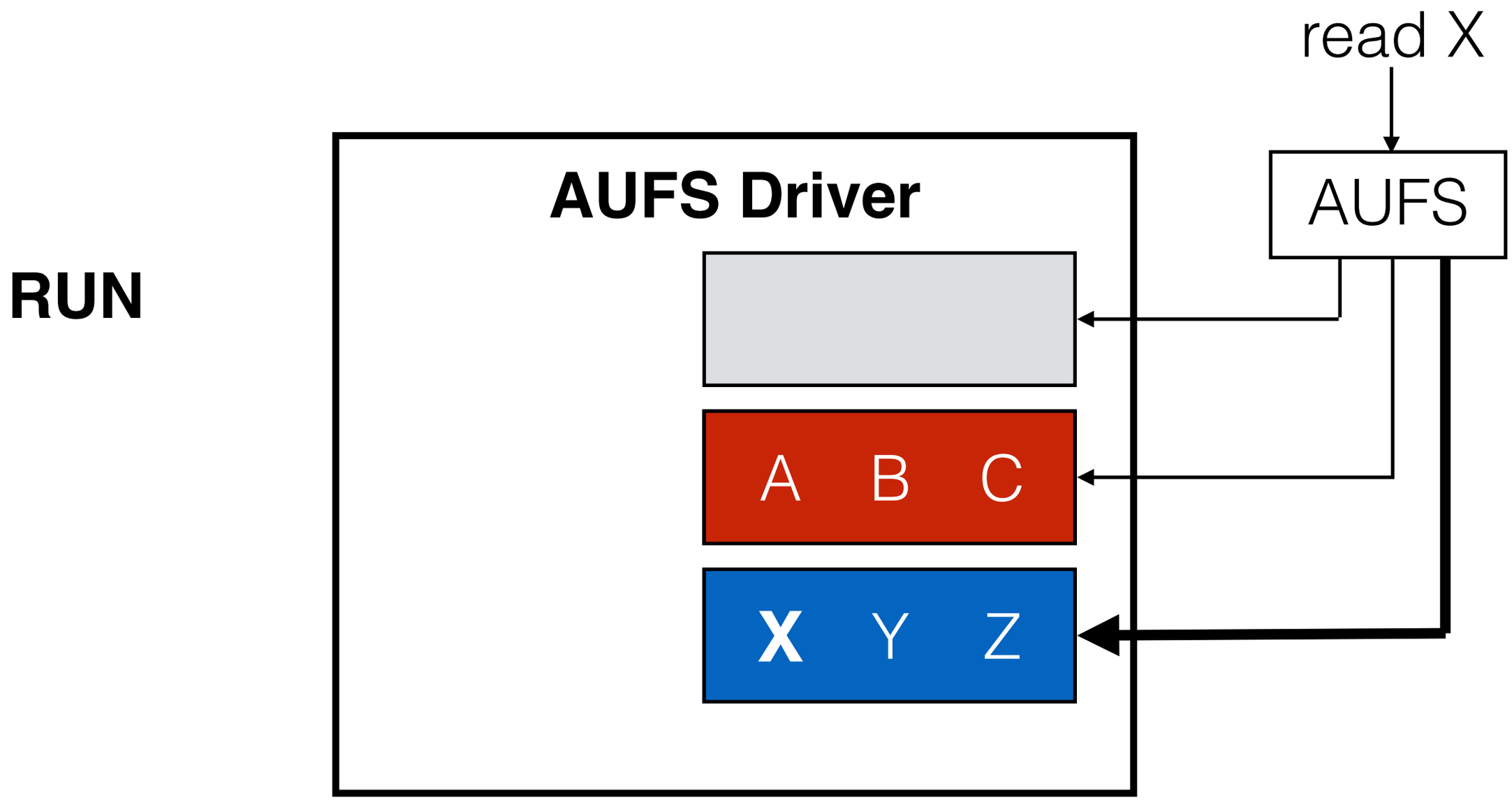
Another Union File System



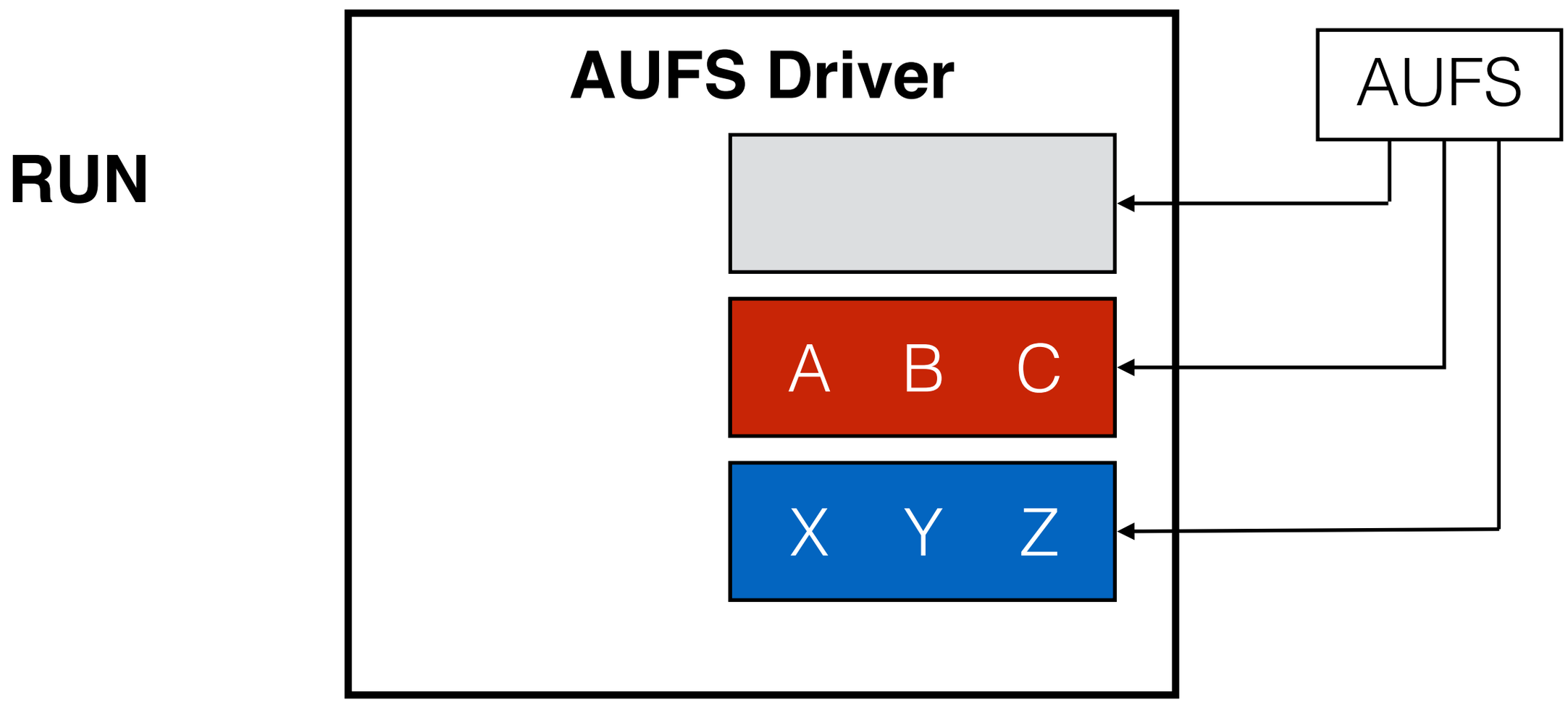
Another Union File System



