PuzzleFS

the next-generation container filesystem

github.com/project-machine/puzzlefs

Ariel Miculas
amiculas@cisco.com

Overview

- Introduction
- Oci drawbacks
- Design goals
- Status + demo
- PuzzleFS data format
- Results
- Linux kernel filesystem driver (POC)
- Questions

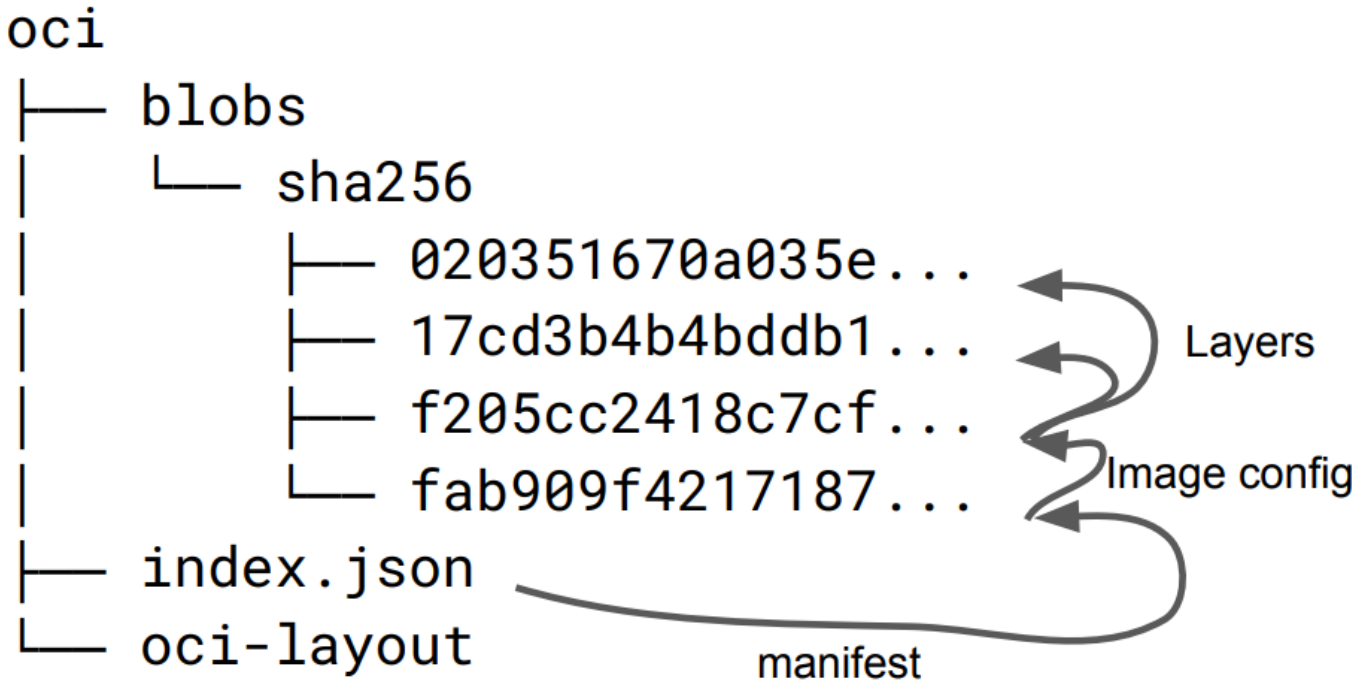
Introduction

- PuzzleFS is an immutable filesystem which shares design goals with the OCIv2 brainstorm
- Uses Content defined chunking (CDC) to split a filesystem into variable sized chunks
- The chunks are kept in a data store (content addressed)
- Metadata is stored separately and it has links to the data blobs

Context

- Started by Tycho Andersen in 2021
- His Fosdem presentation from 2019 “An operator centric way to update application containers with AtomFS” highligted the issues with OCIv1 and introduced AtomFS
- Atomfs was also presented by my colleague Scott Moser at Fosdem 2023 “Quick starting secure container storage using squashfs, overlay and dm-verity”
- PuzzleFS aims to be the successor of AtomFS
- Part of project-machine – an OCI-based secure container linux

