try(1) primer

Inspect a command's effects before modifying your live system

Tianyu (Eric) Zhu
tzhu22@stevens.edu
https://ericz.me

Stevens Institute of Technology

October 19, 2023

Do, or do not. There is no try.
We're setting out to change that.

`https://github.com/binpash/try`

Konstantinos Kallas
Georgios Liargkovas
Michael Greenberg
And the rest of the PaSh team.

Table of Contents

What's try for? - The Bigger Picture

Parallelizing the shell

- Run each line of a script in it's own sandbox
- Observe it's changes to the filesystem (via Riker)
- Merge the correct changes to the filesystem

What's try for? - The Bigger Picture

A Quick Example

```
echo "Making Weather Report!"
echo $(date) > report
echo it's sunny >> report
```

What's try for? - The Bigger Picture

A Quick Example

```
./try -D sandbox-a 'echo "Making Weather Report!"'
./try -D sandbox-b 'echo $(date) > report'
./try -D sandbox-c 'it's sunny >> report'
```

This is a simplified version of what actually happens.
hs will first send the script to our preprocessor PaSh, then call try with riker to receive filesystem changes and intercept network calls.

What's try for? - The Bigger Picture

Checking for dependencies (conflicts)
We use Riker to check for the filesystem changes.

- Did Command B read/write something that Command a used? No! We're good.
- Did Command C read/write something to a file used by Command A and B? Yes! We need to re-run C.

What's try for? - The Bigger Picture

Merge & Re-run

- We know that command A and B are ready to be merged into the host fs
- **After** the previous commands' changes are committed, we will re-run command c
- Commit command c's changes back to the filesystem

Execution complete :)

What's try for? - The other use for try

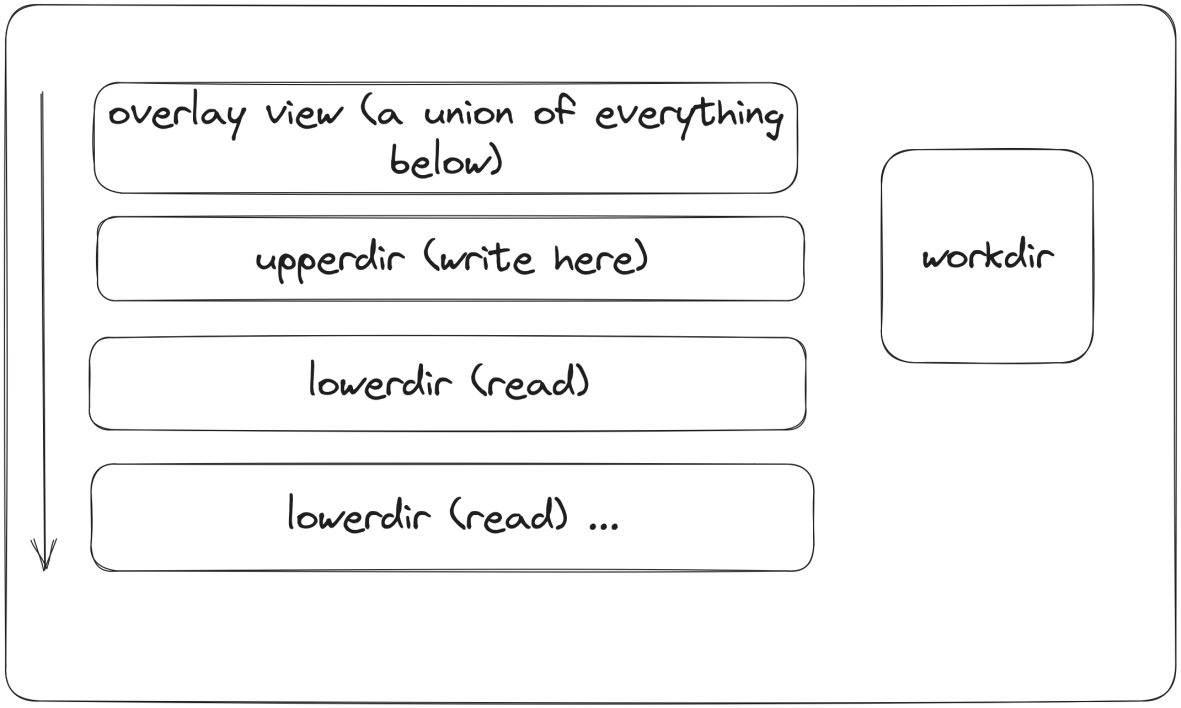
```
./try "curl link-to-a-sus-script.sh | bash"
```

Overlayfs

- Instead of writing changes to a directory directly, overlayfs writes it in another directory.
- Merge multiple directories together into one unified view.
- Program thinks they're just writing making these changes directly.

