loongArch/os模块

陈林峰
北京理工大学
2023.3.26

rCoreloongArch

- rCore
  rCore 是清华大学教学操作系统 uCore 的 Rust 移植版本。致力于使用现代编程语言，提升 OS 的开发体验和质量，探索未来 OS 的设计实现方式
- loongArch64
  2020 年，龙芯中科推出了龙芯指令集（LoongArch®），包括基础架构部分和向量指令、虚拟化、二进制翻译等扩展部分，近 2000 条指令。龙芯指令系统具有较好的自主性、先进性与兼容性。龙芯指令系统从整个架构的顶层规划，到各部分的功能定义，再到细节上每条指令的编码、名称、含义，在架构上进行自主重新设计，具有充分的自主性。

1.loongArch: https://github.com/loongson/LoongArch-Documentation
2.rCore: https://github.com/rcore-os/rCore-Tutorial-v3

Why

- 提供适合普通学生或开发者开发基于loongArch指令集机器的基础知识
- 改善loongArch在rust生态中的不利地位
- Study how to write an os for a new arch
- Study the loongArch knowledge

---

1. 比赛仓库 https://github.com/Godones/rCoreloongArch

1. 启动
    1. How to compile bare metal code
    2. How to run kernel from uefi
2. 硬件抽象
   loongArch 上包含大量的 CSR 寄存器以及一些特殊指令

```
Ticlr::read().clear_timer().write(); //清除时钟专断
Tcfg::read().set_enable(false).write();
Ecfg::read().set_lie_with_index(11, false).write();
Crmd::read().set_ie(false).write(); //关闭全局中断
SltbPs::read().set_page_size(0xe).write(); //设置TLB的页面大小为16KiB
TlbREhi::read().set_page_size(0xe).write(); //设置TLB的页面大小为16KiB
```

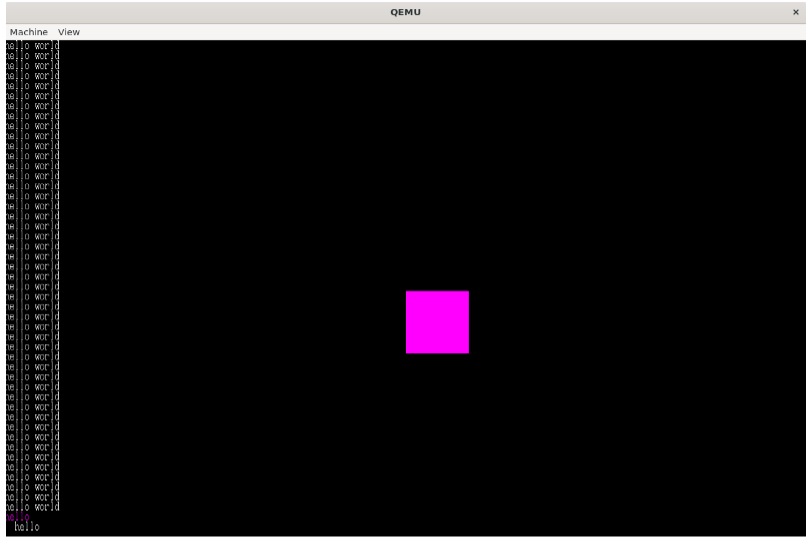
---

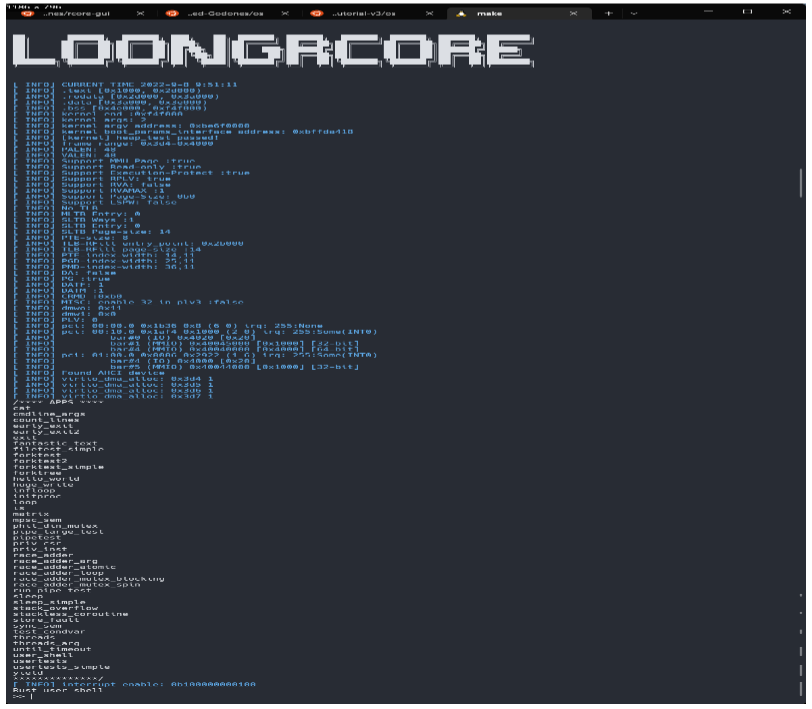
1. crate https://github.com/Godones/loongArch64