OCI format basics



OCI v1 drawbacks

- Blog post written by Aleksa Sarai in 2019 describing the issues with the tar format (layers are usually tar(.gz) files)
  - Not a well defined format, but a collection of different formats, each with their own extensions
  - No index – archive entries consist of header+content
  - Not seekable – applies to compressed tar archives
  - No de-duplication – any change leads to re-downloading the whole
  - No machine-independent representation - directory entries and xattrs
  - Lack of reproducibility, no canonical representation - different tar extensions that solve the same problem (5 for xattrs)

Design goals

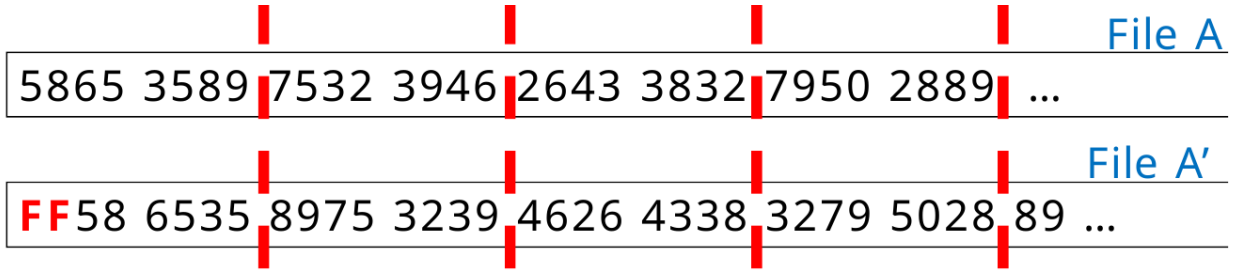
Solve the most pertinent OCI v1 problems

- Reduced duplication
- Reproducible image build
- Direct mounting support
- Data integrity
- Memory safety guarantees
- Same implementation in userspace and kernel

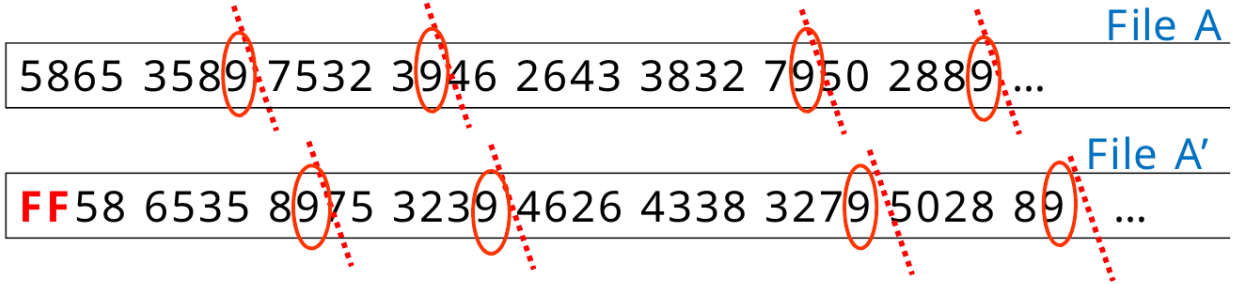
Reduced duplication

- Content defined chunking solves the boundary shift problem
- PuzzleFS uses FastCDC to chunk a filesystem into variable sized data blobs
- Configurable by defining a minimum, average and maximum chunk size

