北京理工大學

Alien OS

陈林峰　　林 晨　　指导老师：陆慧梅

Why Alien?

北京理工大學

- 接触并尝试做了 rcore 实验，产生了对os的兴趣
- 一直都在学习现成的os
- 做过一些跟os相关的内容(碎片化的工作)
  - 内存管理
  - 文件系统
  - 操作系统移植
- 想要实现一个属于自己实现的os，整合已有工作
- os模块化

区域赛 2023.2-2023.6 [排名链接]

北京理工大学

10	202310007101563	Alien/ 北京理工大学	2023-05-29 18:37:06	5	102.0000

决赛一阶段 2023.6-2023.8 **排名链接:** QEMU starfive2 Unmatched u740

QEMU

3	20231000710156 3	Alien/ 北京理工大学	29	2023-07-28 09:07:45	54.0	7.4773458471177 5	20.136314828674067	0.0	27.047302954840323	220.0	45.513128379993369	9.0	0.0	28.351404718514406	411.5255

starfive2

1	20231000 7101563	Alien/ 北京理工大学	10	2023-08-01 21:29:18	54.0	7.934456402208149	32.67676128448016	0.0	32.25633065714811	220.0	50.93855458794903	9.0	0.0	17.334423239578797	424.1405

Unmatched u740

2	20231000 7101563	Alien/ 北京理工大学	2	2023-08-01 21:27:06	54.0	7.987423389729627	35.21007117245986	0.0	44.021122846126346	220.0	56.694515443296360	9.0	0.0	17.278144576953490	444.1913

现场赛

[排名链接]

北京理工大學

队伍	学校	开发板	提交时间	commit	得分														
					lmbench	iozone	iperf	libcbench	netperf	unixbench	cyclictest	busybox	libctest	lua	阶段1映射分数	interrupt	copy	阶段2映射分数	映射总分
Titanix	哈尔滨工业大学（深圳）	HiFive Unmatched 开发板	2023/8/20 14:42:18	851948ec	56.89	38.35	7.34	32.38	8.08	40.21	4.00	54.00	220.00	9.00	100.00	25.00	25.00	100.00	200.00
你说对不队	河南科技大学	华山派-CV1813H	2023/8/20 13:33:21	78e3d7b4	48.58	34.13	7.87	28.73	8.85	34.87	5.67	54.00	220.00	9.00	93.33	25.00	25.00	100.00	193.33
Alien	北京理工大学	VisionFive 2星光二代板	2023/8/20 17:47:53	59283c4a	48.95	31.60	2.00	31.13	7.80	29.82	7.92	53.00	220.00	9.00	86.67	25.00	25.00	100.00	186.67

开发历程

北京理工大學

2月 - 6月 (commit: 167)

目标：支持基本的系统调用，搭建基本内核

- 根据仓库中给出的系统调用说明实现各个系统调用
- 实现基本的内存管理
- 实现单核下的进程模型
- 加入文件系统支持

6月 - 8月 (commits: 242)

开发日志

现实情况：由于这届比赛中，这一阶段的测试程序由上一届的一个增加到了八九个，并且一些测试程序对系统的要求较高(lmbench/unixbench)

目标：尽可能的通过所有测试，首先保证正确性，再考虑提高性能，并且尝试添加网络支持

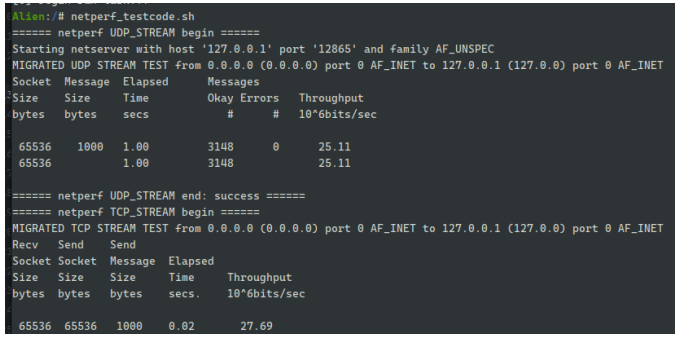
- 增加线程模型
- 增加多核支持
- 改进页表实现
- 改进内存管理
- 加入信号机制
- 加入同步互斥
- 文件系统改进
- 从libc-test开始，逐个添加syscall
- 增加真实开发板支持

开发历程

北京理工大學

8月 - 目前 (决赛第二阶段)(commit:79)