1. 虚拟内存管理
    1. 直接映射窗口
    2. TLB刷新
2. 中断
    1. 外部中断控制器
    2. 桥片中断
    3. 时钟中断
3. 外部设备
    1. PCI总线探测、sata设备、AHCI协议
4. 完善的文档

https://godones.github.io/rCoreloongArch/index.html

rCoreloongArch





rCoreloongArch-tutorial

项目简介

本文档作为 `rCore-Tutortial-Book-v3 3.6.0-alpha.1` 的 `loongArch` 版本，在不涉及硬件细节的部分请移步查看 `rCore` 文档，对于硬件无关的内容本文档只会介绍一些抽象细节和思路。整体的目录与 `rCore` 相同，同时会增加一些其它与项目相关的内容。

使用指南

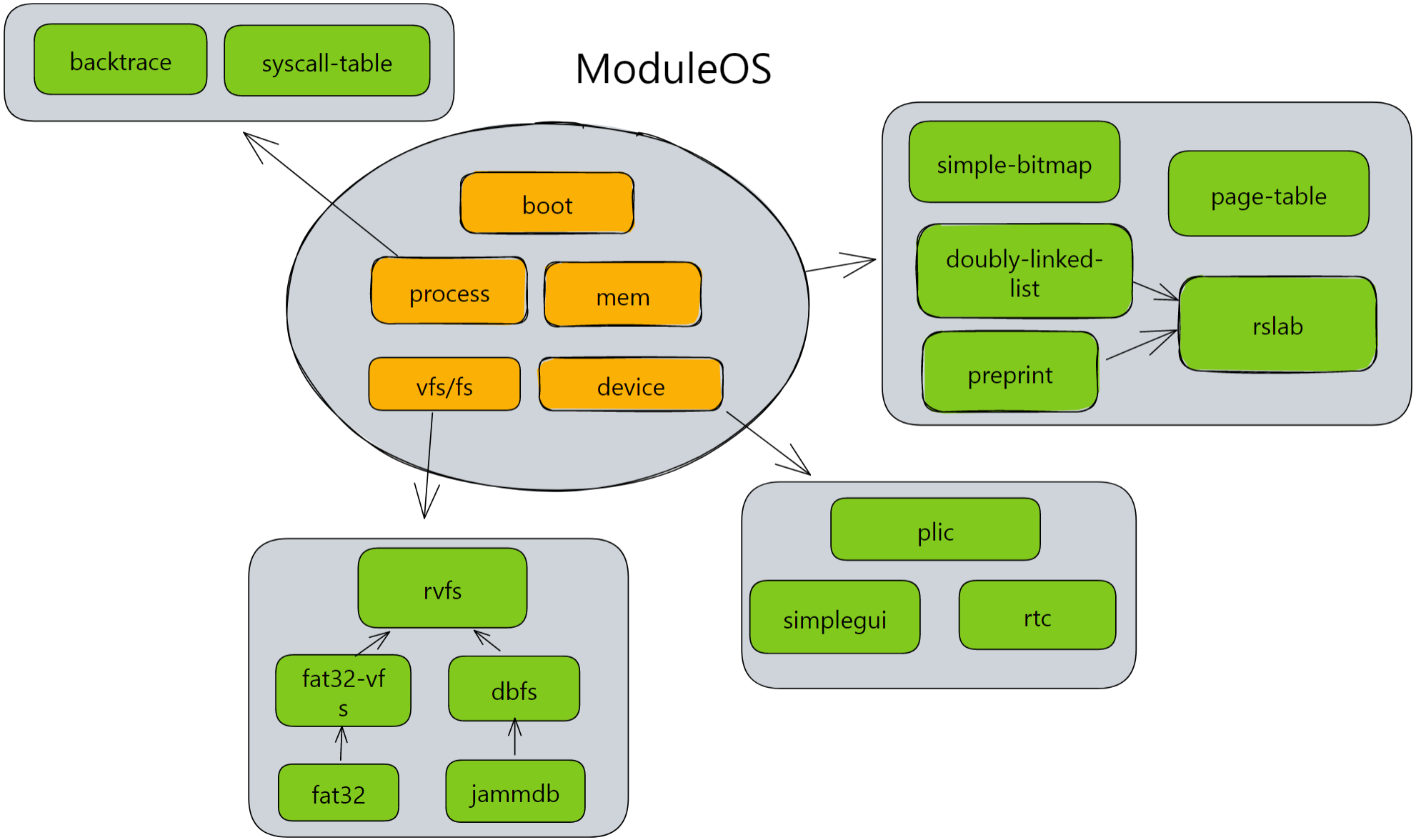
文档使用 `mdbook` 工具构建，代码块中的代码不一定可以运行.