Boundary shift problem



➔➔*FSC: No duplicates will be detected*



➔➔*CDC: Most duplicates will be detected*

Issue: small patches

Ubuntu:N size: 80M

libssl.so PATCH

Ubuntu:N+1 size: 80M delta size: 80M

Solution: CDC

Ubuntu:N size: 80M

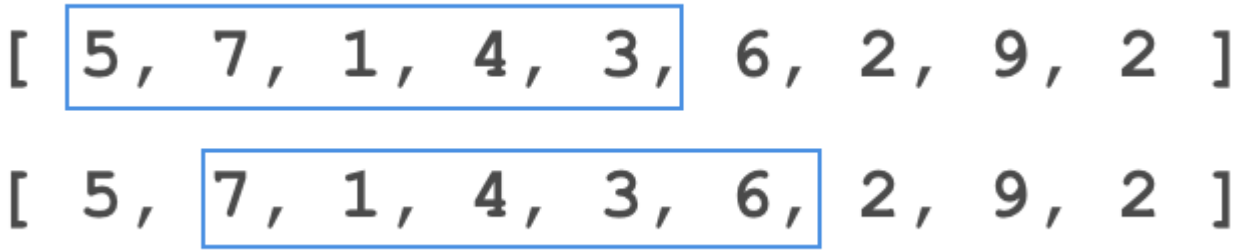
libssl.so PATCH

Ubuntu:N+1 size: 80M delta size: ~80K (avg chunk size)

Content defined chunking

- Sliding window technique to compute the hash of the window (rolling hash)
- If the last n bits of the hash are 0, generate a cut point
- Cut points only depend on the last <window size> bytes

  (e.g. 48 bytes)



Reproducible image build

Canonical representation of the filesystem

- Same traversal order of the filesystem when building an image
- Directory entries, extended attributes are sorted lexicographically
- B-tree maps used instead of hash maps for a defined ordering

Direct mounting support

- The goal is to prevent tampering
- No extraction step necessary (unlike tar)
- Mountable filesystem format - be simple enough to be decoded in the kernel

Data integrity

- Prevent tampering (dm-verity doesn’t fit our use case)
- Puzzlefs has optional support for fs-verity (protects files)
- Must be supported by the underlying filesystem of the puzzlefs image
- Fs-verity is computed for each file and stored in the image manifest
- The image manifest’s fs-verity hash is passed on the command line of “puzzlefs mount” command

Memory safety guarantees

- Implemented in Rust (both the FUSE and the in-kernel filesystem POC)
- Eliminates undefined behavior and entire classes of bugs (dangling pointers, use-after-free, buffer overflow)
- Strong typesystem
- First-class support for writing unit and integration tests
- Painless iterative development

Sharing the same code in user and kernel space

- Rust support for the kernel was merged in Linux 6.1
- Don’t write the same code twice

  Differences:
- The kernel only allows fallible allocations (allowed to fail)
- Cannot handle file operations in the same way as in user space
- Code must duplicated because the kernel cannot fetch code from crates.io (or use the cargo build system)

