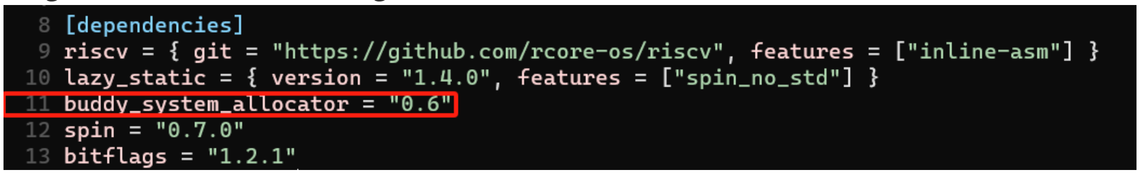
操作系统实验6

> 邓人嘉 21301032

一、实验步骤

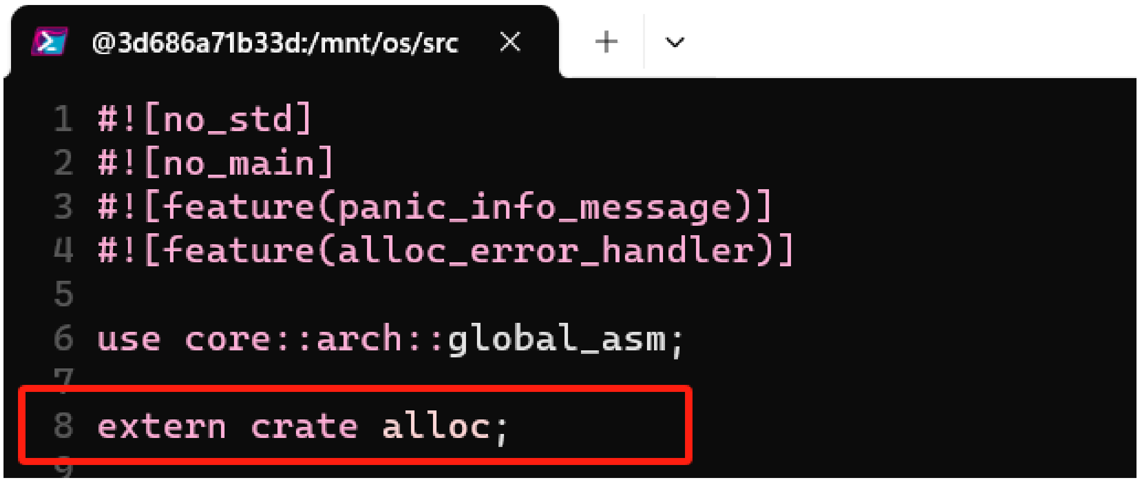
1.1 在内核中支持动态内存分配

- Cargo.toml中增加依赖（os/Cargo.toml）



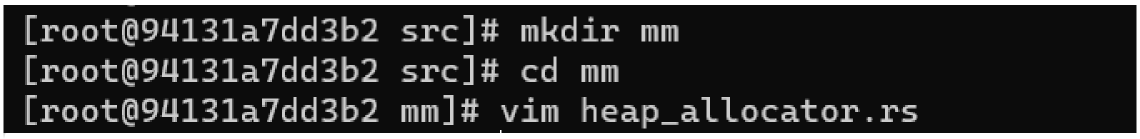
```
[dependencies]
riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
buddy_system_allocator = "0.6"
spin = "0.7.0"
bitflags = "1.2.1"
```

