操作系统 实验6

> 21301114 俞贤皓
>
> 环境（实验0~2, 4~6）： Arch Linux 6.5.3-arch1-1
>
> 环境（实验3~4）： Ubuntu 22.04.3 LTS (WSL)

1. 实验步骤

1.1 在内核中支持动态分配内存

- 根据文档实现代码
  - ```
    YXH_XianYu ~/b/O/G/expt6 (main)> cd os
    YXH_XianYu ~/b/O/G/e/os (main)> vim Cargo.toml
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/main.rs
    YXH_XianYu ~/b/O/G/e/os (main)> mkdir src/mm
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/heap_allocator.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/main.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/heap_allocator.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/main.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/config.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/mod.rs
    ```

1.2~1.4 地址数据结构、页表、物理帧

- 根据文档实现代码
  - ```
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/address.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/address.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/main.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/page_table.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim Cargo.toml
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/page_table.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/mod.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/linker.ld
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/config.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/frame_allocator.rs
    YXH_XianYu ~/b/O/G/e/os (main)> mkdir src/sync
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/sync/up.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/sync/up.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/sync/mod.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/main.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/config.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/mm/mod.rs
    YXH_XianYu ~/b/O/G/e/os (main)> vim src/config.rs
    YXH_XianYu ~/b/O/G/e/os (main)>
    ```
- 执行 `make run`
  - 出现错误，如下图所示：
    -

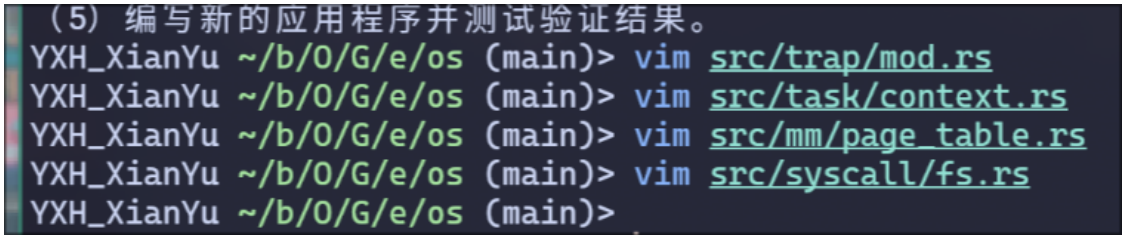
```
root@88a1fcca9270 /m/e/os# make run
  Downloaded buddy_system_allocator v0.6.0 (registry `tuna`)
  Downloaded spin v0.7.1 (registry `tuna`)
  Downloaded 2 crates (29.0 KB) in 3.73s
   Compiling spin v0.7.1
   Compiling os v0.1.0 (/mnt/expt6/os)
   Compiling buddy_system_allocator v0.6.0
error: expected one of `!` or `::`, found `(`
  --> src/main.rs:21:9
   |
21 | mm::init();
   |         ^ expected one of `!` or `::`

error: could not compile `os` due to previous error
make: *** [Makefile:34: kernel] Error 101
```