开发日志



在完成决赛第一阶段后，我们将目标放在让OS支持真实应用上，并且为OS加上网络和GUI的支持

目前 Alien 支持的应用包括：

- Lua
- Bash
- Busy Box
- Redis
- Sqlite3
- Slint 框架
- Embedded graphic 框架

Lua

BASH
THE BOURNE-AGAIN SHELL

BusyBox

redis

SQLite3
SQL
database
engine

北京理工大學

目录

北京理工大學

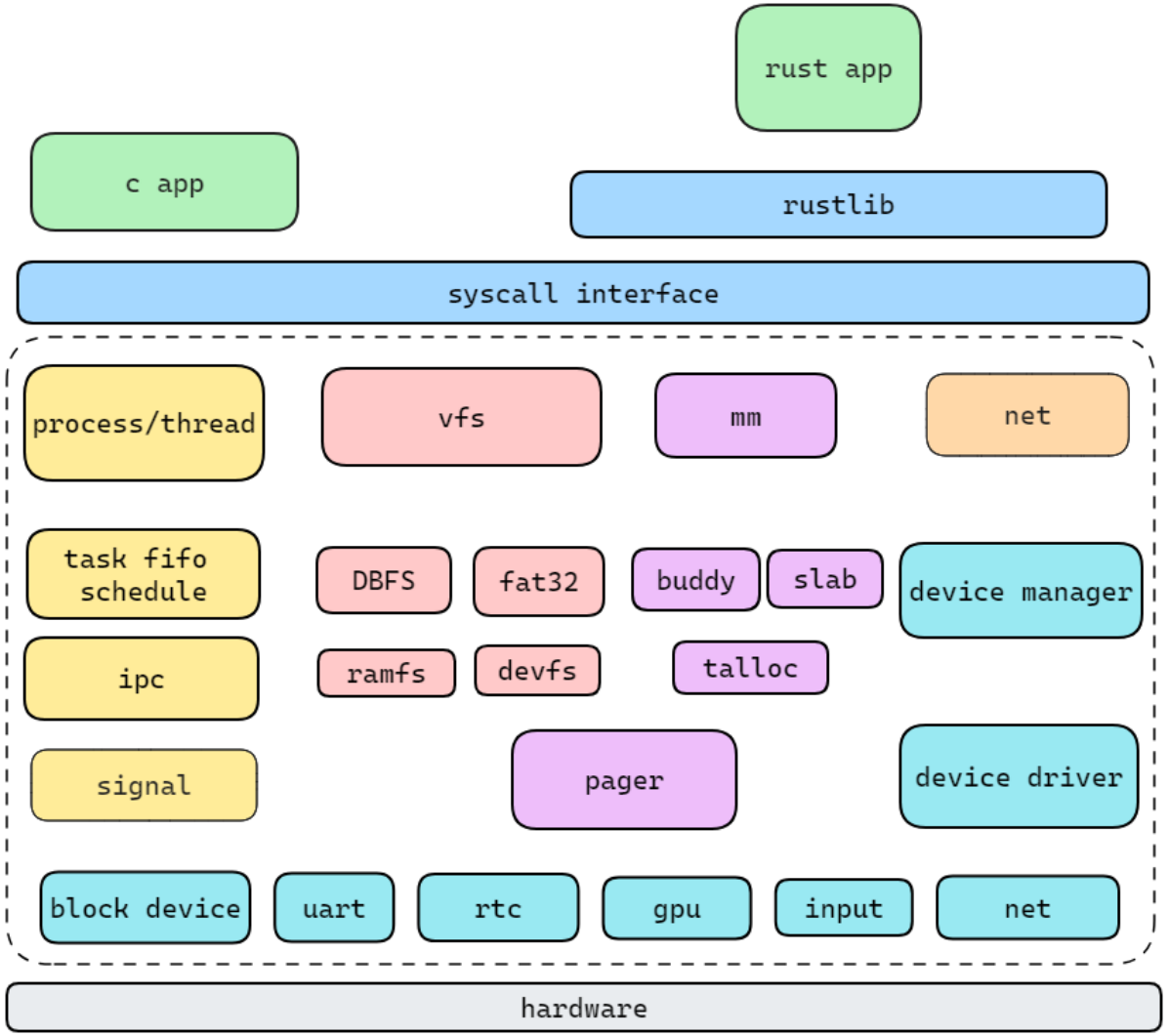
01

系统架构

系统结构

北京理工大學

相关文档



Alien 的基本特性

北京理工大學

- 宏内核
- 多核SMP支持
- 完善的文档(代码即文档/开发文档)
- 模块化/低耦合
  - 内核中的各个子部分由独立的外部模块构成
  - 有的模块完全自己实现
  - 有的模块使用来自社区实现
  - 有的模块是修改社区已有实现以适配内核需求
- 丰富的系统调用（128个）

Functions

clone	一个系统调用，用于创建一个子进程。
current_cpu	get the current cpu info
current_task	get the current_process
current_trap_frame	get the current process's trap frame
current_user_token	get the current process's token (root ppn)
do_brk	一个系统调用，用于改变堆区的大小(目前仅可以增加堆区大小)
do_exec	一个系统调用，用于执行一个文件。
do_exit	一个系统调用，用于终止进程。
do_suspend	一个系统调用，用于使当前正在运行的进程让渡CPU。
exit_group	一个系统调用，退出当前进程(进程组)下的所有线程(进程)。
get_pgid	(待实现)获取进程组的id。目前直接返回0。
get_pid	获取当前正在运行task的pid号。在Alien中pid作为线程组的标识符，位于同一线程组中的线程的pid相同。
get_ppid	获取当前正在运行task的ppid号，即父task的pid号。
get_tid	获取当前正在运行task的tid号。在Alien中tid作为task的唯一标识符。
getegid	(待实现)获取有效用户组 id，即相当于哪个用户组的权限。在实现多用户组权限前默认为最高权限。目前直接返回0。
geteuid	(待实现)获取有效用户 id，即相当于哪个用户的权限。在实现多用户权限前默认为最高权限。目前直接返回0。
getgid	(待实现)获取用户组 id。在实现多用户权限前默认为最高权限。目前直接返回0。
getuid	(待实现)获取用户 id。在实现多用户权限前默认为最高权限。目前直接返回0。
init_per_cpu	
init_process	put init process into process pool
prlimit64	一个系统调用，用于修改进程的资源限制。
set_pgid	(待实现)设置进程组的id。目前直接返回0。
set_sid	创建一个新的session，并使得使用系统调用的当前task成为新session的leader，同时也是新进程组的leader(待实现)
set_tid_address	一个系统调用，用于修改进程clear_child_tid的值，同时返回进程的tid。
wait4	一个系统调用，用于父进程等待某子进程退出。

Alien 使用的模块

北京理工大學

自己开发的模块

- os-module/doubly-linked-list
- os-module/page-table
- os-module/rtrace
- os-module/visionfive2-sd
- os-module/elfinfo
- os-module/preprint
- os-module /rslab
- os-module/rtc
- Godones/rvfs
- Godones/fat32-vfs
- Godones/dbfs2
- Godones/jammdb
- module/basemachine
- module/gmanager
- module/Input2event
- module/pager
- module/plic
- module/simplegui
- module/simple-net
- module/syscall-table
- module/syscall
- module/uart
- module/schedule