- 在main.rs中引入alloc库的依赖（os/src/main.rs）

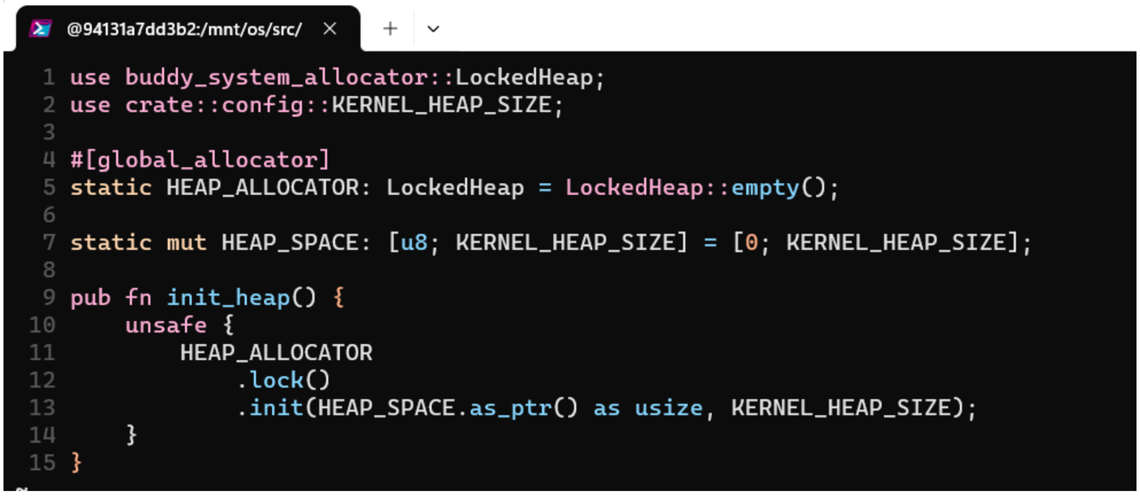


```
@3d686a71b33d:/mnt/os/src
#![no_std]
#![no_main]
#![feature(panic_info_message)]
#![feature(alloc_error_handler)]

use core::arch::global_asm;

extern crate alloc;
```

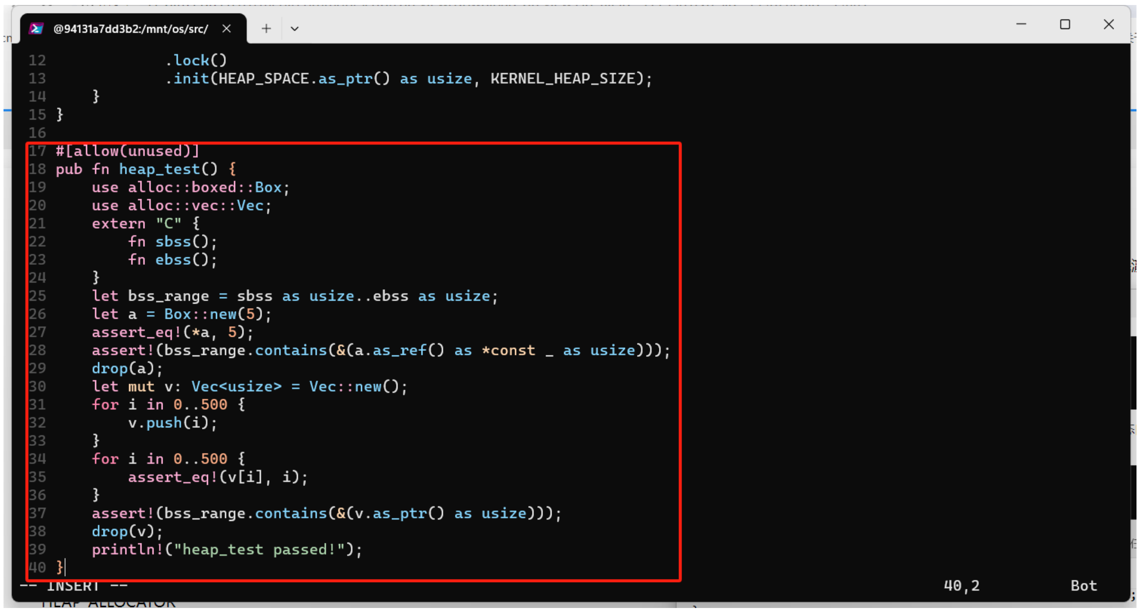
- 根据alloc留好的结构提供全局动态内存分配器，并处理动态内存分配失败的情况（os/src/mm/heap_allocator.rs）



```
[root@94131a7dd3b2 src]# mkdir mm
[root@94131a7dd3b2 src]# cd mm
[root@94131a7dd3b2 mm]# vim heap_allocator.rs
```