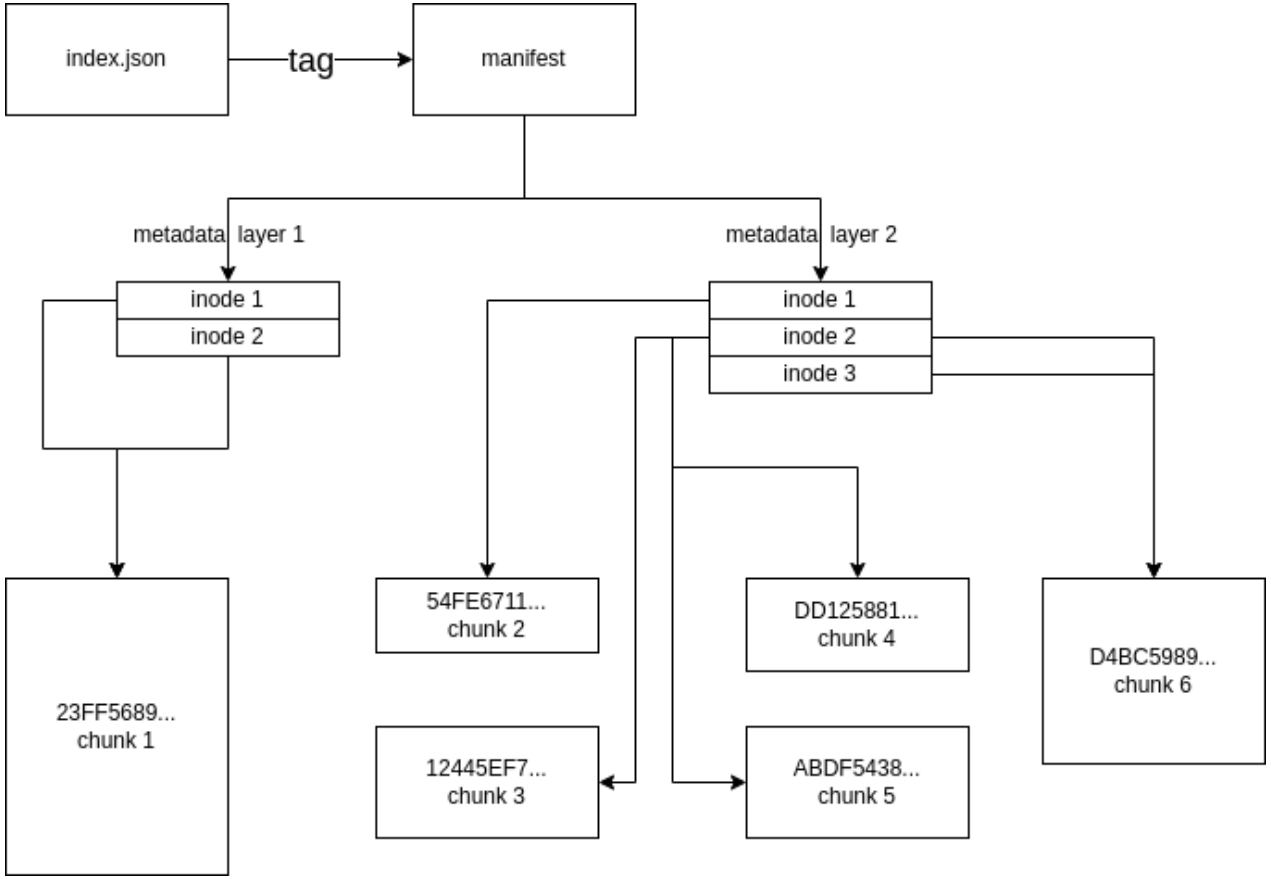
Status

- Build, extract and fuse-mount puzzlefs filesystems
- fs-verity – requires filesystem support from the underlying data store
- Optional zstd compression for the data blobs
- Proof-of-concept Linux filesystem drivers written in Rust

Demo

```
❯ tree /tmp/example-rootfs
/tmp/example-rootfs
├── algorithms
│   └── binary-search.txt
└── lorem_ipsum.txt

2 directories, 2 files
~/work/cisco/puzzlefs master*
~/work/cisco/puzzlefs master*
❯ target/release/puzzlefs build /tmp/example-rootfs ~/oci/puzzlefs-image puzzlefs-example
puzzlefs image manifest digest: da22d5157f337237ab24b8d6843f6525311ec60d0e4a18c56d91bb2d259f2a43
~/work/cisco/puzzlefs master*
❯ target/release/puzzlefs enable-fs-verity ~/oci/puzzlefs-image puzzlefs-example da22d5157f337237ab24b8d6843f65
25311ec60d0e4a18c56d91bb2d259f2a43
~/work/cisco/puzzlefs master*
❯ target/release/puzzlefs mount --digest da22d5157f337237ab24b8d6843f6525311ec60d0e4a18c56d91bb2d259f2a43 ~/oci
/puzzlefs-image puzzlefs-example /tmp/puzzle
~/work/cisco/puzzlefs master*
❯ journalctl --since "2 min ago" | grep puzzlefs
Aug 14 16:09:57 archlinux-cisco puzzlefs[161561]: Mounting /tmp/puzzle
~/work/cisco/puzzlefs master*
❯ mount | grep '/tmp/puzzle'
/dev/fuse on /tmp/puzzle type fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
```