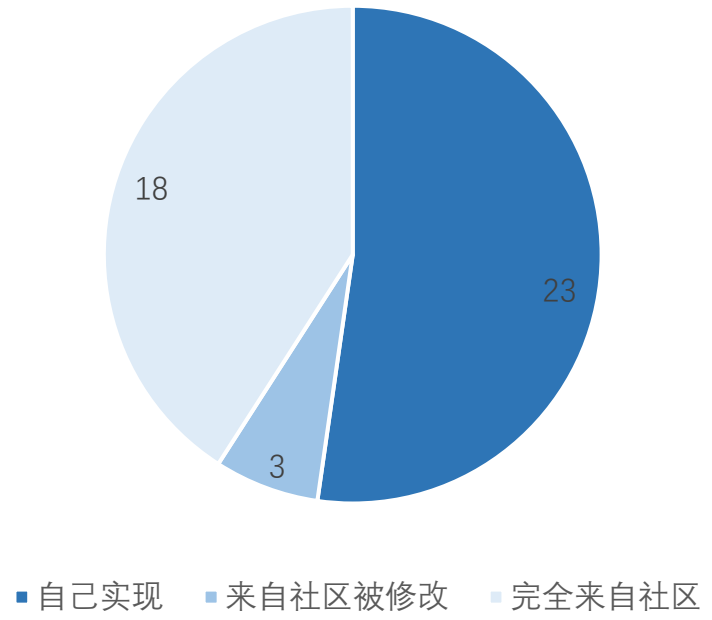
简单修改后得到的模块

- kernel-sync
- pci-rs
- virtio-input-decoder

crates.io

- buddy
- talloc
- riscv
- ……

Alien 中使用的模块类型分布



02

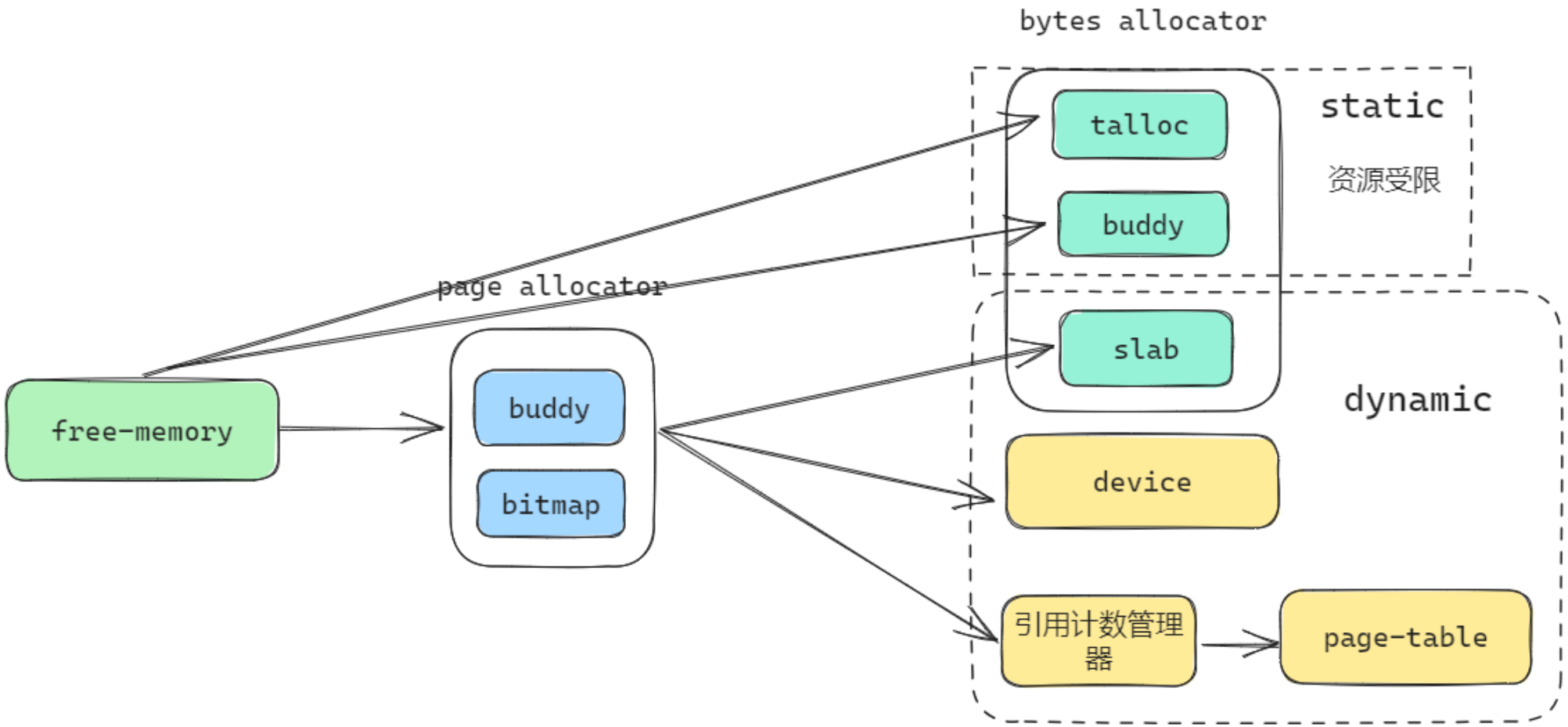
内核设计

内存管理

北京理工大學

内存管理一条龙

相关文档



内存管理

北京理工大學

参考：linux的内存管理方式

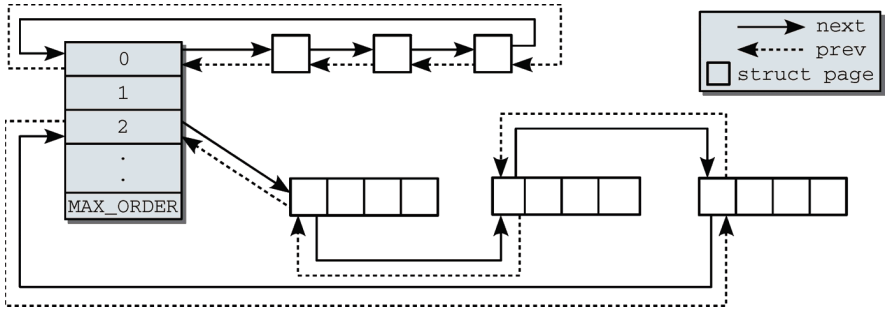
物理页管理：避免内存碎片化

bitmap

1	0	1
1	1	1
0	0	0
1	1	1
1	1	1

```
pub struct Bitmap<const N: usize> {
    /// Current number of allocated pages
    current: usize,
    /// Maximum number of pages
    max: usize,
    /// The bitmap data
    data: [u8; N],
    /// start page
    start: usize,
}
```

buddy




```
pub struct Zone<const MAX_ORDER: usize> {
    /// The pages in this zone
    manage_pages: usize,
    start_page: usize,
    free_areas: [FreeArea; MAX_ORDER],
}
```

```
#[derive(Copy, Clone)]
struct FreeArea {
    /// The number of free pages in this free area
    free_pages: usize,
    /// The list of free pages in this free area
    list_head: ListHead,
}
```