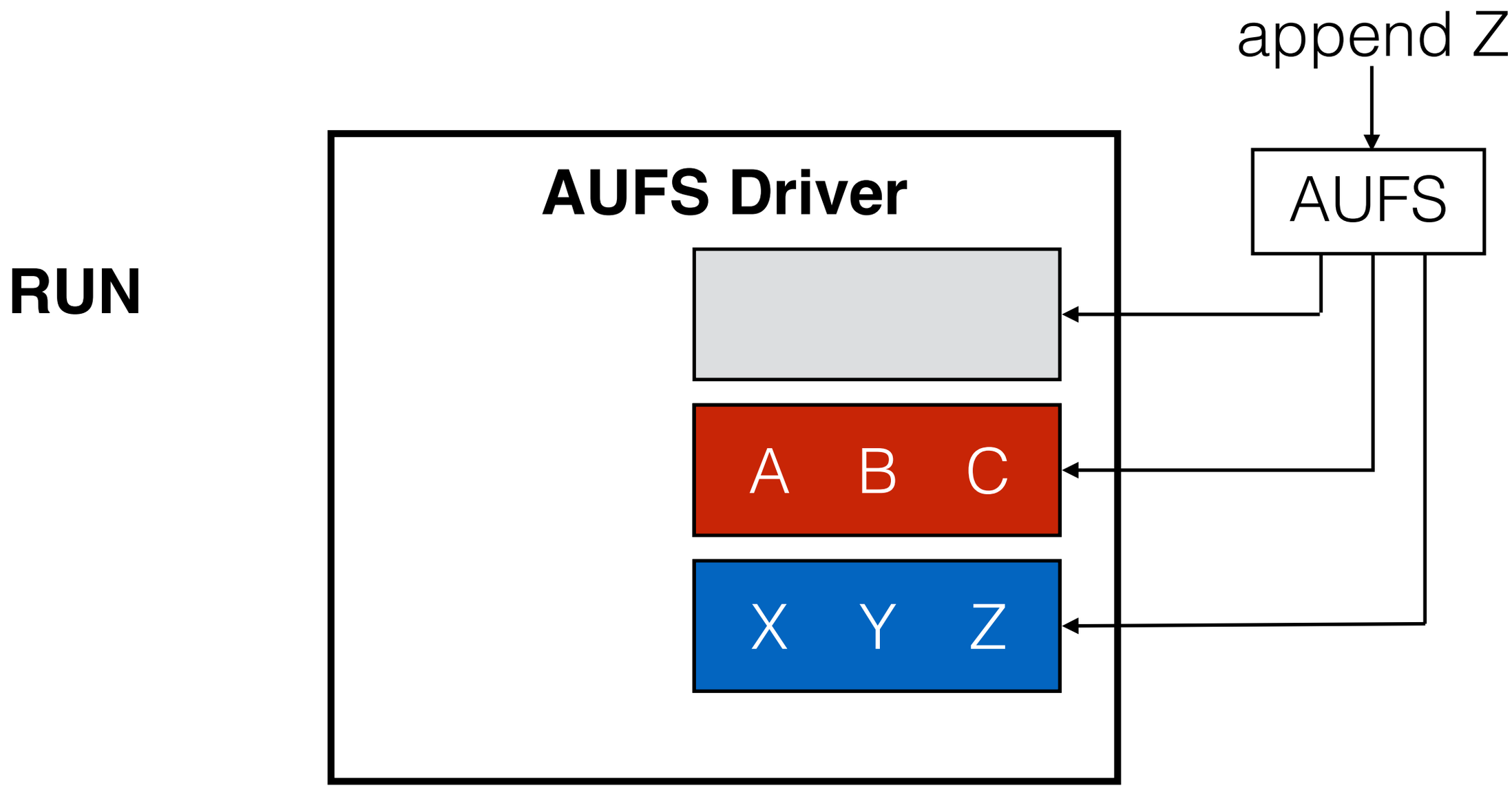
Another Union File System



Another Union File System

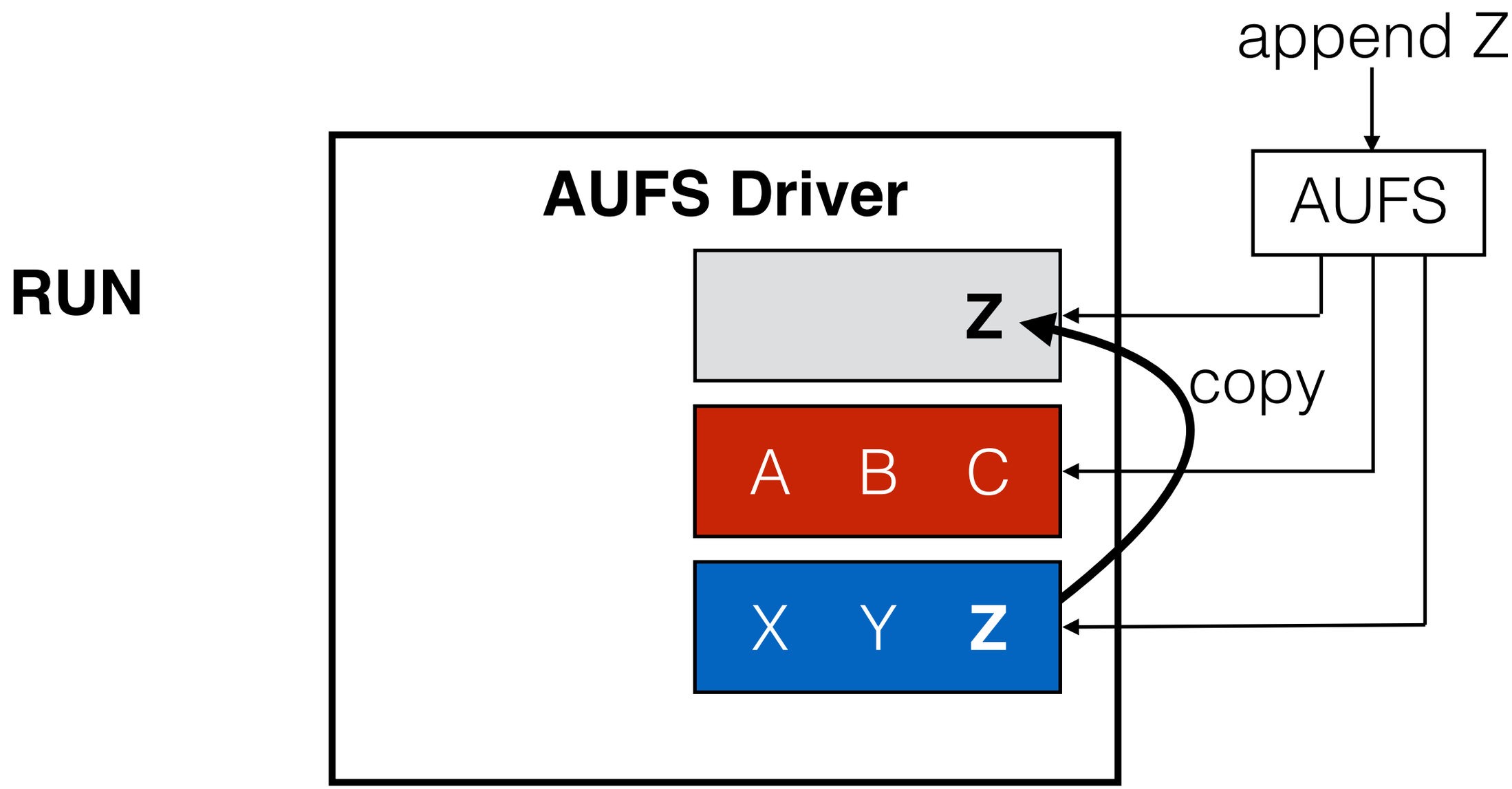


Another Union File System