Overlayfs

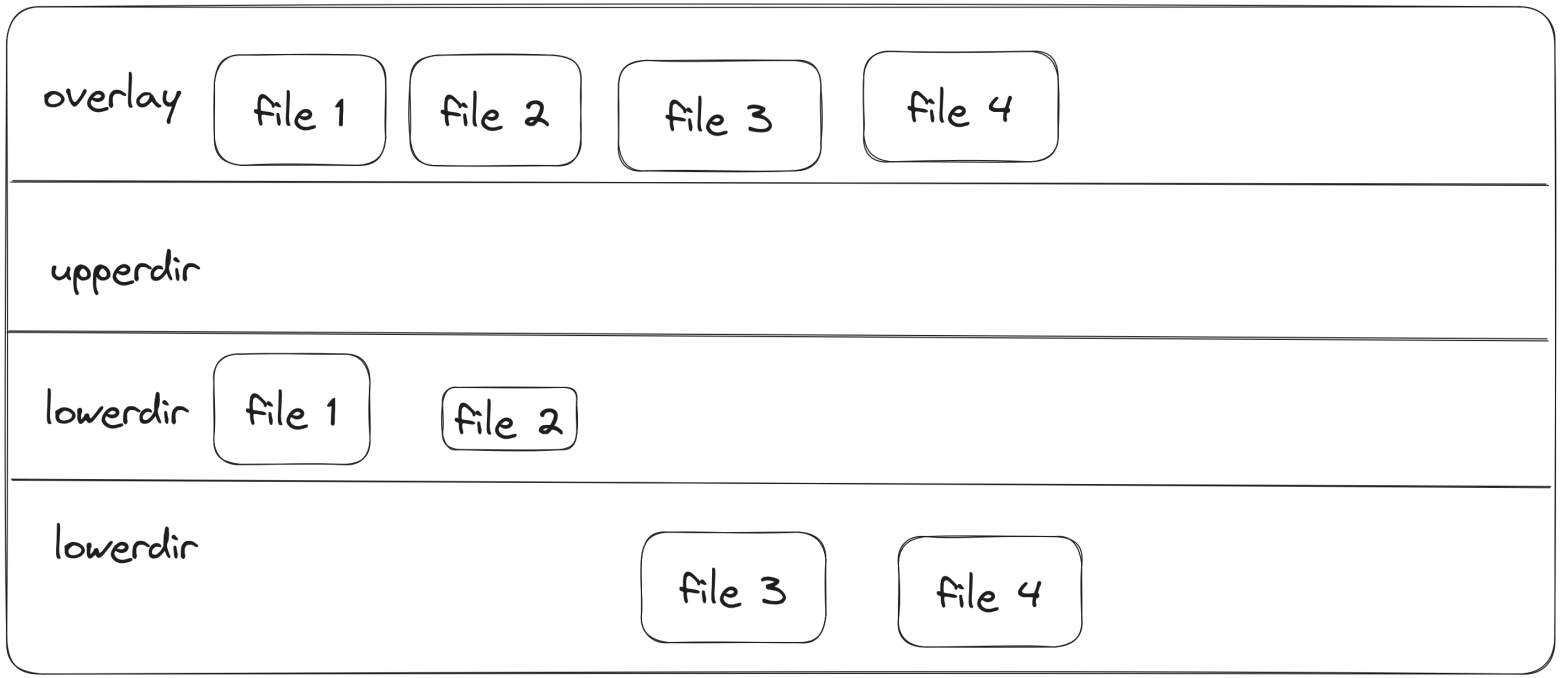
How overlayfs works



`https://docs.kernel.org/filesystems/overlayfs.html`

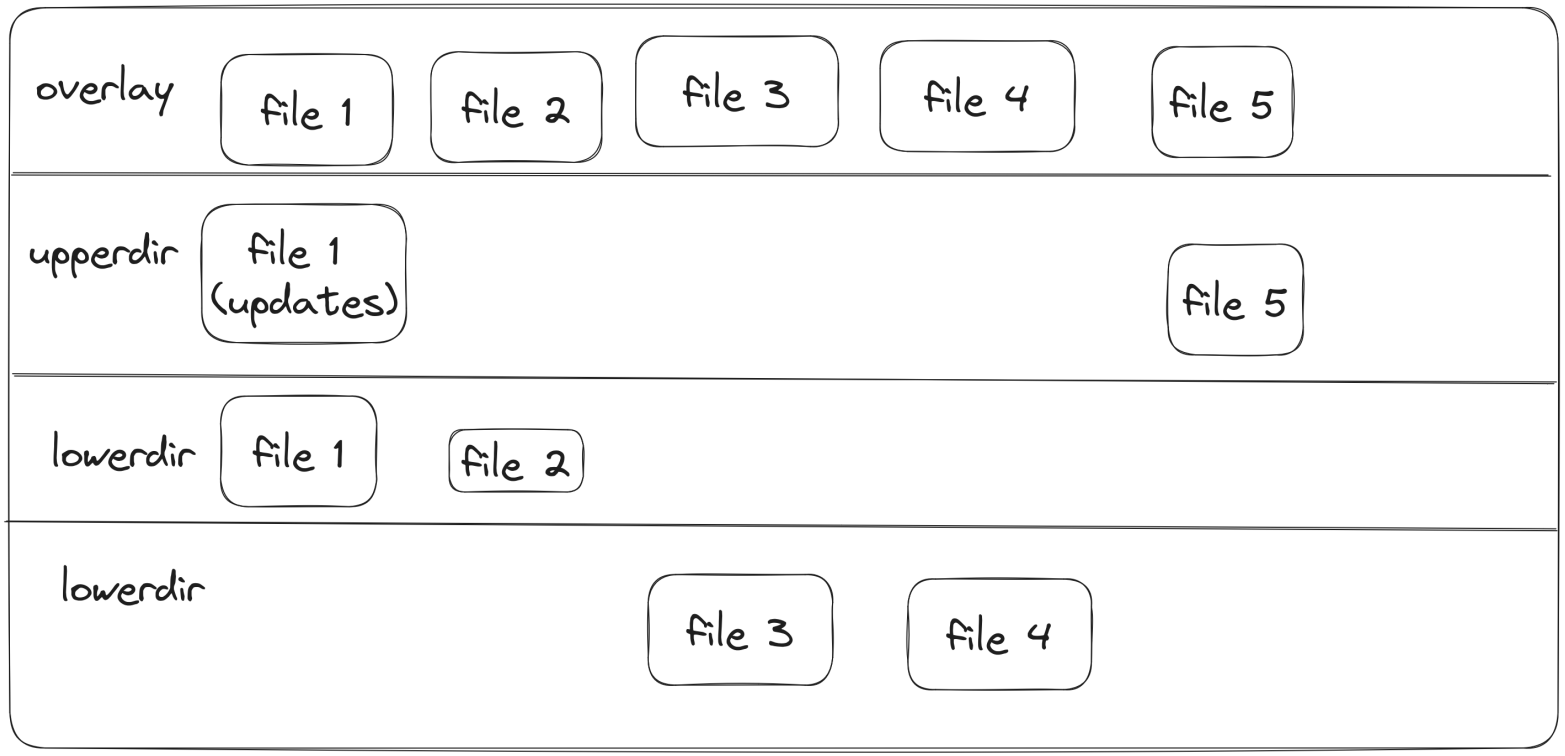
Overlayfs

Beginning state



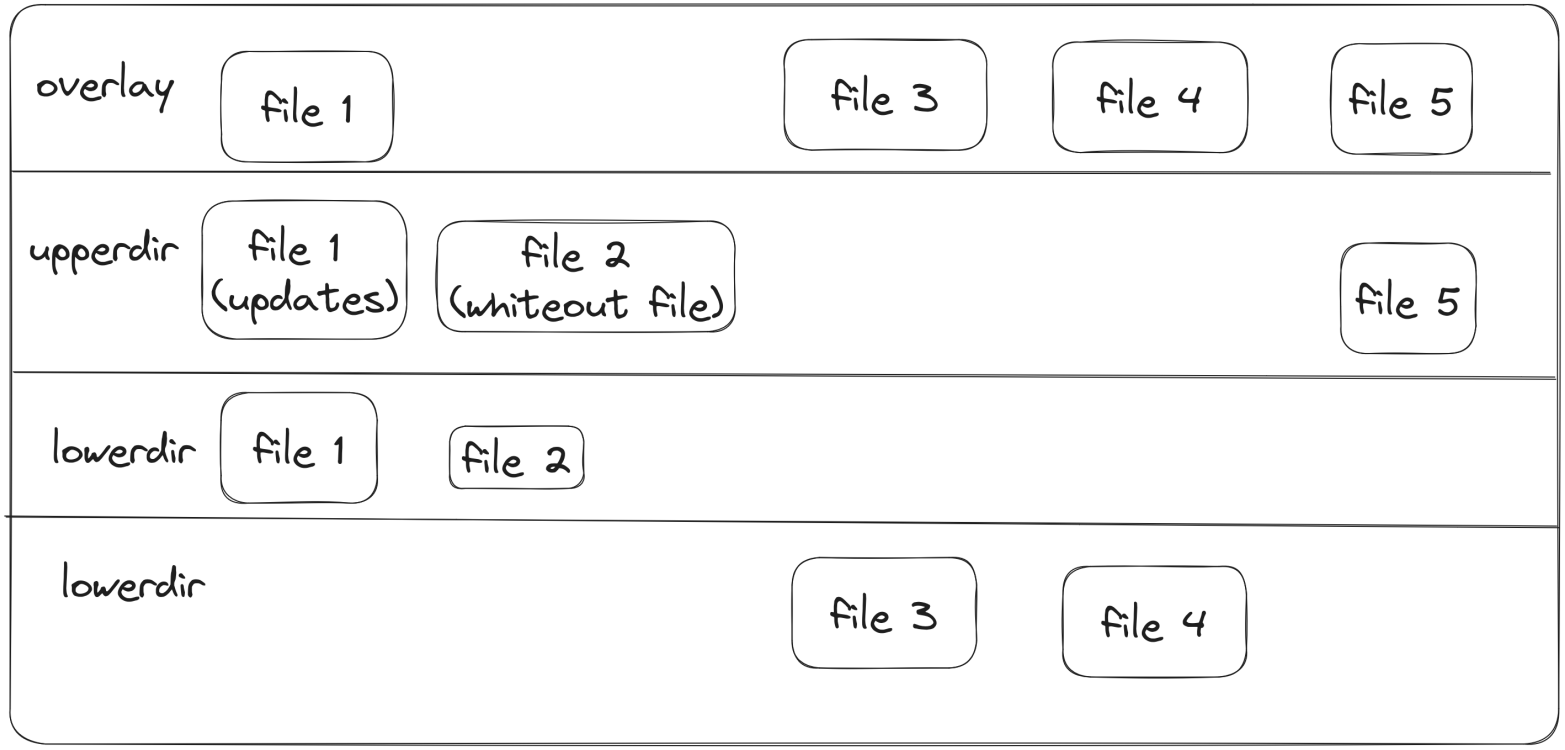
Overlayfs

Makes file 5, edits file 1



Overlayfs

Deletes file 2



Overlayfs

```
mount -t overlay overlay \
-o "lowerdir=$lowerdir1,$lowerdir2,upperdir=$upperdir,workdir=$workdir" "$overlay"
```

Overlayfs

```
$ docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
a803e7c4b030: Pull complete
009c876521a0: Pull complete
9c412905cca2: Pull complete
6463d4bf467a: Pull complete
bd8b983728ed: Pull complete
febc167f3560: Pull complete
d73c81c4ade3: Pull complete
34b3b0ac6e9e: Pull complete
9bd86d074f4e: Pull complete
406f63329750: Pull complete
ec40772694b7: Pull complete
7d3dfa1637e9: Pull complete
e217ca41159f: Pull complete
Digest: sha256:f1aaf6f8be5552bef66c5580efbd2942c37d7277cd0416ef4939fa34bf0baf31
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
```