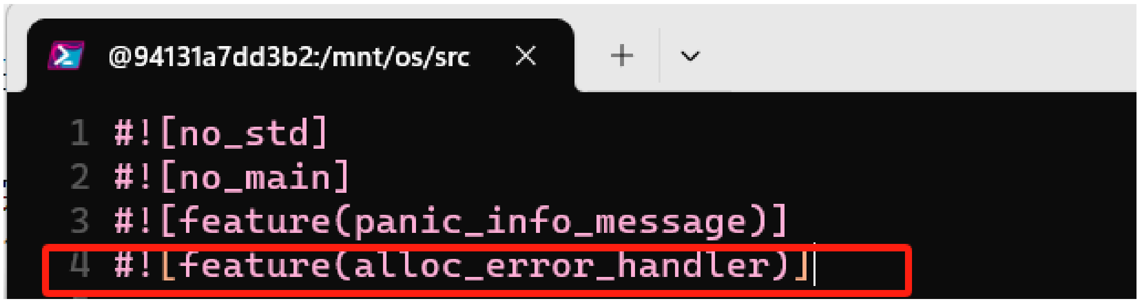


```
@94131a7dd3b2:/mnt/os/src/
use buddy_system_allocator::LockedHeap;
use crate::config::KERNEL_HEAP_SIZE;

#[global_allocator]
static HEAP_ALLOCATOR: LockedHeap = LockedHeap::empty();

static mut HEAP_SPACE: [u8; KERNEL_HEAP_SIZE] = [0; KERNEL_HEAP_SIZE];

pub fn init_heap() {
    unsafe {
        HEAP_ALLOCATOR
            .lock()
            .init(HEAP_SPACE.as_ptr() as usize, KERNEL_HEAP_SIZE);
    }
}
```



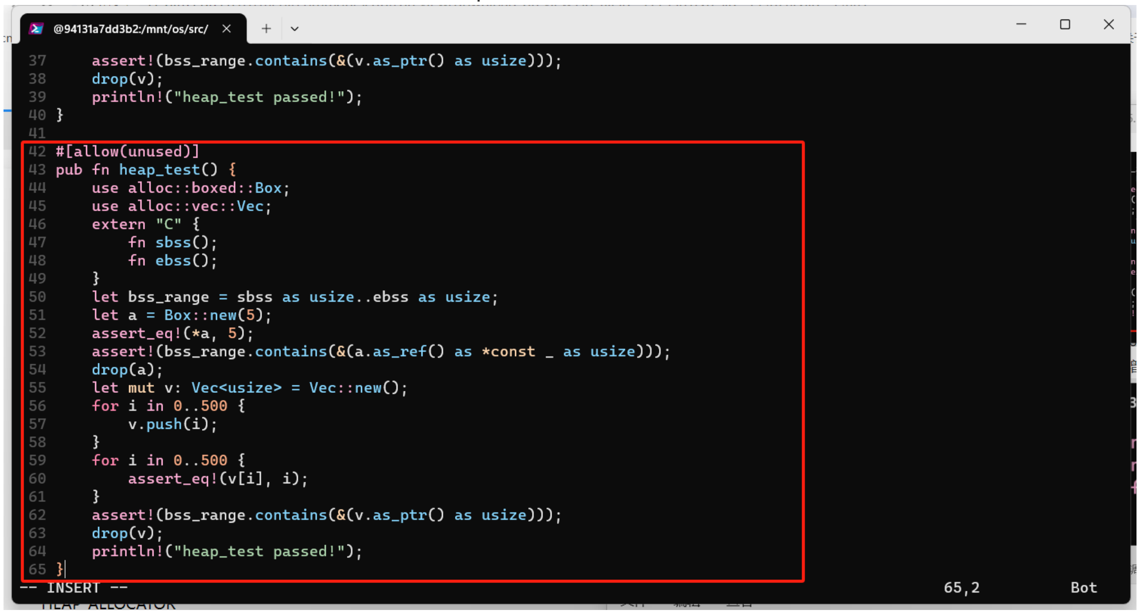
```
            .lock()
            .init(HEAP_SPACE.as_ptr() as usize, KERNEL_HEAP_SIZE);
    }
}

#[allow(unused)]
pub fn heap_test() {
    use alloc::boxed::Box;
    use alloc::vec::Vec;
    extern "C" {
        fn sbss();
        fn ebss();
    }
    let bss_range = sbss as usize..ebss as usize;
    let a = Box::new(5);
    assert_eq!(*a, 5);
    assert!(bss_range.contains(&(a.as_ref() as *const _ as usize)));
    drop(a);
    let mut v: Vec<usize> = Vec::new();
    for i in 0..500 {
        v.push(i);
    }
    for i in 0..500 {
        assert_eq!(v[i], i);
    }
    assert!(bss_range.contains(&(v.as_ptr() as usize)));
    drop(v);
    println!("heap_test passed!");
}
-- INSERT --                                                  40,2          Bot
```