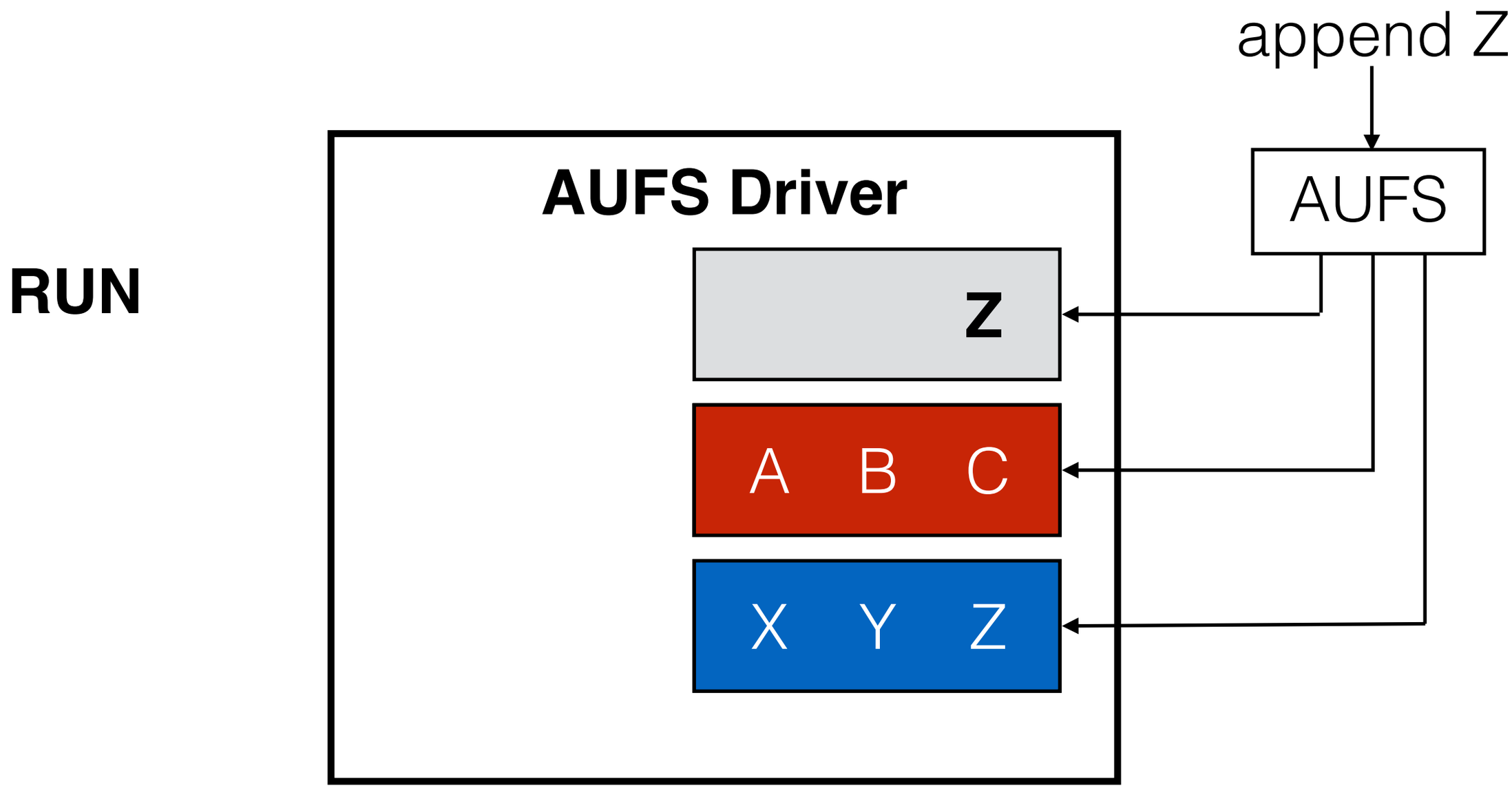


Another Union File System

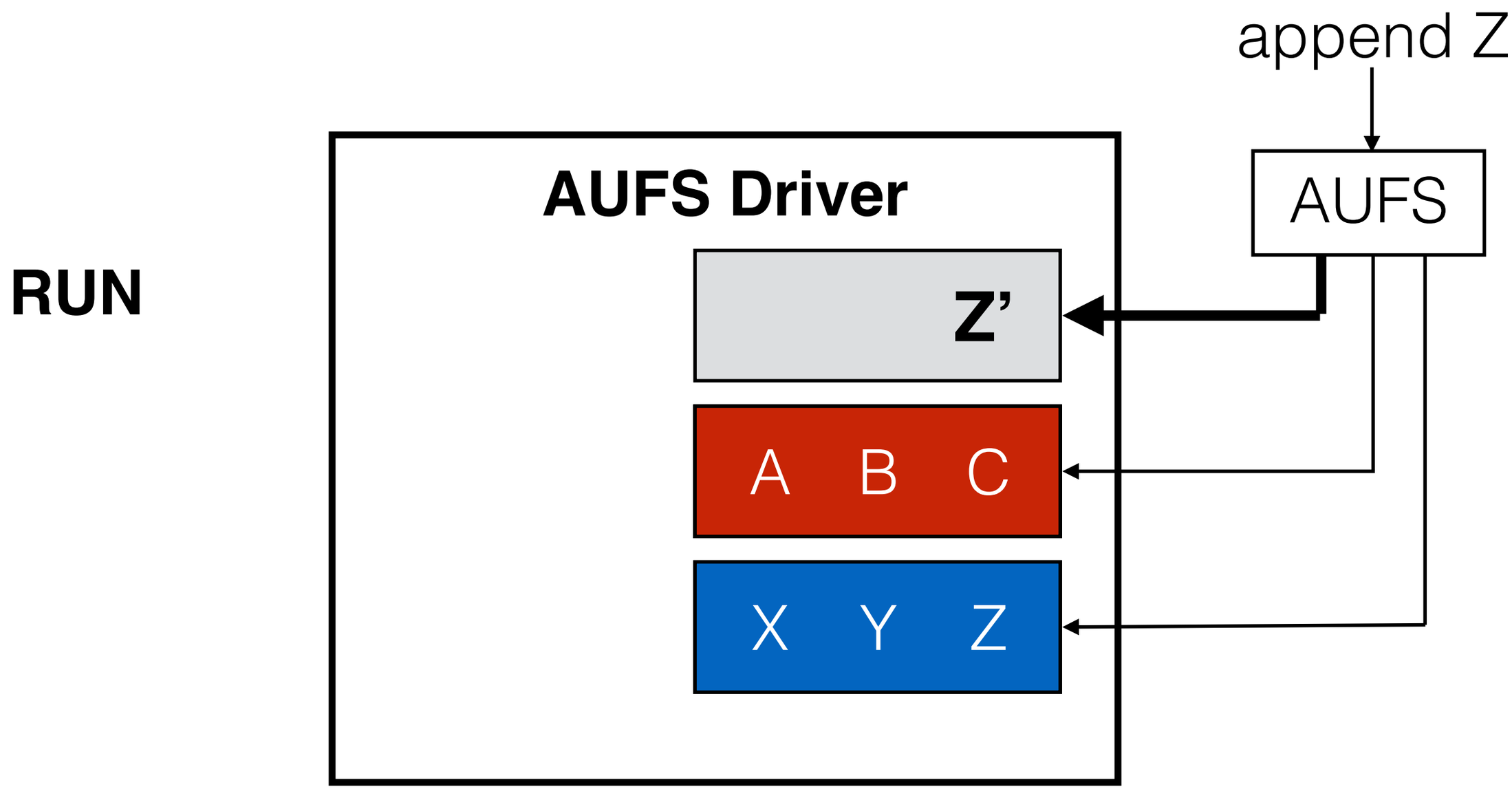
might be very slow if Z is large and append is small!