使用下面的命令可以在线查看文档:

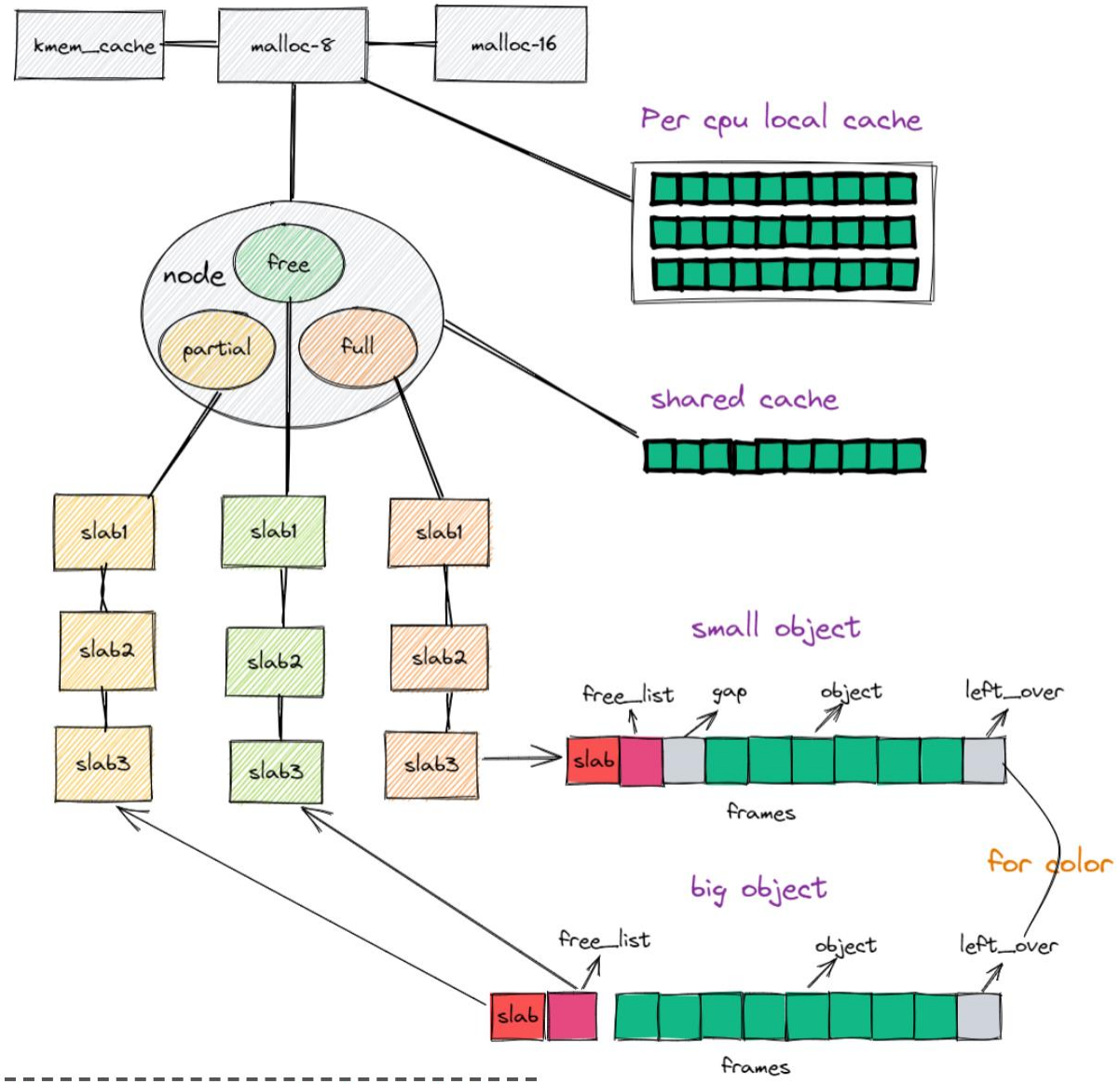
```
mdbook serve --open
```

若本机未安装 `mdbook`，则可以查看目录下提供的 `PDF` 文件。

OS模块化:总览


backtrace
syscall-table
ModuleOS
boot
process
mem
vfs/fs
device
simple-bitmap