PuzzleFS data format



PuzzleFS data format

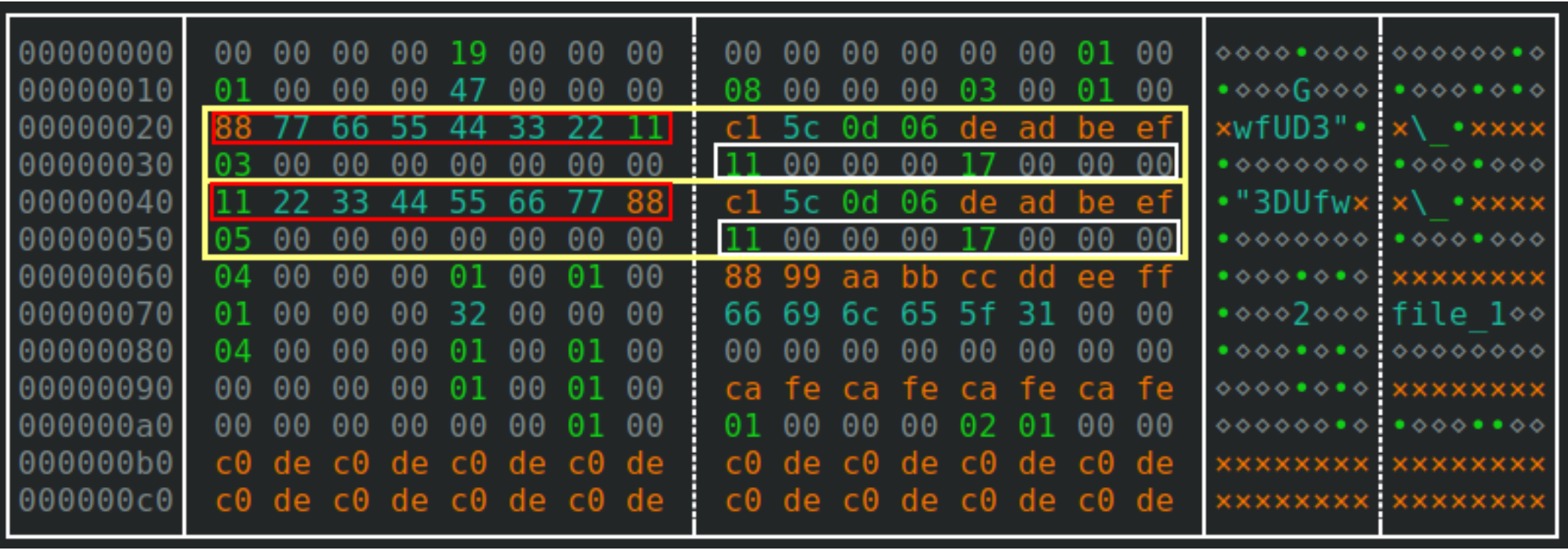
- Metadata is serialized using Capnproto (serialization protocol)
- There are two levels of indirection:
  - The image manifest contains a list of metadata layers and the associated fs-verity data
  - Each metadata layer contains the metadata for its files and directories, and links to data blobs
- Data blobs are stored content-addressed (they are named after their sha256 hash)

PuzzleFS data format

```
struct VerityData {
        digest@0: Data;
        verity@1: Data;
}

struct BlobRef {
    digest@0: Data;
    offset@1: UInt64;
    compressed@2: Bool;
}

struct Rootfs {
        metadatas@0: List(Metadata.BlobRef);
        fsVerityData@1: List(VerityData);
        manifestVersion@2: UInt64;
}
```

```
struct Inode {
    ino@0: UInt64;
    mode: union {unknown@1: Void;
         dir@4: Dir;
         file@6: List(FileChunk);
         ...
      }
    uid@10: UInt32;
    gid@11: UInt32;
    permissions@12: UInt16;
    additional@13: InodeAdditional;
}

struct InodeVector {
    inodes@0: List(Inode);
}
```