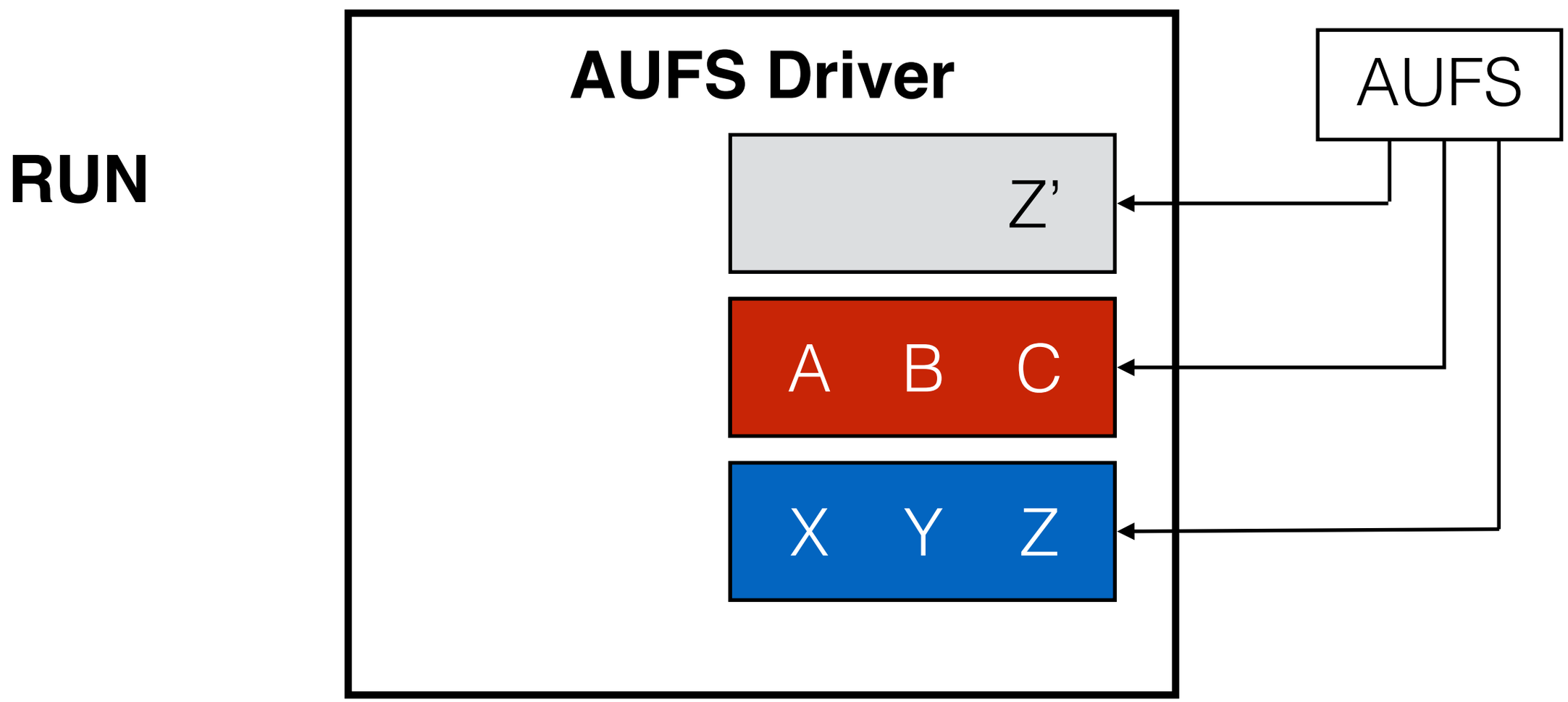
Another Union File System



Another Union File System



Another Union File System



AUFS Performance

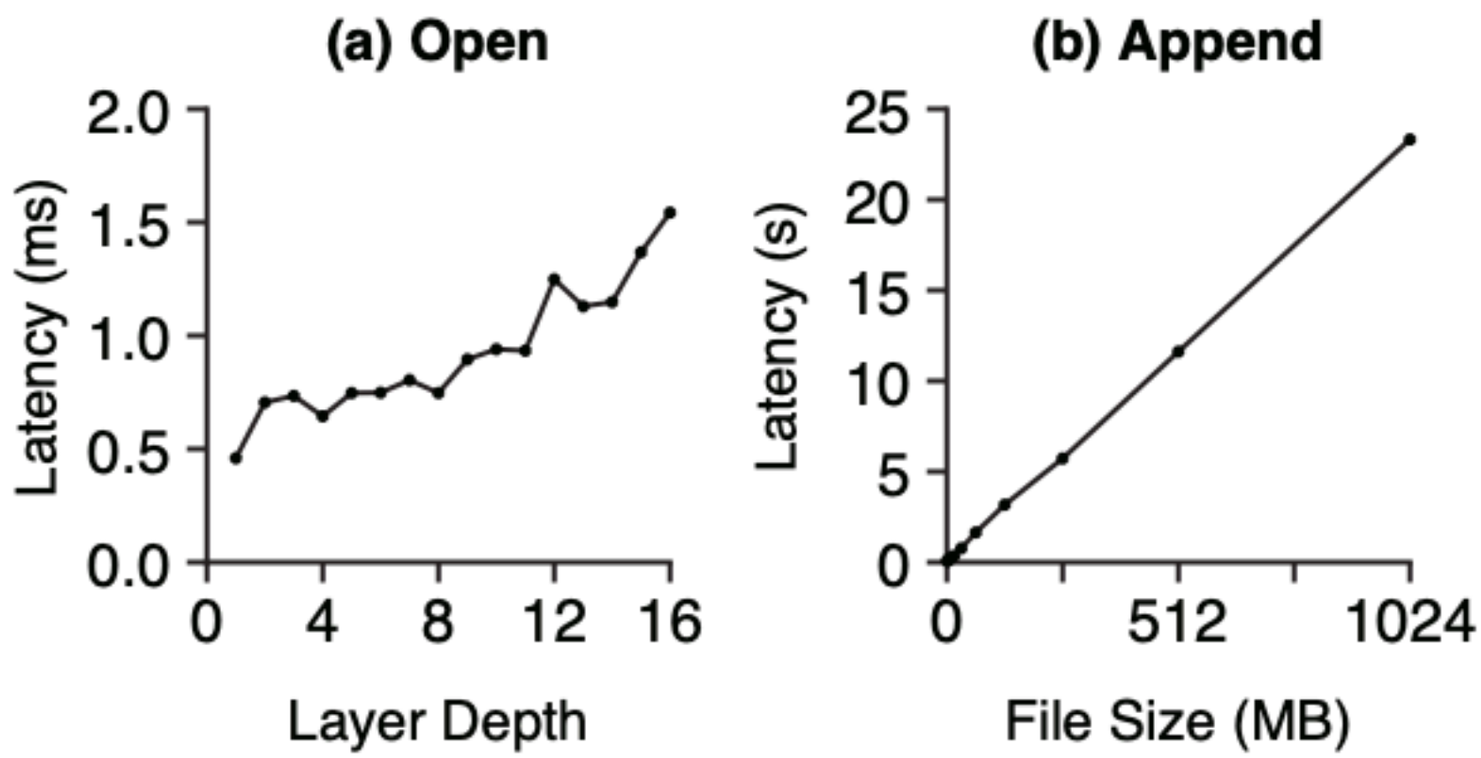


Figure 11: **AUFS Performance.** *Left: the latency of the* `open` *system call is shown as a function of the layer depth of the file. Right: the latency of a one-byte append is shown as a function of the size of the file that receives the write.*

Ought to evaluate overlayfs too...