page-table
doubly-linked-list
rslab
preprint
plic
simplegui
rtc
rvfs
fat32-vfs
dbfs
fat32
jammdb

OS模块化:Slab



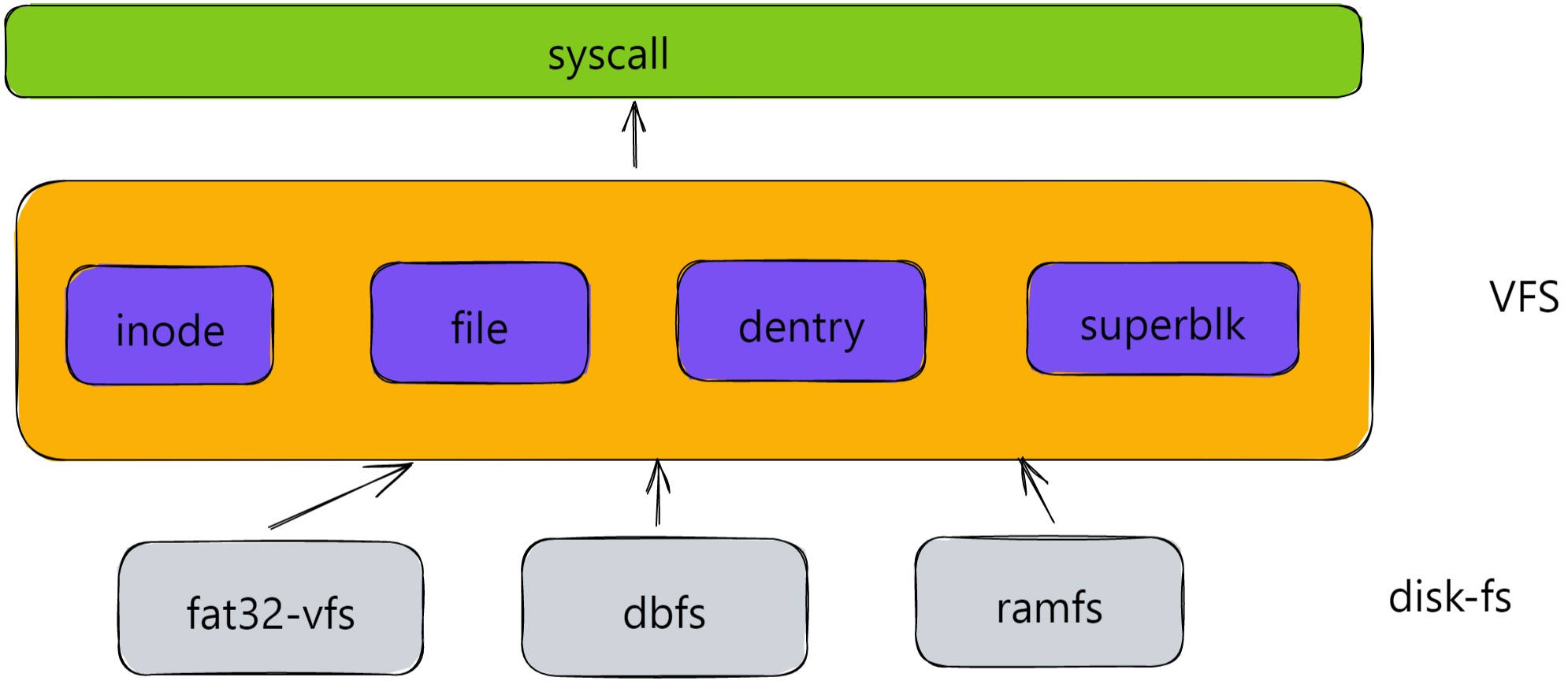
rslab: github https://github.com/os-module/rslab