内存管理

北京理工大學

小内存分配：充分使用内存

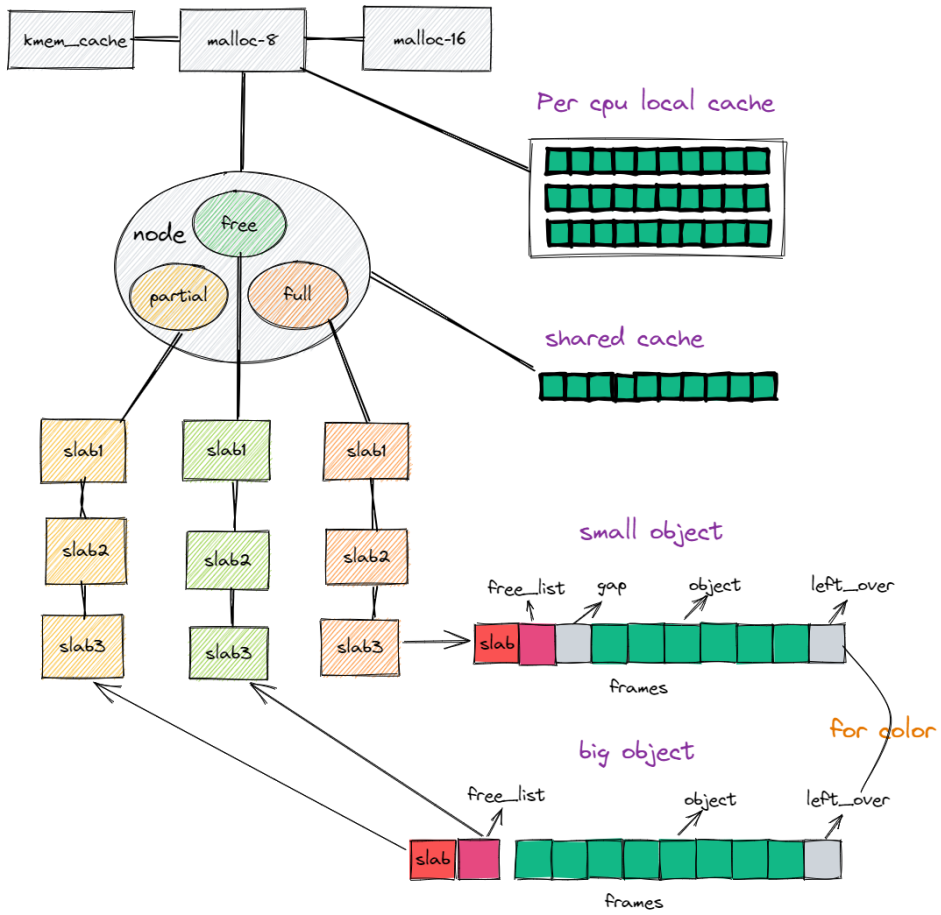
[相关文档]