Outline

Refresher: Unified File System Layout (and chroot)

Bind Mounts

Union/Overlay File Systems

Docker Layers

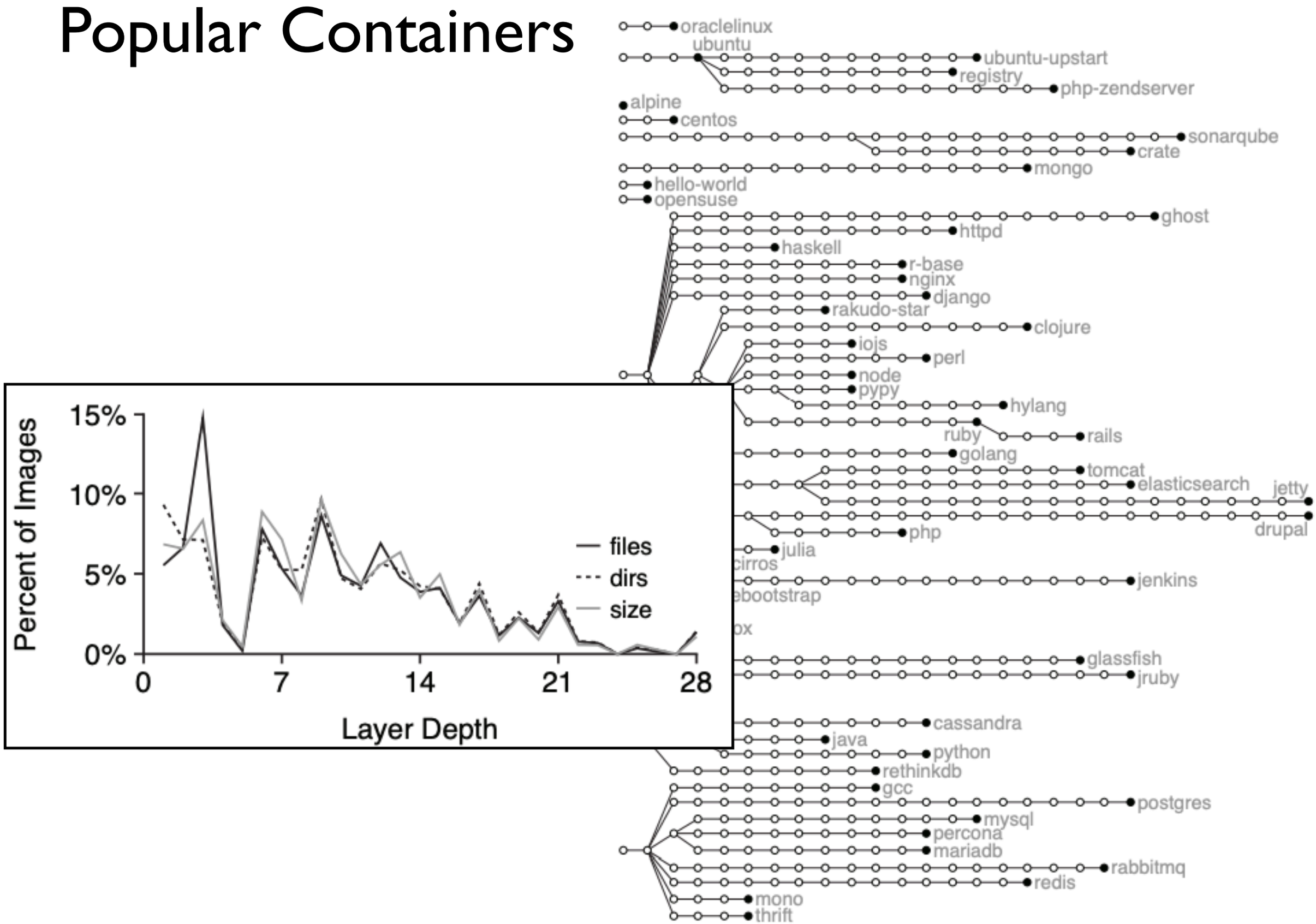
OpenLambda

Docker Layers

Docker images and containers consist of a collection of layers, which are directories where stuff has been installed/initialized.

When a running container needs a complete view of an FS, these are merged together with AUFS, OverlayFS, or simila.

Popular Containers



Outline

Refresher: Unified File System Layout (and chroot)