- 查看 `main.rs: 21`，发现我在非函数中调用了一个普通函数，鉴定为python写多了，所以将 `mm:init();` 移至 `main` 函数中

- 重新编译，成功

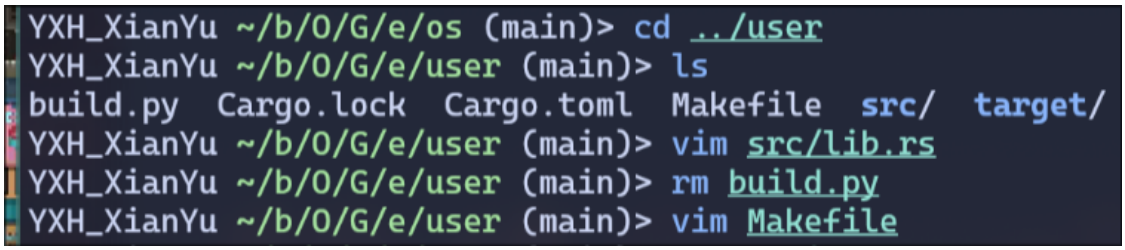
```
warning: `os` (bin "os") generated 7 warnings
    Finished release [optimized] target(s) in 5.97s
[rustsbi] RustSBI version 0.3.1, adapting to RISC-V SBI v1.0.0
.______       __    __      _______.___________.  _______..______   __
|   _  \     |  |  |  |    /       |           | /       ||   _  \ |  |
|  |_)  |    |  |  |  |   |   (----`---|  |----`|   (----`|  |_)  ||  |
|      /     |  |  |  |    \   \       |  |      \   \    |   _  < |  |
|  |\  \----.|  `--'  |.----)   |      |  |  .----)   |   |  |_)  ||  |
| _| `._____| \______/ |_______/       |__|  |_______/    |______/ |__|
[rustsbi] Implementation     : RustSBI-QEMU Version 0.2.0-alpha.2
[rustsbi] Platform Name      : riscv-virtio,qemu
[rustsbi] Platform SMP       : 1
[rustsbi] Platform Memory    : 0x80000000..0x88000000
[rustsbi] Boot HART          : 0
[rustsbi] Device Tree Region : 0x87e00000..0x87e00e66
[rustsbi] Firmware Address   : 0x80000000
[rustsbi] Supervisor Address : 0x80200000
[rustsbi] pmp01: 0x00000000..0x80000000 (-wr)
[rustsbi] pmp02: 0x80000000..0x80200000 (---)
[rustsbi] pmp03: 0x80200000..0x88000000 (xwr)
[rustsbi] pmp04: 0x88000000..0x00000000 (-wr)
heap_test passed!
last 661 Physical Frames.
FrameTracker:PPN=0x8056b
FrameTracker:PPN=0x8056c
FrameTracker:PPN=0x8056d
FrameTracker:PPN=0x8056e
FrameTracker:PPN=0x8056f
FrameTracker:PPN=0x8056f
FrameTracker:PPN=0x8056e
FrameTracker:PPN=0x8056d
FrameTracker:PPN=0x8056c
FrameTracker:PPN=0x8056b
frame_allocator_test passed!
[kernel] Hello, world!
[kernel] Trap initialized.
```

1.5~1.7 多级页表、地址空间、分时多任务

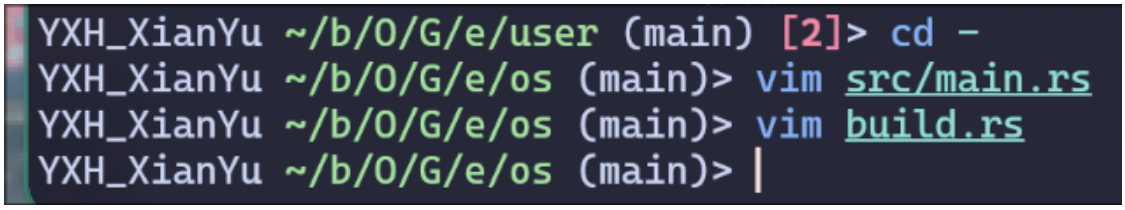
- 根据文档实现代码
  - 

  - 因为不小心把实验手册cat到终端里，导致历史信息翻不到，所以截图信息很少

1.8 修改应用程序

- 根据文档实现代码
  - 

1.9 修改main.rs

- 根据文档实现代码
  - 

1.10 执行!