OS模块化:fat32

```rust
pub trait DirectoryLike: Debug + Send + Sync {
    type Error:  Error + 'static;type FError: Error + 'static;
    fn create_dir(&self, name: &str) -> Result<(), Self::Error>;
    fn create_file(&self, name: &str) -> Result<(), Self::Error>;
    fn delete_dir(&self, name: &str) -> Result<(), Self::Error>;
    fn delete_file(&self, name: &str) -> Result<(), Self::Error>;
    fn cd(&self, name: &str) -> Result<Arc<dyn DirectoryLike<Error = Self::Error, FError=Self::FError>>, Self::Error>;
    fn open(&self, name: &str) -> Result<Arc<dyn FileLike<Error = Self::FError>>, Self::Error>;
    fn list(&self) -> Result<Vec<String>, Self::Error>;
    fn rename_file(&self, old_name: &str, new_name: &str) -> Result<(), Self::Error>;
    fn rename_dir(&self, old_name: &str, new_name: &str) -> Result<(), Self::Error>;
}
pub trait FileLike: Debug + Send + Sync {
    type Error:  Error + 'static;
    fn read(&self, offset: u32, size: u32) -> Result<Vec<u8>, Self::Error>;
    fn write(&self, offset: u32, data: &[u8]) -> Result<u32, Self::Error>;
    fn clear(&self);
    fn size(&self) -> u32;
}
```

fat32: github https://github.com/os-module/fat32

OS模块化:VFS



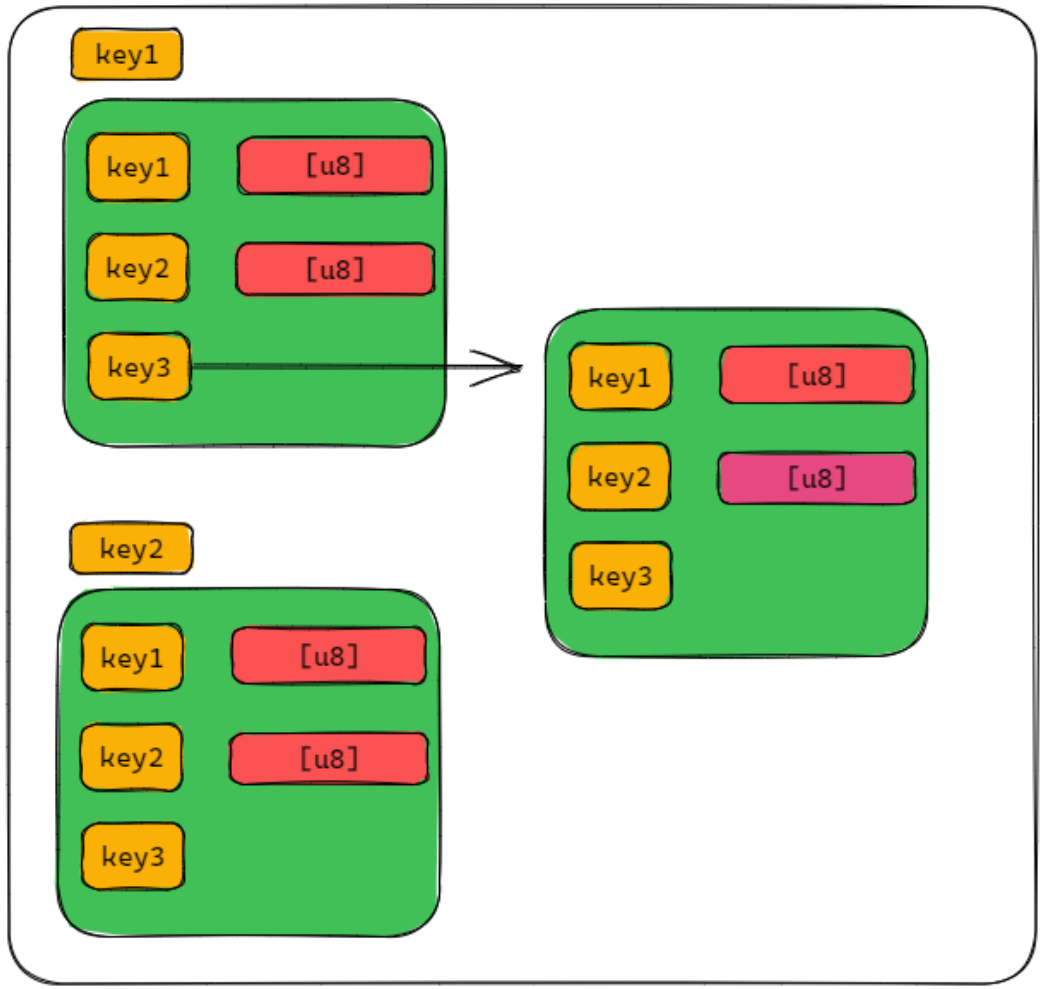
vfs: https://github.com/Godones/rvfs
fat32-vfs: https://github.com/Godones/fat32-vfs
dbfs2: https://github.com/Godones/dbfs2