Compact inode representation



Results

- I’ve downloaded 10 versions of Jammy from hub.docker.com
- These images only have one layer which is in tar.gz format
- I’ve built 10 equivalent puzzlefs images
- Compute the tarball_total_size by summing the sizes of every Jammy tarball (uncompressed) => 766 MB (use this as baseline)
- Sum the sizes of every oci/puzzlefs image => total_size
- Compute the total size as if all the versions were stored in a single oci/puzzlefs repository => total_unified_size
- Saved space = tarball_total_size - total_unified_size

Results

Type	Total size (MB)	Average layer size (MB)	Unified size (MB)	Saved (MB) / 766 MB
Oci (uncompressed)	766	77	766	0 (0%)
PuzzleFS uncompressed	748	74	130	635 (83%)
Oci (compressed)	282	28	282	484 (63%)
PuzzleFS (compressed)	298	30	53	713 (93%)

Kernel filesystem driver

Cisco Posts Rust-Written PuzzleFS File-System Driver For Linux

Written by Michael Larabel in Linux Storage on 9 June 2023 at 05:50 AM EDT. 23 Comments

PuzzleFS is a next-generation container file-system for Linux with fast image building, direct-mount support, and other container-optimized features being worked on by Cisco engineers. And it's written in Rust.

Ariel Miculas of Cisco today posted an initial "request for comments" patch series on this PuzzleFS file-system with the kernel driver written in Rust. For now this Rust driver is considered proof-of-concept. The patch series goes on to describe PuzzleFS as:

> ***Puzzlefs is a container filesystem designed to address the limitations of the existing OCI format. The main goals of the project are reduced duplication, reproducible image builds, direct mounting support and memory safety guarantees, some inspired by the OCIv2 design document.***
>
> ***Reduced duplication is achieved using the content defined chunking algorithm FastCDC. This implementation allows chunks to be shared among layers. Building a new layer starting from an existing one allows reusing most of the chunks.***

Kernel filesystem driver

- Proof-of-concept driver written in Rust and posted to the kernel mailing list
- Two versions based on Wedson Almeida's filesystem and read-only filesystem abstractions (not yet upstream)
- Requires adding third-party crates to the Linux kernel (capnproto-rust and hex)
- Challenges:
  - many missing rust abstractions, infrastructure is still under development
  - Requires no-std support and can only use fallible allocation APIs (try_new instead of new, try_push instead of push etc.)

Filesystem driver demo

```
~ # cat /proc/filesystems | grep puzzlefs
nodev   puzzlefs
~ # cat /home/puzzlefs_oci/index.json
{"schemaVersion":-1,"manifests":[{"digest":"sha256:c43e5ab9d0cee1dcfbf442d18023b34410de3deb0f
~ # mount -t puzzlefs -o oci_root_dir="/home/puzzlefs_oci" -o image_manifest="c4
3e5ab9d0cee1dcfbf442d18023b34410de3deb0f6dbffcec72732b6830db09" none /mnt
~ # ls -la /mnt/
total 0
drwxr-xr-x    2 0        0               0 Aug 11 13:50 dir-1
drwxr-xr-x    2 0        0               0 Aug 11 13:50 dir-2
drwxr-xr-x    2 0        0               0 Aug 11 13:50 dir-3
drwxr-xr-x    2 0        0               0 Aug 11 13:50 dir-4
-rw-r--r--    1 0        0               0 Aug 11 13:50 file1
-rw-r--r--    1 0        0               0 Aug 11 13:50 file2
~ # wc /mnt/file1
      202       202      5454 /mnt/file1
~ # cat /mnt/file2
ana are mere bla bla bla
~ #
```

Capnproto-rust kernel integration

- No alloc support – easier than supporting kernel's custom alloc
- Replace Strings with enum in error codes (no String in kernel)
- Introduce NoAllocBufferSegments - a version of BufferSegments suitable for no alloc environments (to avoid parsing the capnp message every time a field is accessed)

Questions?

Let’s stay in touch!

https://github.com/project-machine/puzzlefs

amiculas@cisco.com