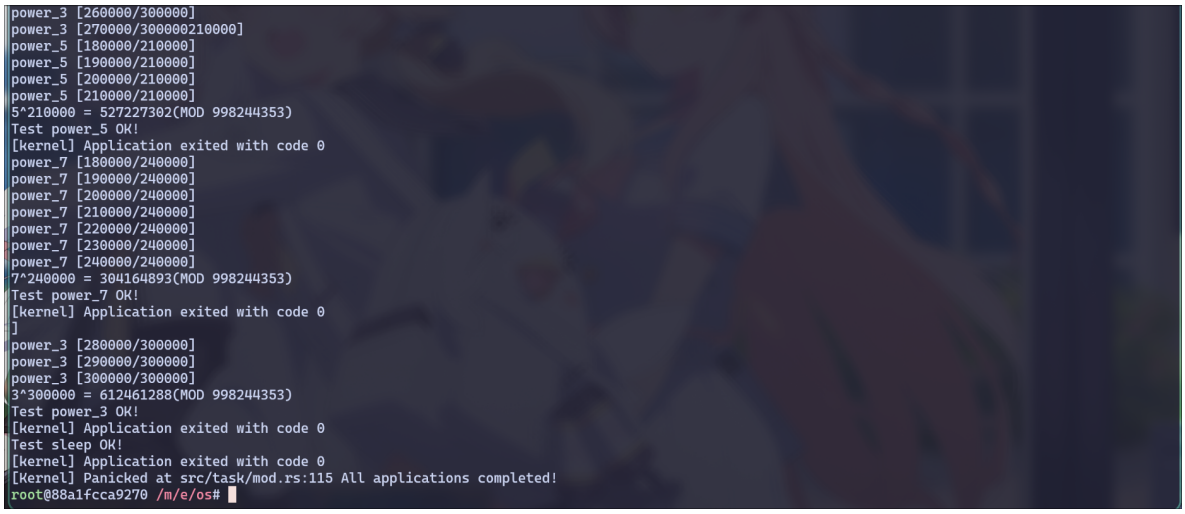
- 遇到了很多错误，对其进行修复
- 修复完成后:
-

```
root@88a1fcca9270 /m/e/os# make run
(rustup target list | grep "riscv64gc-unknown-none-elf (installed)") || rustup target add riscv64gc-unknown-none-elf
riscv64gc-unknown-none-elf (installed)
cargo install cargo-binutils
    Updating `ustc` index
     Ignored package `cargo-binutils v0.3.6` is already installed, use --force to override
rustup component add rust-src
info: component 'rust-src' is up to date
rustup component add llvm-tools-preview
info: component 'llvm-tools-preview' for target 'x86_64-unknown-linux-gnu' is up to date
make[1]: Entering directory '/mnt/expt6_std/user'
    Finished release [optimized] target(s) in 0.01s
rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/00power_3 --strip-all -O binary  target/riscv64gc-unknown-no
ne-elf/release/00power_3.bin;  rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/01power_5 --strip-all -O bina
ry  target/riscv64gc-unknown-none-elf/release/01power_5.bin;  rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/releas
e/02power_7 --strip-all -O binary  target/riscv64gc-unknown-none-elf/release/02power_7.bin;  rust-objcopy --binary-architecture=riscv64 target/ri
scv64gc-unknown-none-elf/release/03sleep --strip-all -O binary  target/riscv64gc-unknown-none-elf/release/03sleep.bin;
make[1]: Leaving directory '/mnt/expt6_std/user'
   Compiling os v0.1.0 (/mnt/expt6_std/os)
    Finished release [optimized] target(s) in 0.90s
[rustsbi] RustSBI version 0.2.0-alpha.3
.______       __    __      _______.___________.  _______..______   __
|   _  \     |  |  |  |    /       |           | /       ||   _  \ |  |
|  |_)  |    |  |  |  |   |   (----`---|  |----`|   (----`|  |_)  ||  |
|      /     |  |  |  |    \   \       |  |      \   \    |   _  < |  |
|  |\  \----.|  `--'  |.----)   |      |  |  .----)   |   |  |_)  ||  |
| _| `._____| \______/ |_______/       |__|  |_______/    |______/ |__|

[rustsbi] Platform: QEMU (Version 0.2.0)
[rustsbi] misa: RV64ACDFIMSU
[rustsbi] mideleg: 0x222
[rustsbi] medeleg: 0xb1ab
[rustsbi-dtb] Hart count: cluster0 with 1 cores
[rustsbi] Kernel entry: 0x80200000
[kernel] Hello, world!
last 613 Physical Frames.
```

-
```
last 613 Physical Frames.
.text [0x80200000, 0x8020b000)
.rodata [0x8020b000, 0x8020f000)
.data [0x8020f000, 0x8028a000)
.bss [0x8028a000, 0x8059b000)
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section
mapping physical memory
[kernel] back to world!
remap_test passed!
init TASK_MANAGER
num_app = 4
power_3 [10000/300000]
power_3 [20000/300000]
power_3 [30000/300000]
power_3 [40000/300000]
power_3 [50000/300000]
power_3 [60000/300000]
power_3 [70000/300000]
power_3 [80000/300000]
power_3 [90000/300000]
power_5 [10000/210000]
power_5 [20000/210000]
power_5 [30000/210000]
power_5 [40000/210000]
power_5 [50000/210000]
power_5 [60000/210000]
power_5 [70000/210000]
power_5 [80000/210000]
power_5 [90000/210000]
power_5 [100000/210000]
power_5 [110000/210000]
power_5 [120000/210000]
power_5 [130000/210000]
```

-
```
power_5 [140000/210000]
power_5 [150000/210000]
power_5 [160000/210000]
power_5 [170000/power_7 [10000/240000]
power_7 [20000/240000]
power_7 [30000/240000]
power_7 [40000/240000]
power_7 [50000/240000]
power_7 [60000/240000]
power_7 [70000/240000]
power_7 [80000/240000]
power_7 [90000/240000]
power_7 [100000/240000]
power_7 [110000/240000]
power_7 [120000/240000]
power_7 [130000/240000]
power_7 [140000/240000]
power_7 [150000/240000]
power_7 [160000/240000]
power_7 [170000/240000]
power_3 [100000/300000]
power_3 [110000/300000]
power_3 [120000/300000]
power_3 [130000/300000]
power_3 [140000/300000]
power_3 [150000/300000]
power_3 [160000/300000]
power_3 [170000/300000]
power_3 [180000/300000]
power_3 [190000/300000]
power_3 [200000/300000]
power_3 [210000/300000]
power_3 [220000/300000]
power_3 [230000/300000]
power_3 [240000/300000]
power_3 [250000/300000]
```