OS模块化:Other

- backtrace tool
  一种利用二进制代码信息进行堆栈回溯的方法
- jammdb
  数据库的no_std移植
- simplegui
  gui界面的简单尝试
- syscall-table
  利用宏和编译前措施自动收集代码中的系统调用函数

dbfs

- ACID compliance
- serializable
- isolated transactions
- multiple readers and a single writer.
- B+ tree
- Memory Mapped

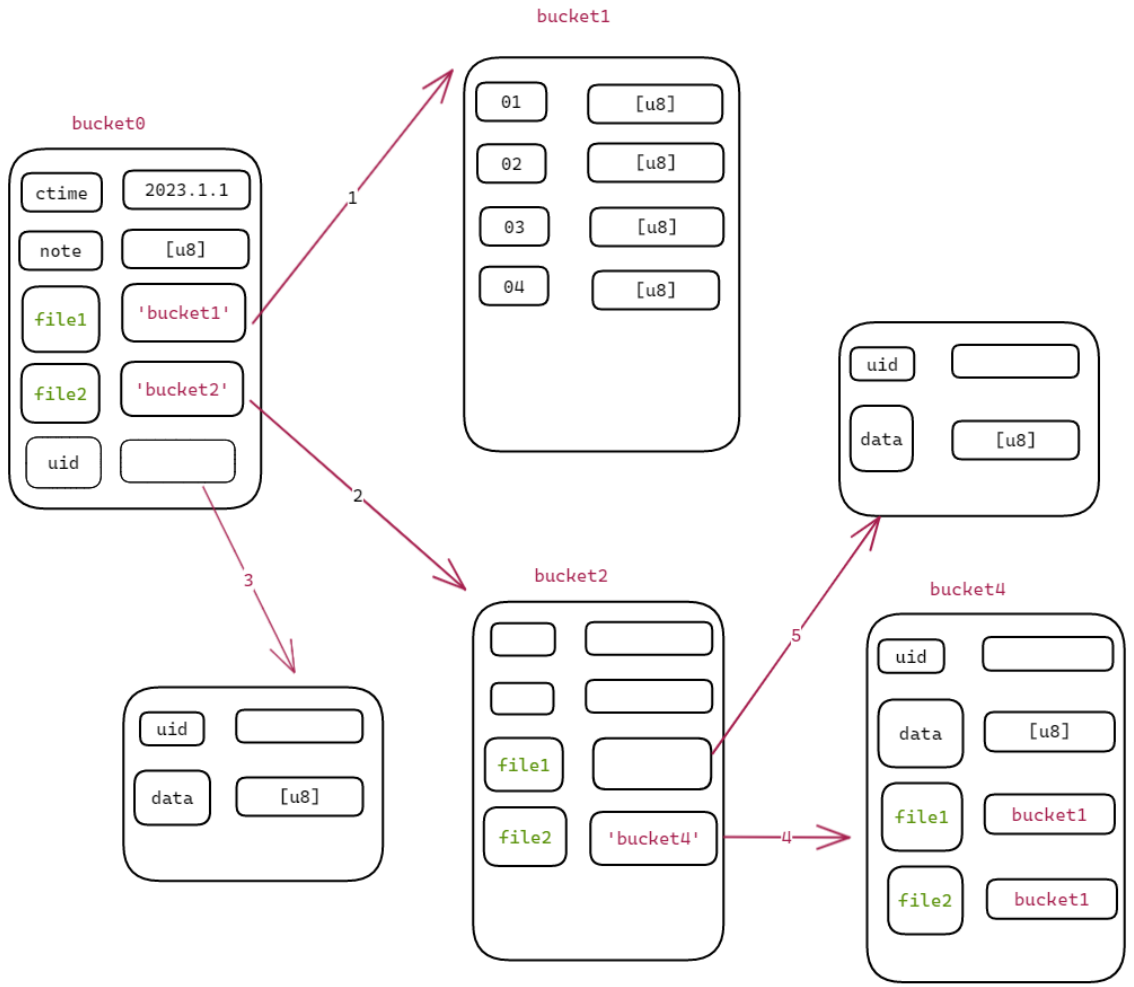


---

1. jammdb is an embedded, single-file database that allows you to store key / value pairs as bytes.
2. https://github.com/pjtatlow/jammdb