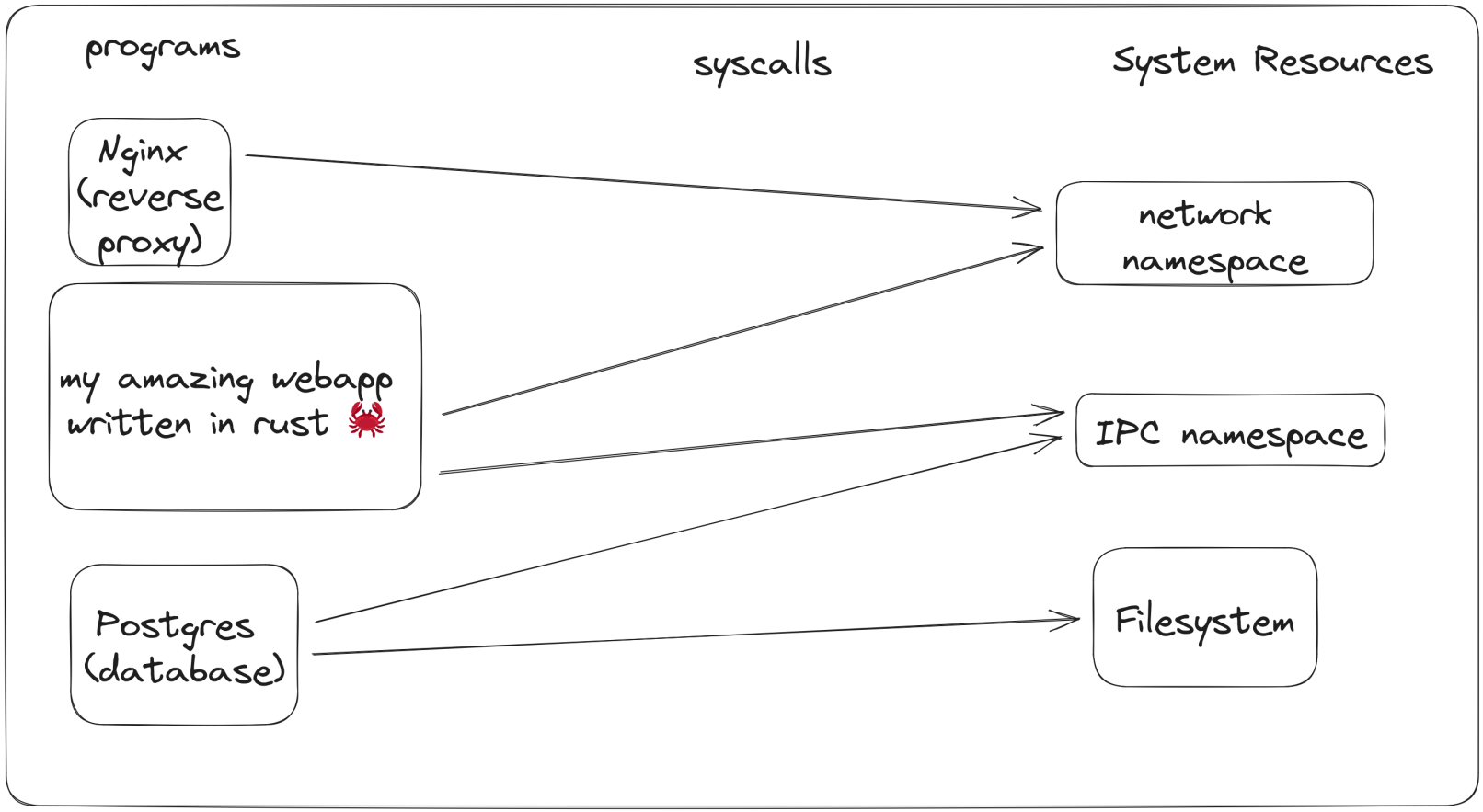
Overlayfs

```
$ cd "$(mktemp -d)" && mkdir lower upper work merged
/tmp/tmp.ANtPzAhoyk

$ mount -t overlay overlay \
-o "lowerdir=$PWD/lower,upperdir=$PWD/upper,workdir=$PWD/work" \
merged
mount: /tmp/tmp.ANtPzAhoyk/merged: must be superuser to use mount.
dmesg(1) may have more information after failed mount system call.
```