- 多核支持
- 本地缓存
- 8bytes-8MB
- 可伸缩

option

- Buddy system
- Talloc

slab



内存管理 – 优化手段

北京理工大學

1. Lazy Allocation

对于应用程序的栈、堆、mmap区域的分配推迟到访问时，大大减少不必要的内存分配。

例: 大多数测试对栈的需求不高，但是有几个程序需要数MB的栈空间

2. COW

使用引用计数管理page-table的物理页使用情况，可以在父子进程之间共享物理页，并在发生写操作时进行复制

3. 条件编译

针对不同平台设置不同的参数，对于内存有限的qemu，减少设备缓存的上限值，对于starfive2/unmatched调高设备缓存的上限以获得更好的性能

4. 页面回收

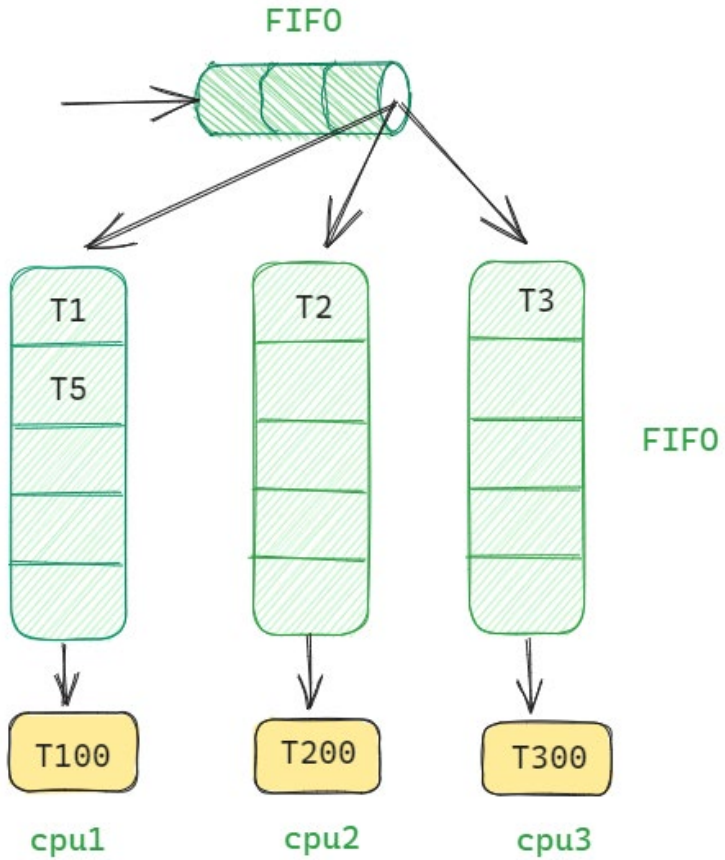
在使用slab做分配器时，当物理页紧缺时，可以向分配器回收空闲页

进程管理

北京理工大學

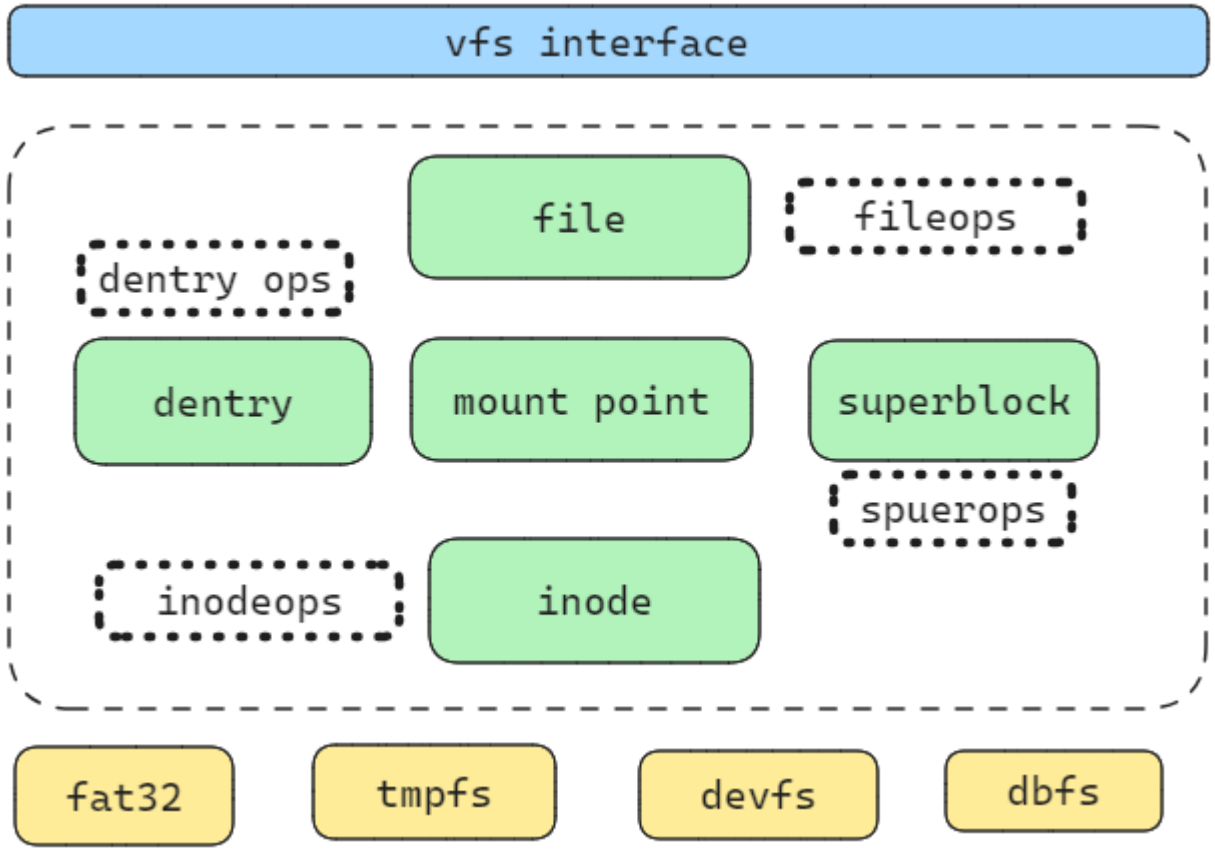
参考linux的做法，将进程和线程统一用一个task-struct进行管理，利用rust语言的优势，使用Arc进行资源的共享