- 在main.rs中增加处理动态内存分配失败的情况（os/src/main.rs）



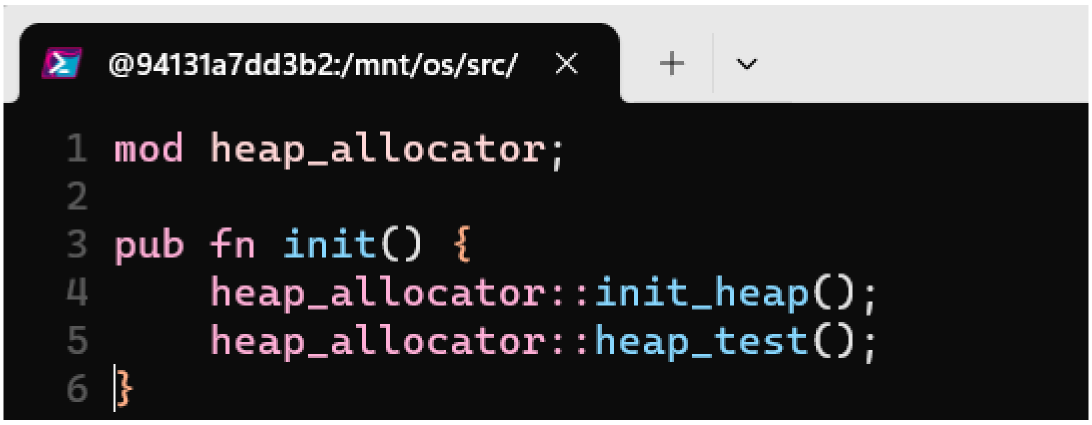
```
#![no_std]
#![no_main]
#![feature(panic_info_message)]
#![feature(alloc_error_handler)]
```