-



2. 思考问题

2.1 虚拟地址和物理地址的设计与实现

- 虚拟地址（`VirtAddr`）和物理地址（`PhysAddr`）通过Rust结构体定义，保证了内存管理的类型安全和清晰性。
- 引入虚拟页号（`VirtPageNum`）和物理页号（`PhysPageNum`），用于管理页式内存的基本单位，从而简化了地址转换和内存管理的复杂性。
- 实现了这些地址和页号类型与`usize`之间的相互转换。这种设计方法方便了虚拟地址和物理地址之间的转换，也便于与系统其他部分的交互。

2.2 物理帧的管理与分配

- `FrameTracker`用于监控和管理单个物理页的使用情况，确保每个物理页的使用都被适当地追踪。
- 采用栈式结构来管理物理内存，提供了一种简单而有效的方式来分配和回收物理页。
- 空闲物理页存储在一个栈中，分配时从栈顶取出一个页号，回收时则将页号放回栈顶。这种方法减少了内存分配的开销，提高了内存分配的效率。

2.3 内核与应用程序的地址空间实现

- `MemorySet`包含了内核的所有内存区域（如代码、数据和栈）。使用`MapArea`结构来表示特定的内存区域，通过页表条目的方式实现内存的映射。
- 在加载应用程序（基于ELF格式）时，根据程序头创建相应的内存映射。每个应用程序拥有独立的地址空间，实现了内存隔离和保护。
- 这种设计提高了系统的安全性和稳定性，防止了应用程序之间的相互干扰，确保了操作系统的稳定运行。