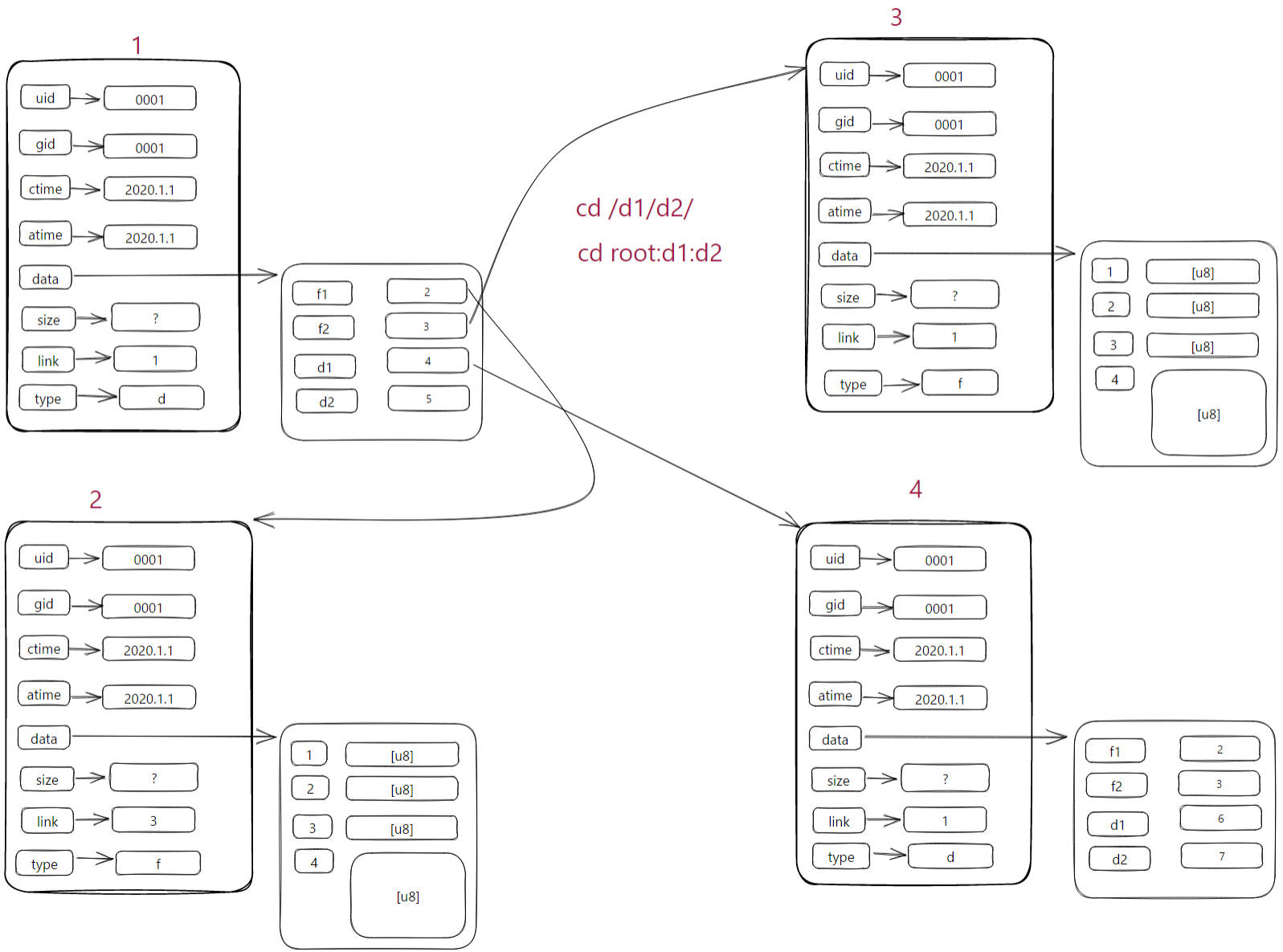
dbfs:统一模型

- bucket对应一个存储数据的载体
- 传统文件或目录使用键值对表述
- 操作系统控制一部分键值对
- 存储有结构的数据
- 用户有机会控制存储数据的结构
- 访问数据使用键值对进行
  root:key1:key2:ctime



bucket0:file2:file2:data

dbfs:兼容性


1
uid
0001
gid
0001
ctime
2020.1.1
atime
2020.1.1
data
size
?
link
1
type
d
f1
2
f2
3
d1
4
d2
5
cd /d1/d2/
cd root:d1:d2
3
uid
0001
gid
0001
ctime
2020.1.1
atime
2020.1.1
data
size
?
link
1
type
f
1
[u8]
2
[u8]
3
[u8]
4
[u8]
2
uid
0001
gid
0001
ctime
2020.1.1
atime
2020.1.1
data
size
?
link
3
type
f
1
[u8]
2
[u8]
3
[u8]
4
[u8]
4
uid
0001
gid
0001
ctime
2020.1.1
atime
2020.1.1
data
size
?
link
1
type
d
f1
2
f2
3
d1
6
d2
7

dbfs:统一模型下用户控制数据

1. 用户自定义对bucket的操作

```
let bucket = tx.get_bucket("test").unwrap();
bucket.put("name", b"hello".to_vec()).unwrap();
bucket.put("data1", b"world".to_vec()).unwrap();
bucket.put("data2", b"world".to_vec()).unwrap();
let mut start = 0;
let v1 = bucket.get_kv("name").unwrap();
buf[start..start + min_].copy_from_slice(&v1.value()[..min_]);
let v2 = bucket.get_kv("data1").unwrap();
buf[start..start + min_].copy_from_slice(&v2.value()[..min_]);
let v3 = bucket.get_kv("data2").unwrap();
buf[start..start + min_].copy_from_slice(&v3.value()[..min_]);
let dir = bucket.create_bucket("dir1").unwrap();