- 实现测试动态内存分配（os/src/mm/heap_allocator.rs）



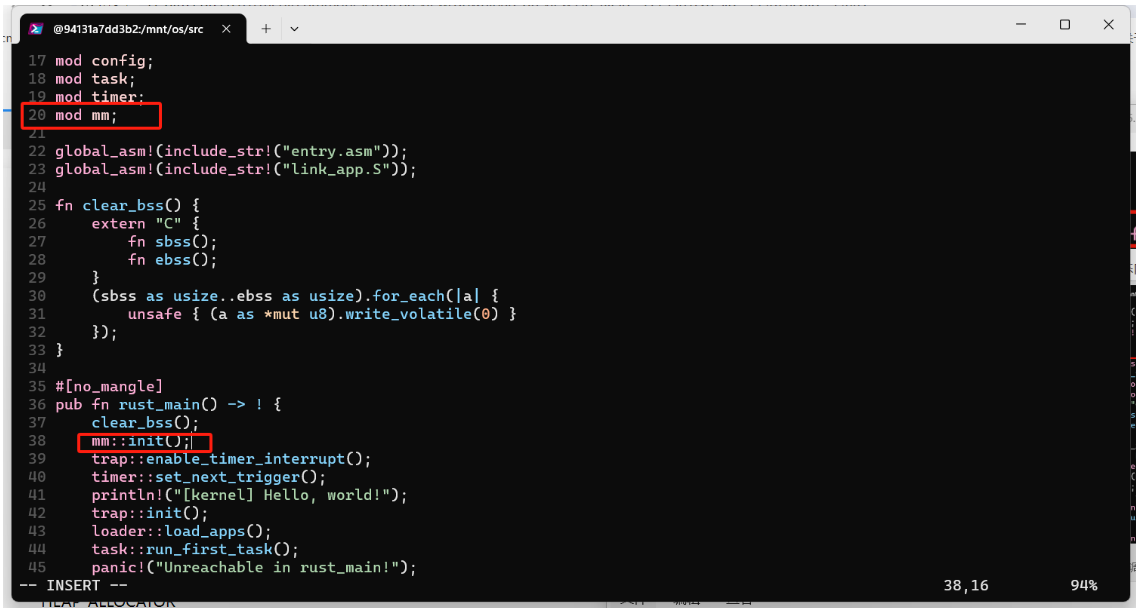
```
    assert!(bss_range.contains(&(v.as_ptr() as usize)));
    drop(v);
    println!("heap_test passed!");
}

#[allow(unused)]
pub fn heap_test() {
    use alloc::boxed::Box;
    use alloc::vec::Vec;
    extern "C" {
        fn sbss();
        fn ebss();
    }
    let bss_range = sbss as usize..ebss as usize;
    let a = Box::new(5);
    assert_eq!(*a, 5);
    assert!(bss_range.contains(&(a.as_ref() as *const _ as usize)));
    drop(a);
    let mut v: Vec<usize> = Vec::new();
    for i in 0..500 {
        v.push(i);
    }
    for i in 0..500 {
        assert_eq!(v[i], i);
    }
    assert!(bss_range.contains(&(v.as_ptr() as usize)));
    drop(v);
    println!("heap_test passed!");
}
-- INSERT --                                                  65,2          Bot
```

- 封装heap_allocator（os/src/mm/mod.rs）



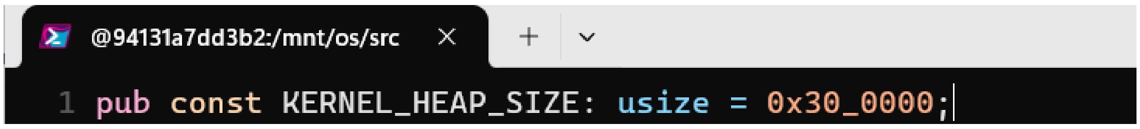
```
mod heap_allocator;

pub fn init() {
    heap_allocator::init_heap();
    heap_allocator::heap_test();
}
```

- 在main.rs中对mm进行初始化



```
mod config;
mod task;
mod timer;
mod mm;

global_asm!(include_str!("entry.asm"));
global_asm!(include_str!("link_app.S"));

fn clear_bss() {
    extern "C" {
        fn sbss();
        fn ebss();
    }
    (sbss as usize..ebss as usize).for_each(|a| {
        unsafe { (a as *mut u8).write_volatile(0) }
    });
}

#[no_mangle]
pub fn rust_main() -> ! {
    clear_bss();
    mm::init();
    trap::enable_timer_interrupt();
    timer::set_next_trigger();
    println!("[kernel] Hello, world!");
    trap::init();
    loader::load_apps();
    task::run_first_task();
    panic!("Unreachable in rust_main!");
-- INSERT --                                                38,16         94%
```