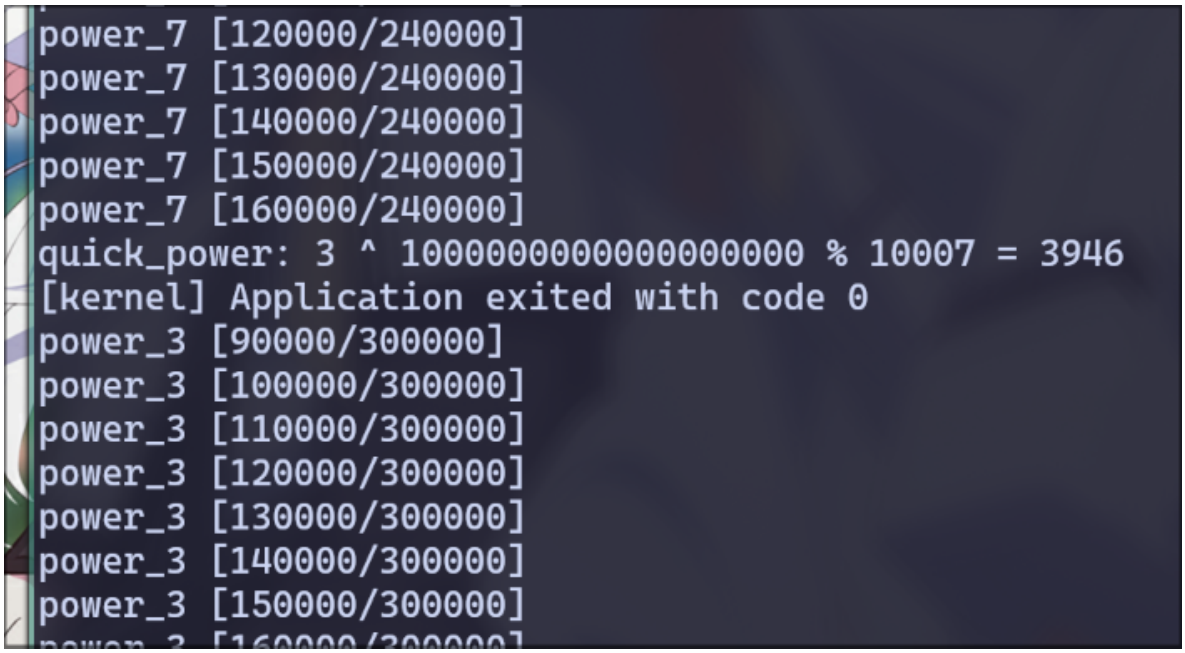
2.4 基于地址空间的分时多任务实现

- 使用`TaskControlBlock`和`TaskManager`来管理任务的生命周期，包括创建、运行、挂起和终止。
- 每个任务都有其独立的地址空间和上下文`TrapContext`，使得任务切换时能够保留和恢复任务的状态。

- 利用satp寄存器在不同任务之间切换其地址空间，同时在用户态和内核态之间进行必要的上下文切换。
- 通过时间片轮转调度算法，实现了多任务的“同时”运行，从而提高了CPU的利用率和系统的整体响应性。

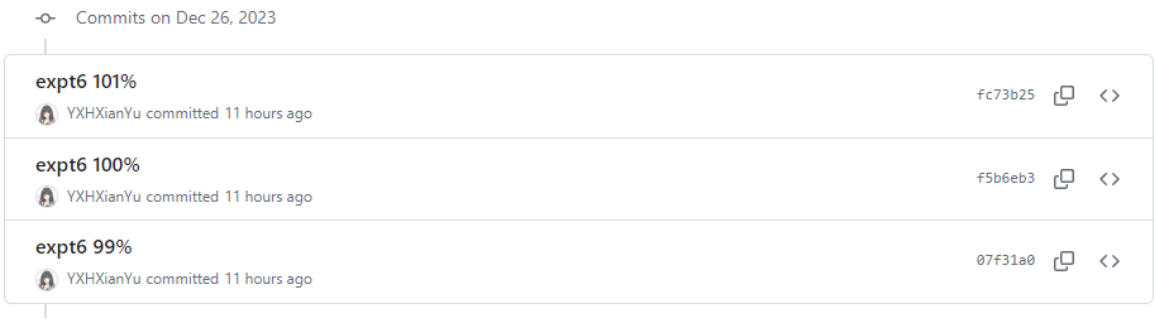
2.5 编写新的应用程序并测试验证结果

-
```rust
#![no_std]
#![no_main]
#[macro_use]
extern crate user_lib;

fn quick_power(base: u64, expo: u64, modu: u64) -> u64 {
    let mut a: u64 = base;
    let mut n: u64 = expo;
    let mut ans: u64 = 1;
    while n ≥ 1 {
        if n & 1 ≠ 0 { ans = ans * a % modu; }
        a = a * a % modu;
        n >>= 1;
    }
    ans
}

#[no_mangle]
fn main() -> i32 {
    const P: u32 = 3;
    const STEP: usize = 1_000_000_000_000_000_000; // 1e18
    const MOD: u32 = 10007;
    println!(
        "quick_power: {} ^ {} % {} = {}",
        P, STEP, MOD,
        quick_power(P as u64, STEP as u64, MOD as u64)
    );
    0
}
```
- 我编写了一个快速幂程序
-



- 快速幂程序第一个结束，执行时间最短。

3. Git提交截图

- 仓库链接
- 

4. 其他说明

- 无