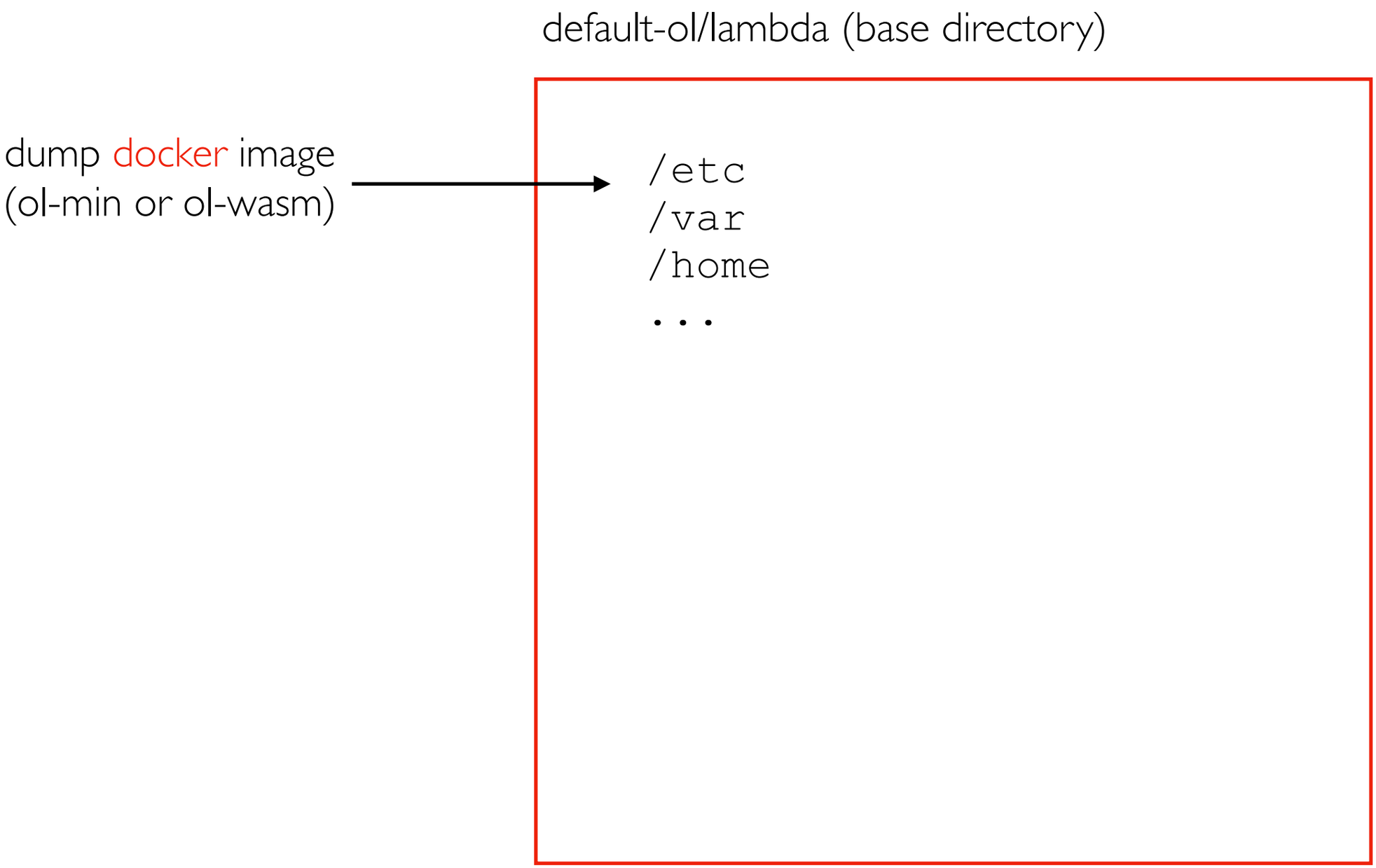
Bind Mounts

Union/Overlay File Systems

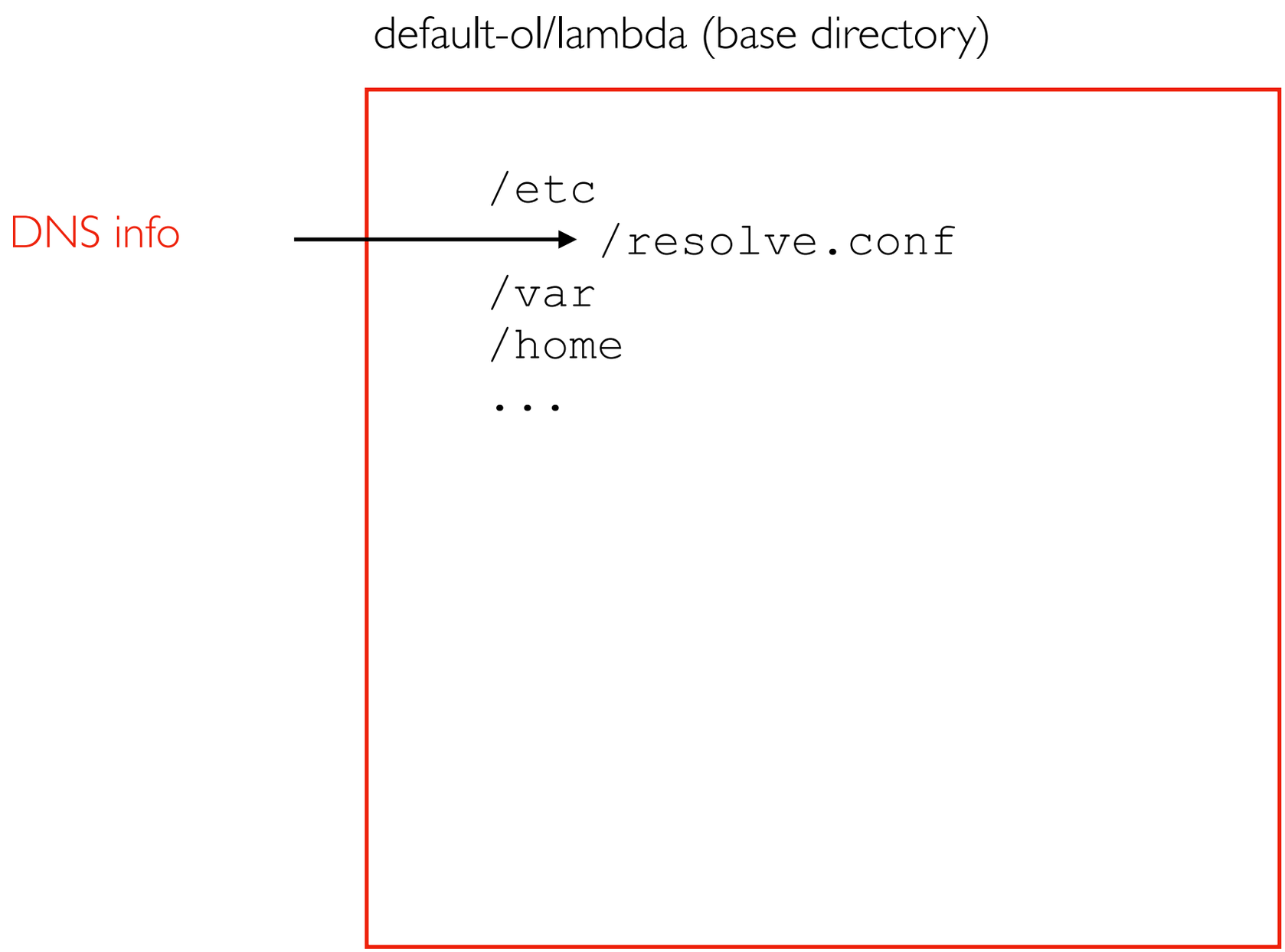
Docker Layers

OpenLambda

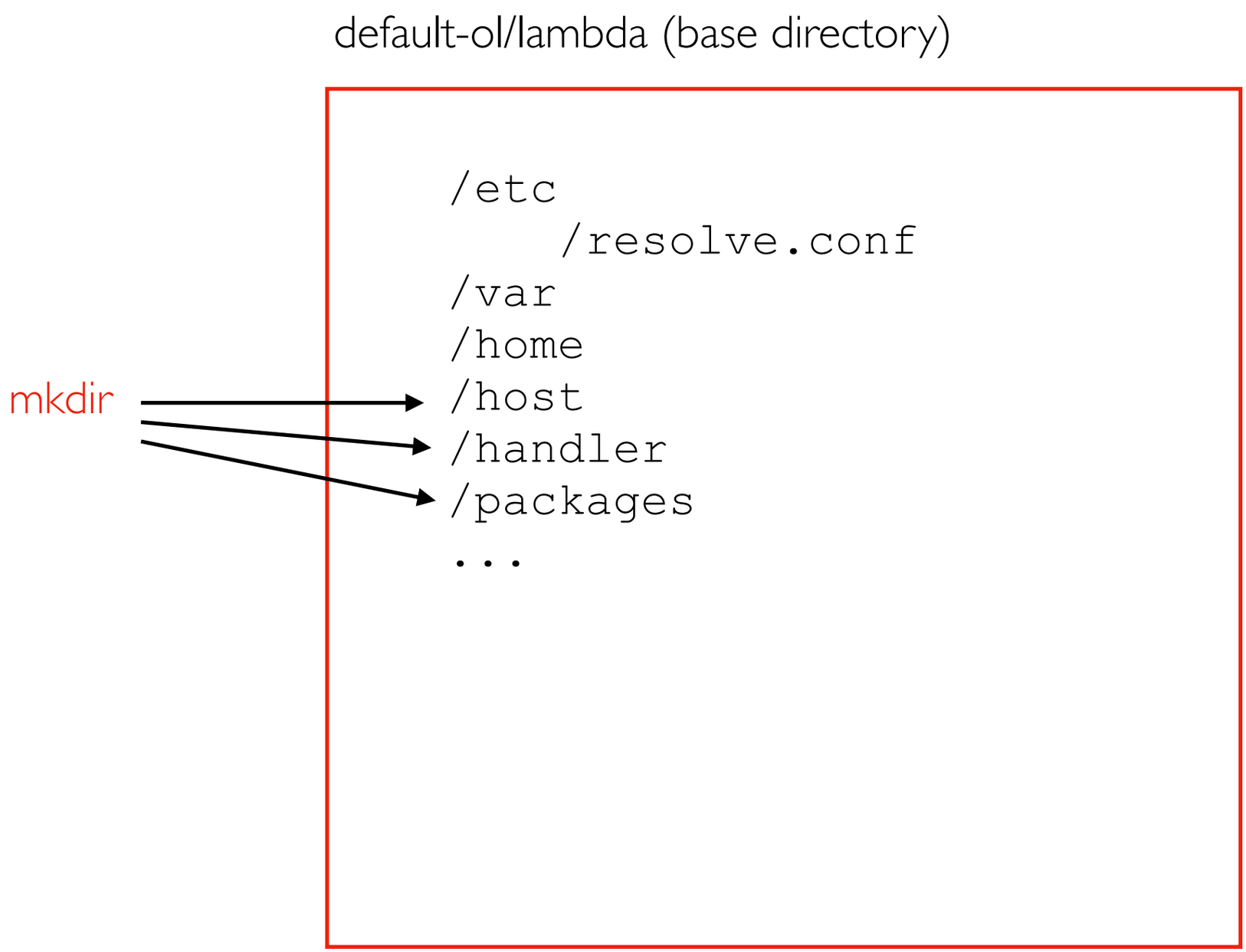
OpenLambda Init Worker



OpenLambda Init Worker

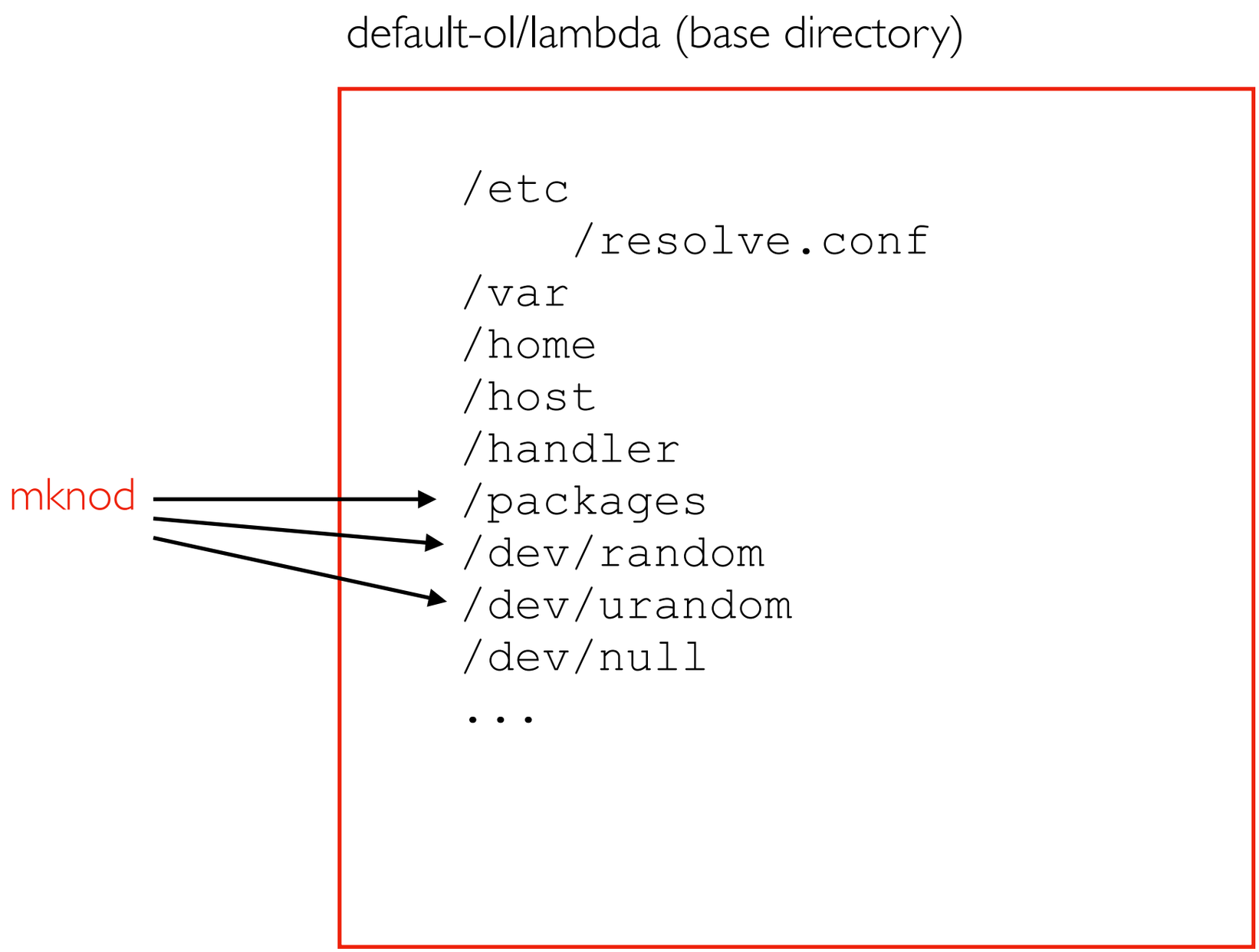


OpenLambda Init Worker



- /packages is shared read only into ALL containers
- /host is private, and the only writeable directory on a SOCK container

OpenLambda Init Worker



OpenLambda Init Worker

default-ol/lambda (base directory)

```
/etc
    /resolve.conf
/var
/home
/host
/handler
/packages
/dev/random
/dev/urandom
/dev/null
...
```

procfs is not mounted

OpenLambda Init SOCK Sandbox

default-ol/lambda (base directory)

```
/etc
    /resolve.conf
/var
/home
/host
/handler
/packages
/dev/random
/dev/urandom
/dev/null
...
```

bind (read only) →

```
/etc
    /resolve.conf
/var
/home
/host
/handler
/packages
/dev/random
/dev/urandom
/dev/null
...
```