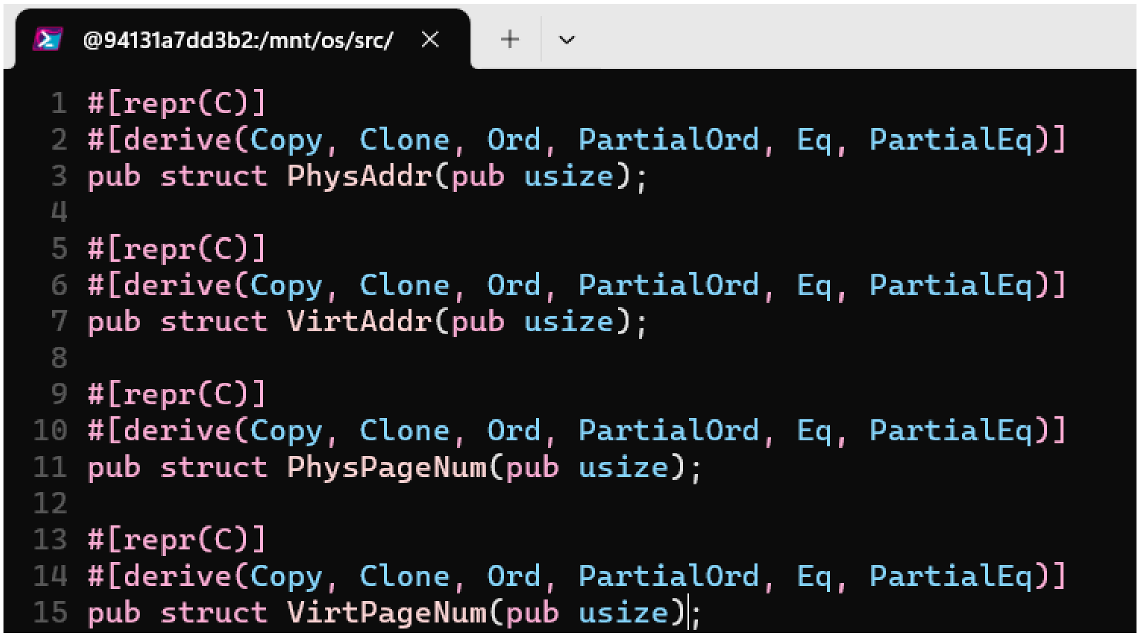
- 修改config.rs，添加：



```
pub const KERNEL_HEAP_SIZE: usize = 0x30_0000;
```