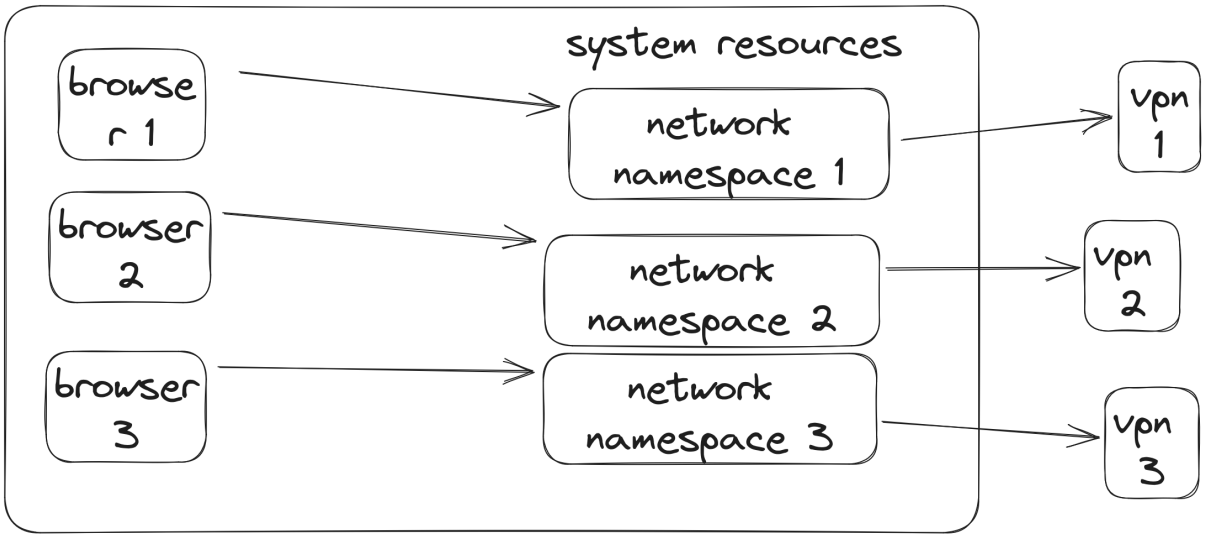
Linux Namespace via unshare

This is your computer, a program usually have access to all of these system resources provided by the Kernel.
By default, processes you call will inherit all of your namespaces



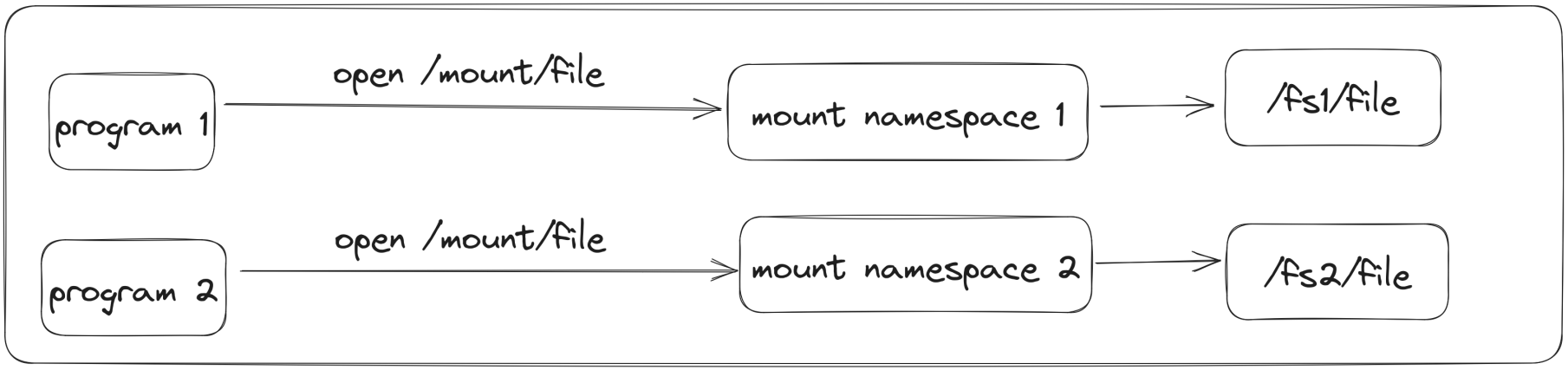
Linux Namespace via unshare

TD's Computer (he's a bit paranoid)