1. 进程/线程==task-struct
2. Per-cpu队列，无锁
3. FIFO调度
4. 时间片轮转
5. 通过clone参数控制各个资源的共享关系



文件系统

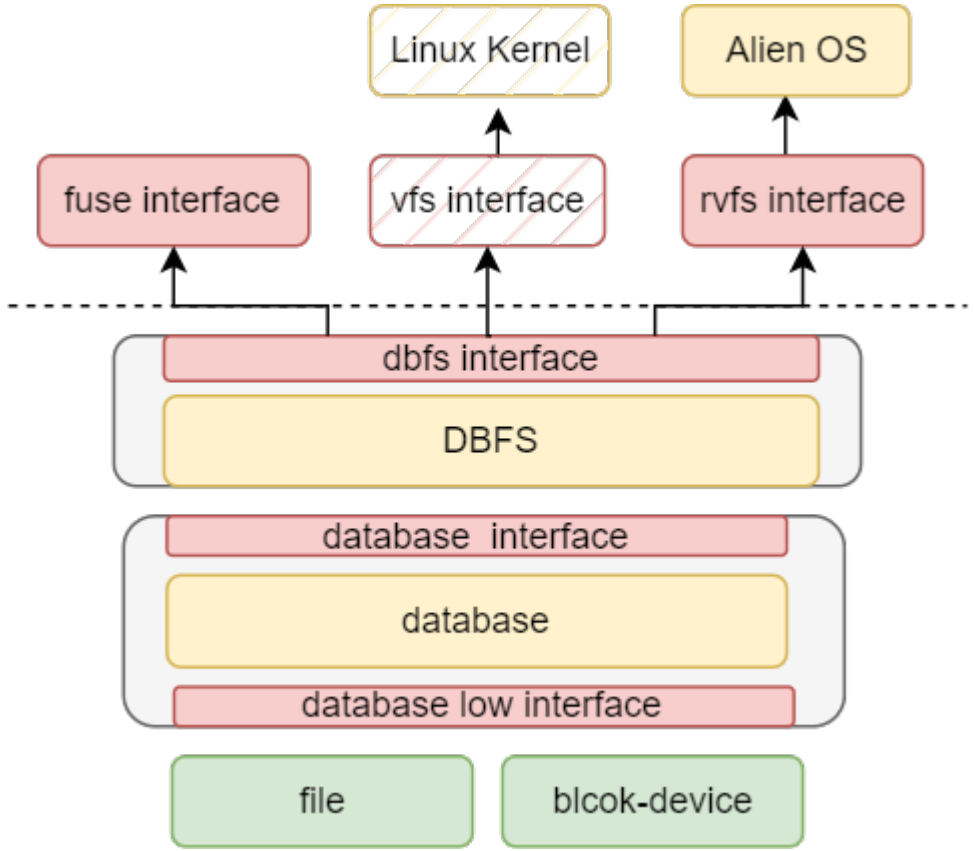
北京理工大學

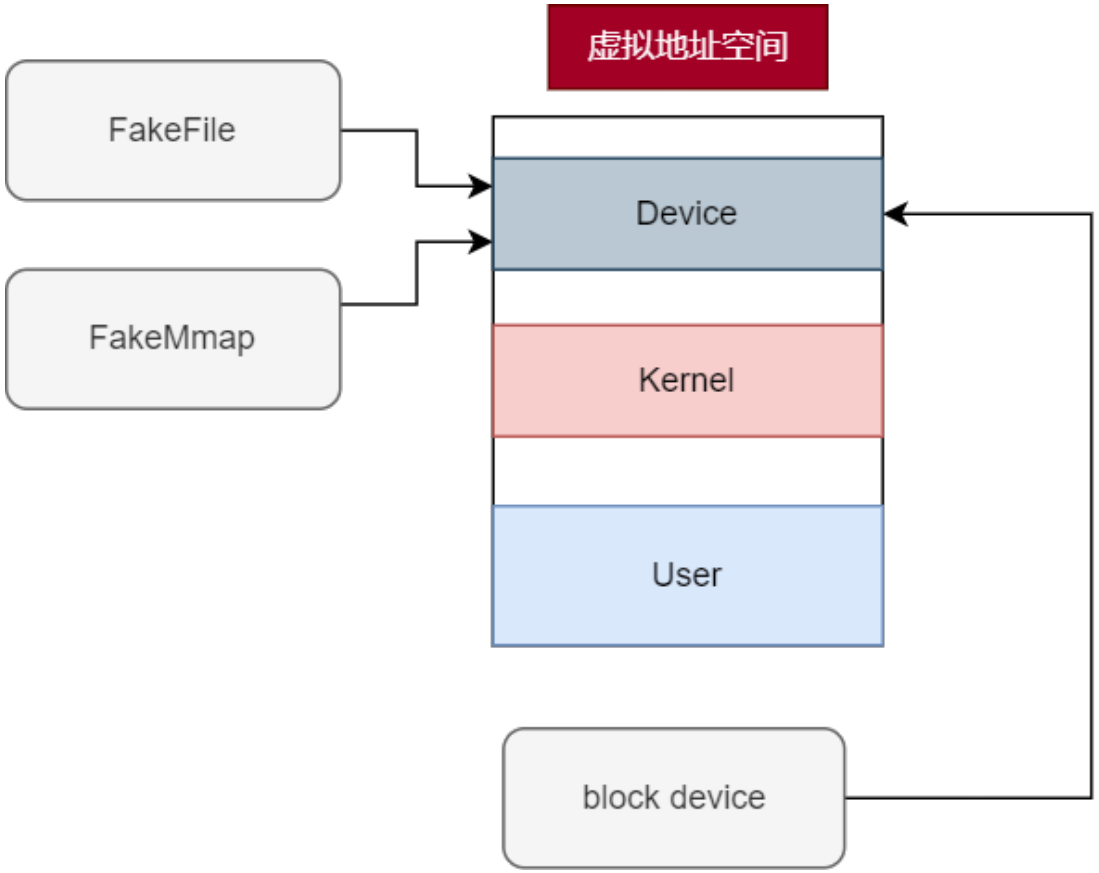


- 完整的vfs实现，包含绝大部分linux中vfs的功能(4000行以上代码)
- 独立模块可复用
- File/Dentry 会被缓存，不需要重复磁盘查找
- 允许重复挂载
- 多个文件系统支持
  - Fat32、Dbfs、Tmpfs、Devfs、Rootfs
- LRU缓存
- 多块读写(4096/512)

文件系统 - DBFS

北京理工大學





文件系统 - DBFS

北京理工大學

- 得益于vfs的抽象，dbfs只需要实现定义的各个接口即可接入vfs
- 支持绝大部分文件系统功能
  - 软硬链接
  - 属性
- 高性能

表 4: Alien OS 测试结果

(MB/s)/FS	DBFS	FAT32
顺序写 (1MB)	32MB/s	15MB/s
顺序写 (Re)	384MB/s	104MB/s
顺序读 (1MB)	1000MB/s	110MB/s
顺序读 (4KB)	27MB/s	50MB/s
随机写 (1MB)	386MB/s	56MB/s
随机读 (1MB)	1066MB/s	58MB/s

Sd 卡驱动

北京理工大學

存在的问题

1. Starfive2的sd卡使用sdio协议，与qemu/unmatched 协议不一样
2. 开发板不提供手册，寄存器资料也不全
3. 网络上只能从linux/uboot中查看源代码，无有效的实现可供参考

解决方案

- 从另外一组队伍 哈工大MankorOS 寻求帮助，得到了一个可用的手册
- 参考 STM32 视频，熟悉sdio的工作方式
- 参考 starfive2 给出的寄存器说明和队伍 哈工大MankorOS 的实现
- 遵循手册的说明而不是标准的 sdio 过程

```
[1] bus_mode(DMA): BusModeReg {
    reserved: 0,
    pbl: 2,
    de: 1,
    dsl: 0,
    fd: 1,
    swr: 0,
}
[1] dma_desc_base: 0xff735dc0
[1] clock_divider: ClockDividerReg {
    clk_divider3: 0,
    clk_divider2: 0,
    clk_divider1: 0,
    clk_divider0: 2,
}
[1] reset clock success
[1] reset fifo success
[1] reset dma success
[1] card is in idle state
[1] card voltage: 0x1aa
[1] card version: 2.0
[1] card is ready
[1] card is high capacity
[1] cid: mid:3 oid:SD pnm:SD prv:8.5 psn:1566488968 mdt:2021-10
[1] rca: 0xaaaa
[1] status: 10000000000111000000000000110010
[1] Bus width supported: 101
[1] test read, try read 0 block
[1] sd header 16bytes: [eb, 58, 90, 6d, 6b, 66, 73, 2e, 66, 61, 74, 0, 2, 20, 20, 0]
[1] init sd success
[1] buf: [eb, 58, 90, 6d, 6b, 66, 73, 2e, 66, 61, 74, 0, 2, 20, 20, 0]
[1] name: ".\0\0\0\0\0\0\0\0\0\0\0"
[1] name: "root.txt"
[1] read: 11
```

其它驱动

北京理工大學

UART

相关文档

- 使用S态驱动，不需要进入SBI，减少特权级切换
- 异步读写(中断+读写缓冲区+等待队列)
- 多平台支持
  - Qemu/unmatched的寄存器位宽为1byte
  - Starfive2是四字节

PLIC

相关文档

- 根据PLIC规范实现
- 各厂商可以根据规范调整
  - Unmatched/starfive2的第一个核只有M态
  - Qemu模拟的所有核都具有M/S态
  - 寄存器发布不均匀
- 抽象数据结构，一个PLIC可以适应不同的实现

自动化与测试

北京理工大學

自动化系统调用生成

```
#[syscall_func(116)]
pub fn syslog(log_type: u32, buf: usize, len: usize) -> isize {
    let log_type = SyslogAction::try_from(log_type);
```

```
pub fn register_all_syscall(){
        let mut table = Table::new();
        register_syscall!(table,
        (2002, sys_event_get),
        (25, fcntl),
        (29, ioctl),
        (88, utimensat),
        (48, faccessat),
        (52, chmod),
        (53, chmodat),
        (55, fchown),
```

- 宏+编译前代码生成

多种方法堆栈回溯

1. 启用fp指针（降低执行效率）
2. 使用DWARF(标准方案)
3. 根据编译器的代码生成规律

自动化调试信息生成

- 从磁盘读取
  - 编译完源代码，生成符号信息保存到磁盘上
  - Panic时读取磁盘
- 两阶段编译
  - 符号信息位于内核数据区
  - Panic读内存

03

扩展功能

网络支持

北京理工大学

这次的测试中多出了两个网络相关的netperf/iperf，但我们并没有找到很多往届参赛选手在网络支持方面的资料。

Start from ~~scratch~~? Or Smoltcp?