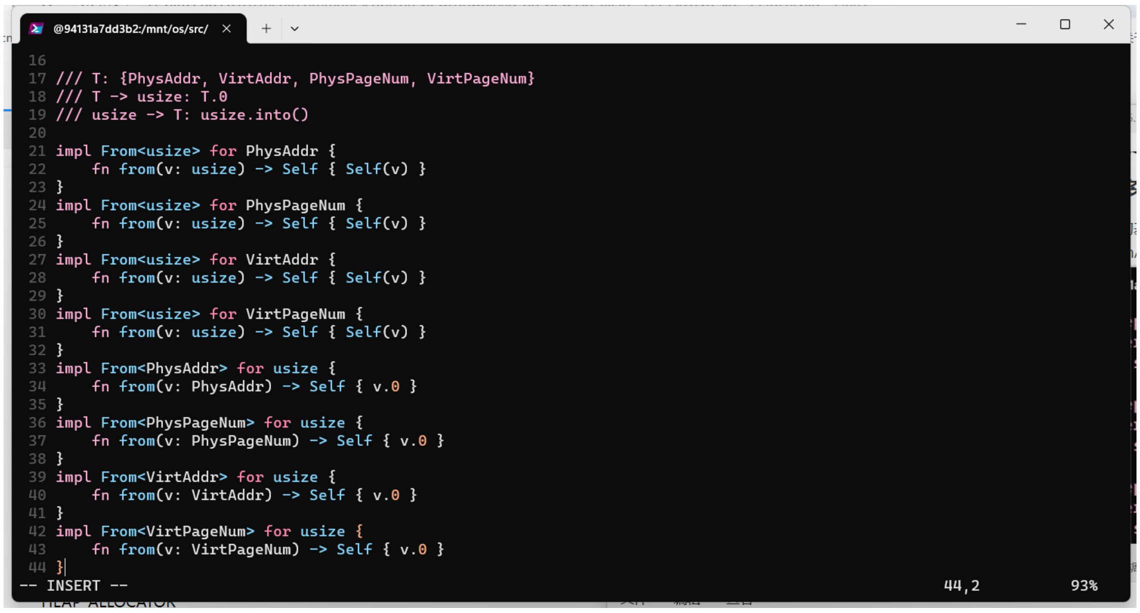
1.2 实现虚拟地址与物理地址的基本定义

- 定义所需要的基本数据结构，包括物理地址、虚拟地址、物理页号、虚拟页号（os/src/mm/address.rs）



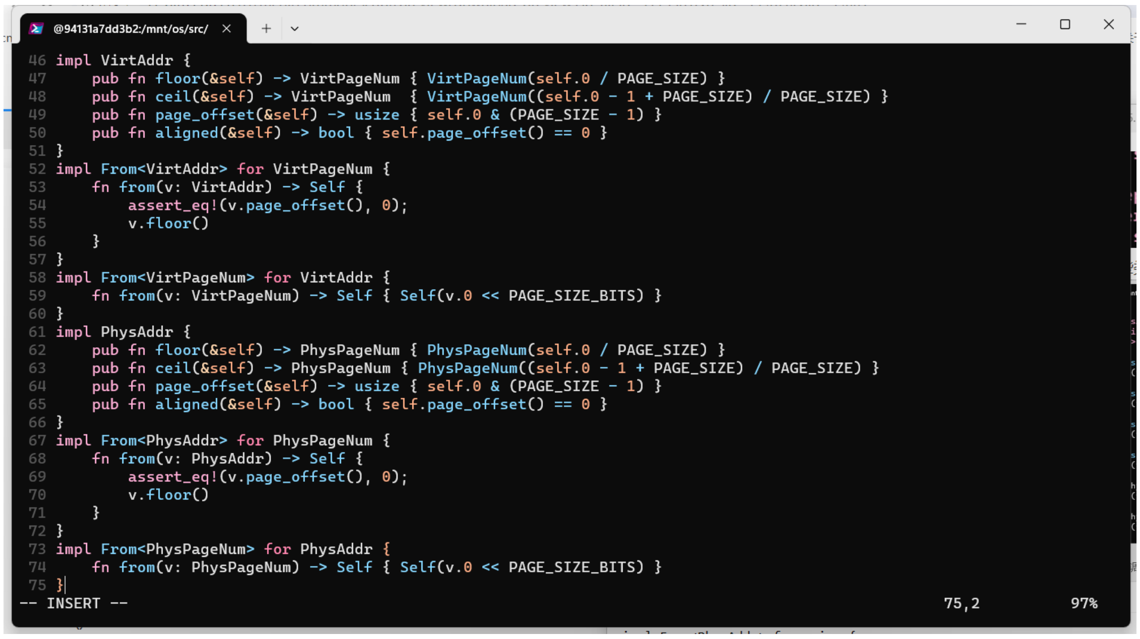
```
#[repr(C)]
#[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
pub struct PhysAddr(pub usize);

#[repr(C)]
#[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
pub struct VirtAddr(pub usize);

#[repr(C)]
#[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
pub struct PhysPageNum(pub usize);

#[repr(C)]
#[derive(Copy, Clone, Ord, PartialOrd, Eq, PartialEq)]
pub struct VirtPageNum(pub usize);
```