OpenLambda Init SOCK Sandbox

directory with lambda code

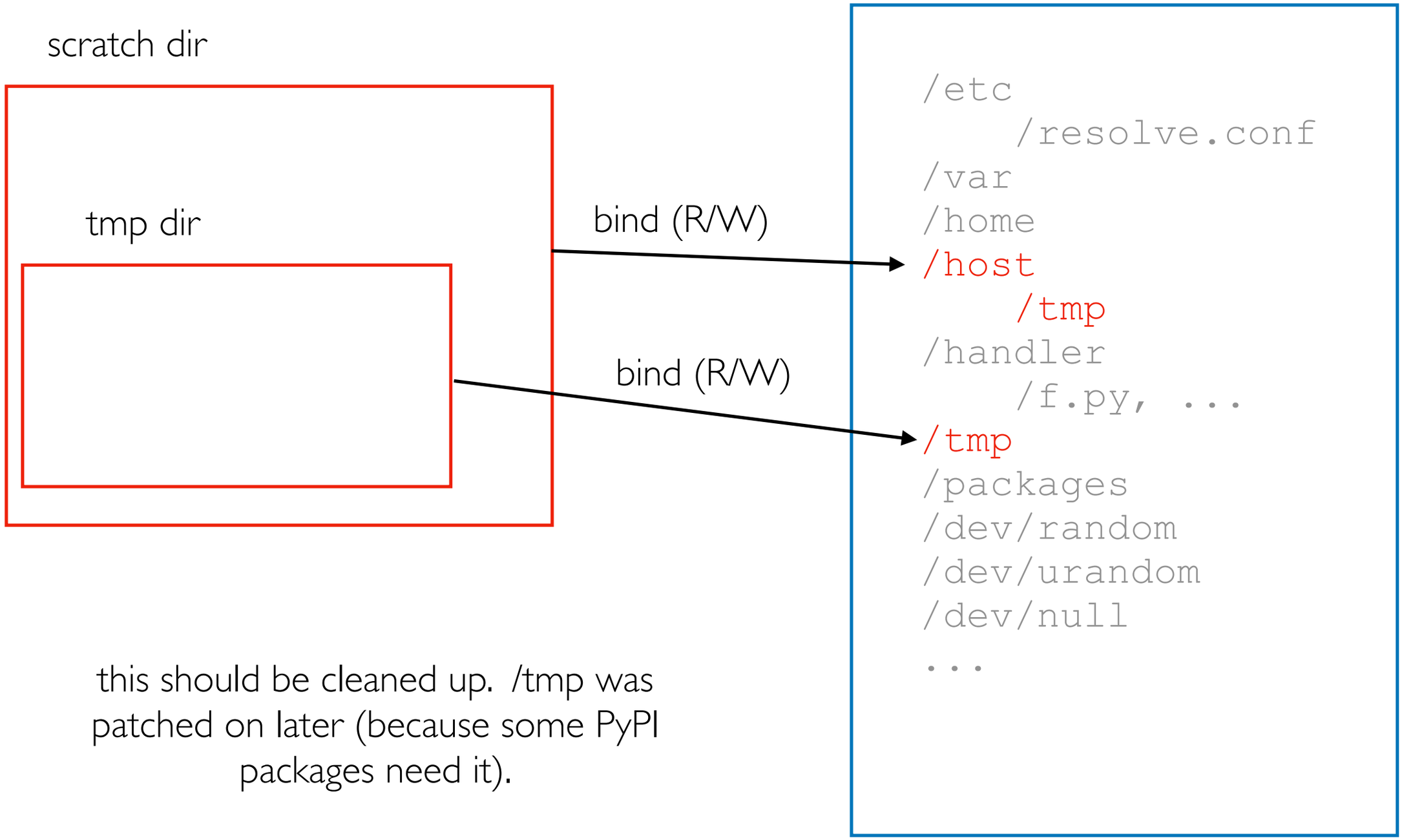
```
f.py
README.txt
```

bind (read only) →

```
/etc
    /resolve.conf
/var
/home
/host
/handler
    /f.py, ...
/packages
/dev/random
/dev/urandom
/dev/null
...
```

OpenLambda Init SOCK Sandbox

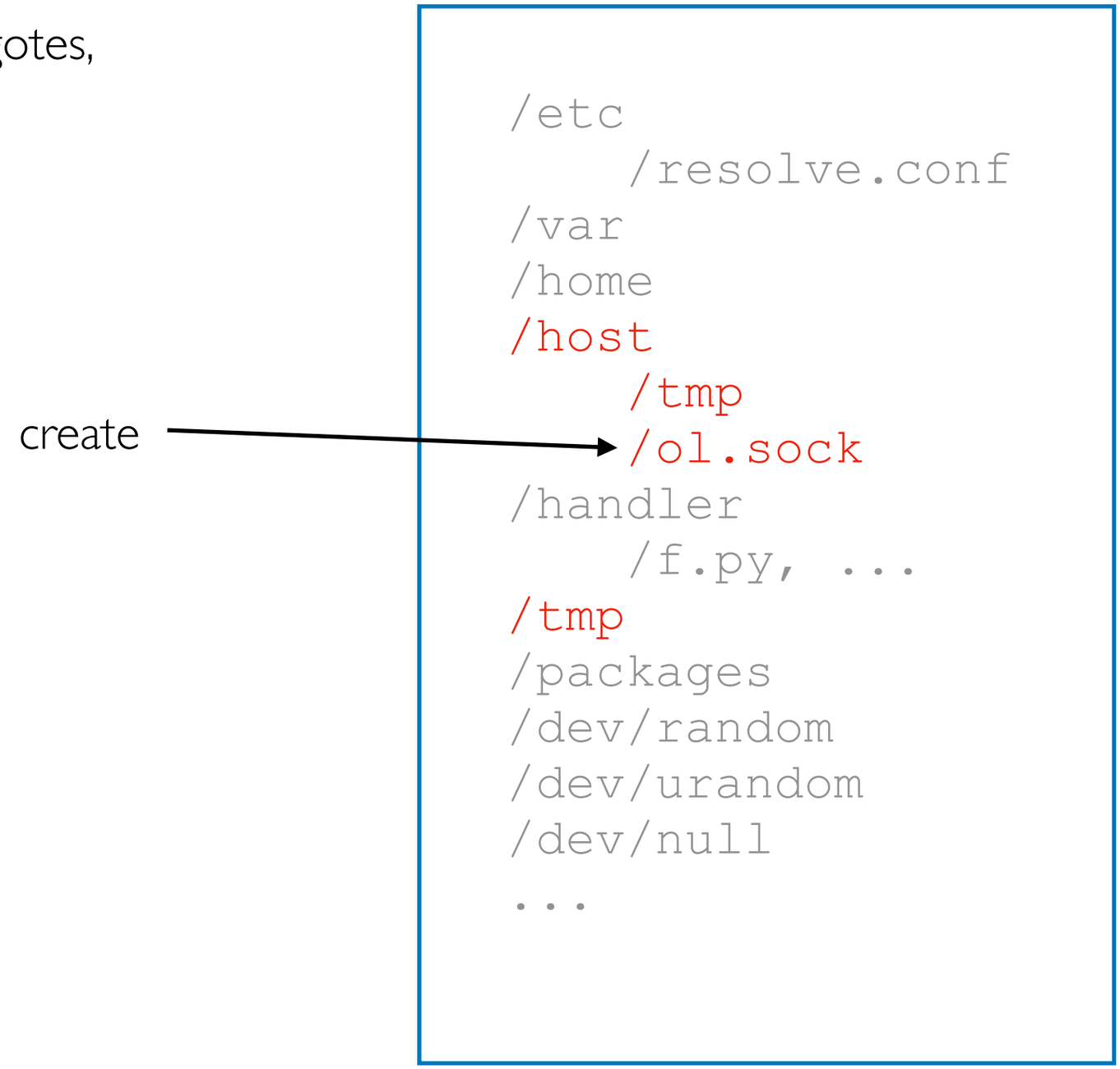
OpenLambda Init SOCK Sandbox



this should be cleaned up. /tmp was patched on later (because some PyPI packages need it).

OpenLambda Init SOCK Sandbox

ol.sock is used to send FDs for Zygotes,
and to send data for requests



OpenLambda Init SOCK Sandbox

SOCK will chroot to this.
We don't use union file systems.
We don't use mount namespaces
(at least per container)

```
/etc
    /resolve.conf
/var
/home
/host
    /tmp
    /ol.sock
/handler
    /f.py, ...
/tmp
/packages
/dev/random
/dev/urandom
/dev/null
...
```

Package Install

```
/etc
    /resolve.conf
/var
/home
/host
    /tmp
    /ol.sock
/handler
    /f.py, ...
/tmp
/packages
    /numpy/...
/dev/random
/dev/urandom
/dev/null
...
```

read only from inside most containers, but stuff gets added during runtime (special install containers can write to specific sub directories)

Request Forwarding