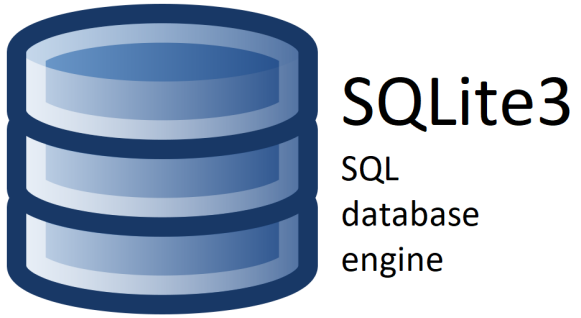
- 从零开始既没有时间也没有必要
- 初赛阶段对网络支持理解不够深
  同时man手册解释不清
- 支持相关syscall，但是无法通过
  试，内核引入大量的全局变量

```
Alien:/# netperf_testcode.sh
====== netperf UDP_STREAM begin ======
Starting netserver with host '127.0.0.1' port '12865' and family AF_UNSPEC
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 127.0.0.1 (127.0.0) port 0 AF_INET
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

 65536    1000   1.00         3148      0      25.11
 65536           1.00         3148             25.11

====== netperf UDP_STREAM end: success ======
====== netperf TCP_STREAM begin ======
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 127.0.0.1 (127.0.0) port 0 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 65536  65536   1000     0.02       27.69
```

应用支持

北京理工大學

在区域赛阶段，本次比赛添加了更多的测试，里面就包含了 Lua 和 BusyBox 两个应用，在后期，我们对 BusyBox 的子命令做了更多支持。



GUI支持 - slint

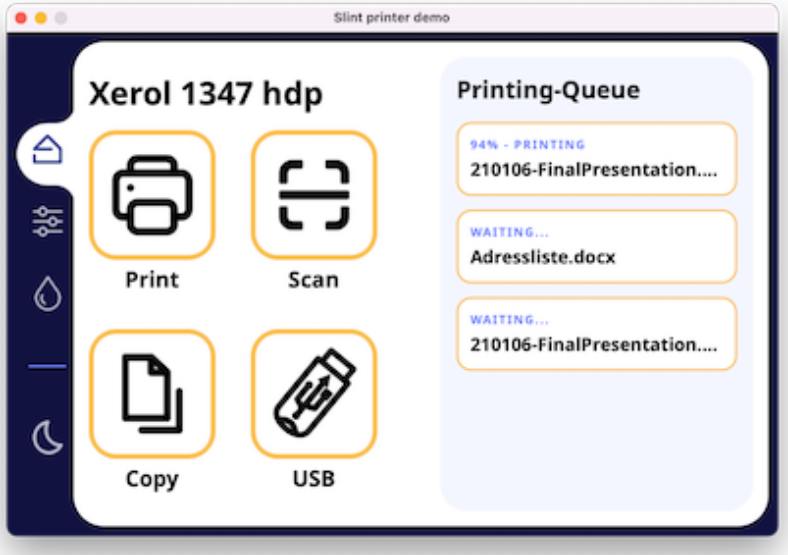
北京理工大學

Why GUI ?

- 很少听说有队伍做过 GUI 的尝试
- 人机交互对普通人来说很重要

目前 Alien 支持的功能：

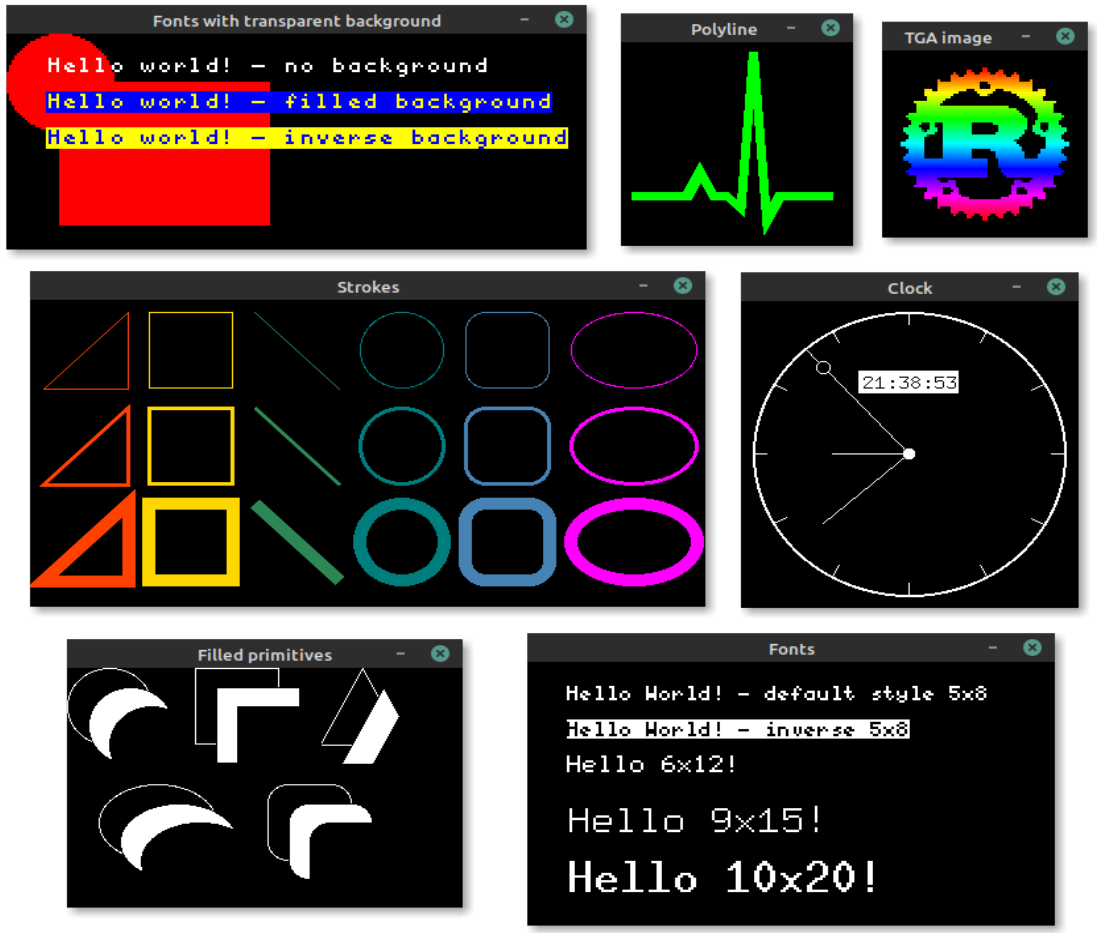
- 处理键盘/鼠标的输入
- 增加 slint 嵌入式 GUI 的支持
- 尽可能高效地传递事件
  - 一次从内核读取尽可能多的事件
  - 内核只存储规定数量的事件，事件频繁将被丢弃