dir.put("uid", b"111".to_vec()).unwrap();
dir.put("gid", b"222".to_vec()).unwrap();
dir.put("mode", b"333".to_vec()).unwrap();
dir.create_bucket("dir2").unwrap();
```

- 内核无法控制用户函数的正确性
- 内核无法执行用户态的这些函数

dbfs:统一模型下用户控制数据

2. 内核提供限制性的操作，用户组合这些操作

```
add_key!(addkey_operate,
        ("name",b"hello".to_vec()),("data1",b"world".to_vec()),("data2",b"world".to_vec()));
let buf = [0u8; 20];
read_key!(read_operate,["name","data1","data2"],buf.as_ptr(),20);
let mut add_bucket = AddBucketOperate::new("dir1",None);
add_key!(add_operate,
    ("uid",b"111".to_vec()),("gid",b"222".to_vec()),("mode",b"333".to_vec())
);
let add_bucket1 = AddBucketOperate::new("dir2",None);
make_operate_set!(operate_set,
    [Operate::AddKey(add_operate),Operate::AddBucket(add_bucket1)]
);
add_bucket.add_other(Box::new(operate_set));
make_operate_set!(operate_set,
    [Operate::AddKey(addkey_operate),Operate::AddBucket(add_bucket),Operate::Read(read_operate)]
);
```

- 细粒度，灵活性，可组合性
- 保证内核的控制一定程度上限制了用户的操作

使用场景

- 压缩文件: 只读取压缩包内某一个特定的文件，而无需拉取整个压缩包
- 日志记录/文档记录： 读取特定范围/特定段落的内容

问题

1. 数据库的存储优化
2. 更灵活的操作组合
3. 如何对失败操作进行处理
4. 如何有效利用数据库的特性

收获

- 对操作系统的整体设计或工作原理有更好的认识
- 锻炼自己的工程能力
- 了解系统模块实现细节
- 做有意义的工作

Question ?

陈林峰

北京理工大学
2023.3.26

ppt/pdf: https://github.com/Godones/OS2022-study