Linux Namespace via unshare

We have two namespaces here, each has a different directory mounted at 'mount'.



But since Program 1 and 2 are in their own mount namespaces, they can access the same file path, but receive a different file from the disk.

Linux Namespace via unshare

```
$ mkdir fs1 fs2 mount
$ echo a > fs1/file
$ echo b > fs2/fil3


$ unshare --map-root-user --mount /bin/bash
# mount --bind fs1 mount
# cat mount/file
a


$ unshare --map-root-user --mount /bin/bash
# mount --bind fs2 mount
# cat mount/file
b
```

try walkthrough

```
Usage: $TRY_COMMAND [-nvhy] [-i PATTERN] [-D DIR] [-U PATH] CMD [ARG ...]

  -n                don't prompt for commit
  -y                assume yes to all prompts (implies -n is not used)
  -i PATTERN        ignore paths that match PATTERN on summary and commit
  -D DIR            work in DIR (implies -n)
  -U PATH           path to unionfs helper (e.g., mergerfs, unionfs-fuse)

  -v                show version information (and exit)
  -h                show this usage message (and exit)


Subcommands:
  $TRY_COMMAND summary DIR   show the summary for the overlay in DIR
  $TRY_COMMAND commit DIR    commit the overlay in DIR
  $TRY_COMMAND explore DIR   start a shell inside the overlay in DIR
EOF
```

try walkthrough

```
mkdir -p \
    "$SANDBOX_DIR/upperdir" \
    "$SANDBOX_DIR/workdir" \
    "$SANDBOX_DIR/temproot"

DIRS_AND_MOUNTS="$(mktemp)"
export DIRS_AND_MOUNTS
find / -maxdepth 1 >"$DIRS_AND_MOUNTS"
findmnt --real -r -o target -n >>"$DIRS_AND_MOUNTS"
sort -u -o "$DIRS_AND_MOUNTS" "$DIRS_AND_MOUNTS"
```

try walkthrough

```
unshare --mount --map-root-user --pid --fork \
    "$mount_and_execute"
```

try walkthrough

```
make_overlay() {
    sandbox_dir="$1"
    lowerdir="$2"
    mountpoint="$3"
    mount -t overlay overlay -o userxattr -o "\
        lowerdir=$lowerdir,\
        upperdir=$sandbox_dir/upperdir/$mountpoint,\
        workdir=$sandbox_dir/workdir/$mountpoint" \
        "$sandbox_dir/temproot/$mountpoint"
}
devices_to_mount="tty null zero full random urandom"
```

try walkthrough

```
for mountpoint in $(cat "$DIRS_AND_MOUNTS")
    make_overlay "$SANDBOX_DIR" "/$mountpoint" "$mountpoint"
    if [ "$?" -ne 0 ]
        "$UNION_HELPER" $mountpoint $merger_dir
        make_overlay "$SANDBOX_DIR" "$merger_dir" "$mountpoint"
    fi
mount_devices "$SANDBOX_DIR"
unshare --root="$SANDBOX_DIR/temproot" /bin/sh "$chroot_executable"
```

try walkthrough

```
#!/bin/sh

mount -t proc proc /proc &&
cd "$START_DIR" &&
. "$script_to_execute"
```

try demo

Try demo!

Riker

https://github.com/curtsinger-lab/riker
https://www.usenix.org/conference/atc22/presentation/curtsinger
Charlie Curtsinger, Grinnell College; Daniel W. Barowy, Williams College

Riker demo

Riker demo! (last demo I promise)

Thank you!

Questions?
tzhu22@stevens.edu
eric@ericz.me
https://ericz.me/talks