GUI支持 - embedded graphic

北京理工大學

- Embedded graphic 中只包含了基本的图形绘制功能，不提供任何窗口部件和管理
- 自由度更高，但需要使用者自定义大多数控件
- 实现了一些基础部件
  - 窗口
  - 按钮
  - 状态栏
  - 文本框
- 由基础部件构成的复杂控件
  - 终端
  - 桌面
  - 游戏



站在前辈的肩膀上前进

北京理工大學

- 信号 sigreturn 可能不会返回原有上下文的问题
  - 来自 pthread-cancel
- Bash允许其它程序时输入不回显的问题
  - 运行 sqlite3 时发现
- 设置AUX数组
- 设置环境
- 浮点寄存器保存
- SD卡驱动

NPUcore 链接

Maturin 链接

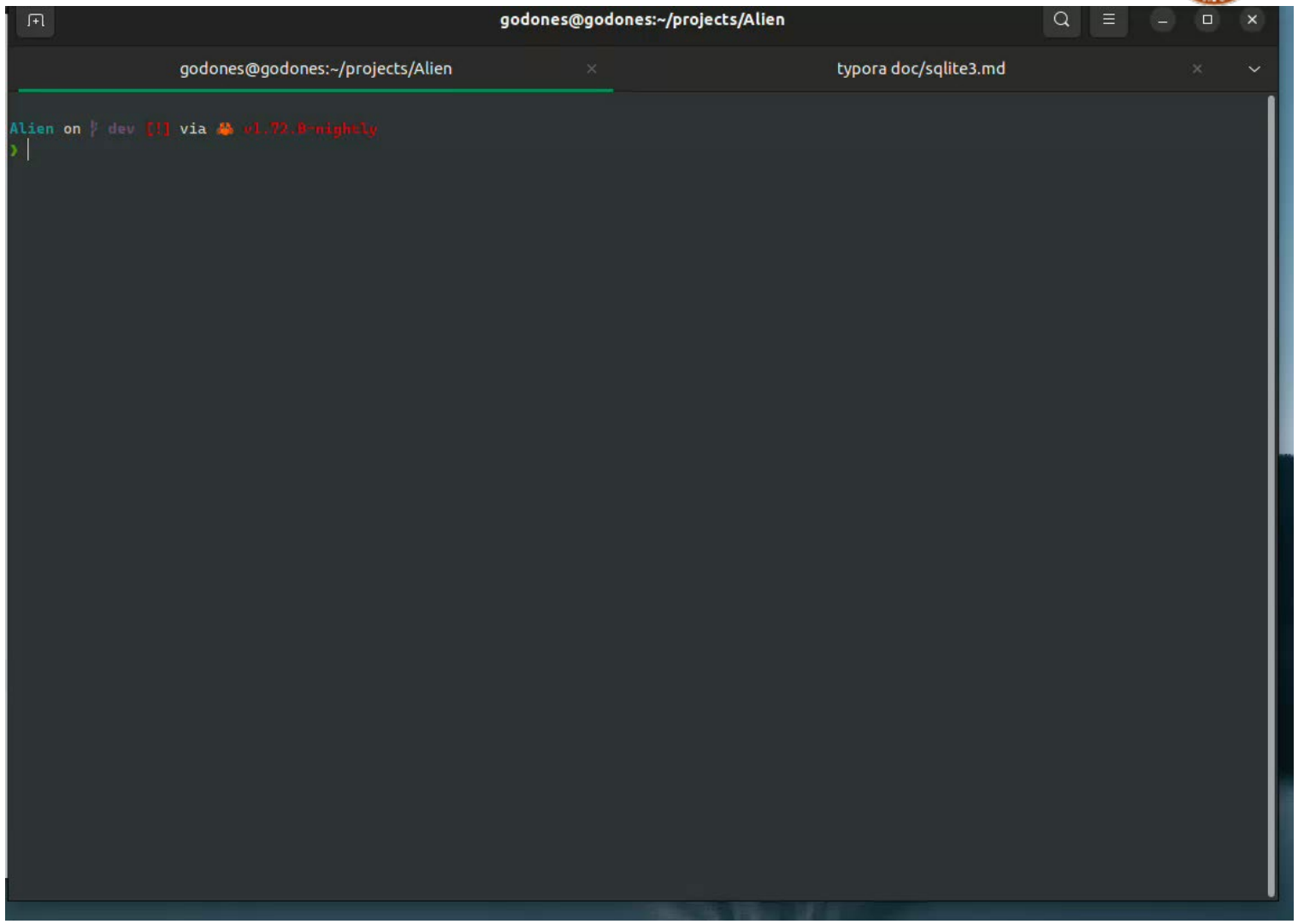
图漏图森破 链接

04

演示视频

视频演示

北京理工大學


godones@godones:~/projects/Alien
godones@godones:~/projects/Alien
typora doc/sqlite3.md
Alien on dev [!] via v1.72.0-nightly

05

总结与展望

总结与展望

北京理工大學

项目总结：

- 投入时间和精力
- 假定硬件不会出错
- 需要高效的debug方法
  - 多线程
  - 网络
  - 可视化
- 利用可用的资源而不用重复造轮子
- 丰富的文档有利于他人理解
- 感谢前辈们的踩坑经验

未来工作：

- Async支持
- 更加模块化的内核
- 更完善的网络支持
- GUI改进
- Rust std支持
- 多体系结构/多平台支持

Thanks!