- 实现上述这些类型和usize之间的相互转换。



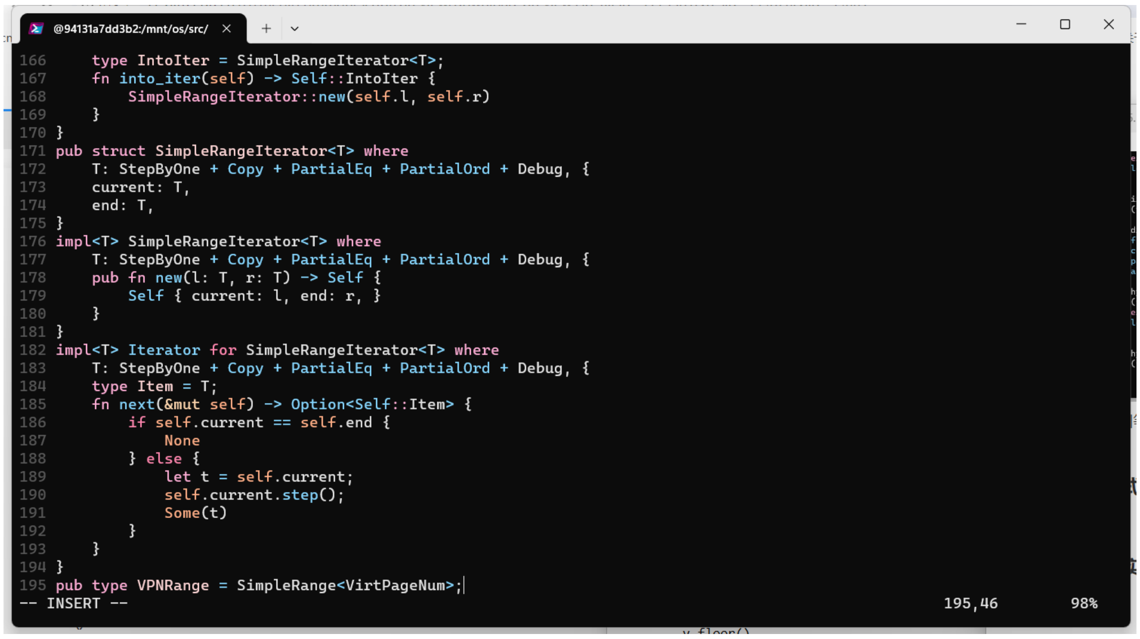
```

/// T: {PhysAddr, VirtAddr, PhysPageNum, VirtPageNum}
/// T -> usize: T.0
/// usize -> T: usize.into()

impl From<usize> for PhysAddr {
    fn from(v: usize) -> Self { Self(v) }
}
impl From<usize> for PhysPageNum {
    fn from(v: usize) -> Self { Self(v) }
}
impl From<usize> for VirtAddr {
    fn from(v: usize) -> Self { Self(v) }
}
impl From<usize> for VirtPageNum {
    fn from(v: usize) -> Self { Self(v) }
}
impl From<PhysAddr> for usize {
    fn from(v: PhysAddr) -> Self { v.0 }
}
impl From<PhysPageNum> for usize {
    fn from(v: PhysPageNum) -> Self { v.0 }
}
impl From<VirtAddr> for usize {
    fn from(v: VirtAddr) -> Self { v.0 }
}
impl From<VirtPageNum> for usize {
    fn from(v: VirtPageNum) -> Self { v.0 }
}
-- INSERT --                                                    44,2          93%
```

- 实现地址和页号之间的相互转换。



```
impl VirtAddr {
    pub fn floor(&self) -> VirtPageNum { VirtPageNum(self.0 / PAGE_SIZE) }
    pub fn ceil(&self) -> VirtPageNum  { VirtPageNum((self.0 - 1 + PAGE_SIZE) / PAGE_SIZE) }
    pub fn page_offset(&self) -> usize { self.0 & (PAGE_SIZE - 1) }
    pub fn aligned(&self) -> bool { self.page_offset() == 0 }
}
impl From<VirtAddr> for VirtPageNum {
    fn from(v: VirtAddr) -> Self {
        assert_eq!(v.page_offset(), 0);
        v.floor()
    }
}
impl From<VirtPageNum> for VirtAddr {
    fn from(v: VirtPageNum) -> Self { Self(v.0 << PAGE_SIZE_BITS) }
}
impl PhysAddr {
    pub fn floor(&self) -> PhysPageNum { PhysPageNum(self.0 / PAGE_SIZE) }
    pub fn ceil(&self) -> PhysPageNum { PhysPageNum((self.0 - 1 + PAGE_SIZE) / PAGE_SIZE) }
    pub fn page_offset(&self) -> usize { self.0 & (PAGE_SIZE - 1) }
    pub fn aligned(&self) -> bool { self.page_offset() == 0 }
}
impl From<PhysAddr> for PhysPageNum {
    fn from(v: PhysAddr) -> Self {
        assert_eq!(v.page_offset(), 0);
        v.floor()
    }
}
impl From<PhysPageNum> for PhysAddr {
    fn from(v: PhysPageNum) -> Self { Self(v.0 << PAGE_SIZE_BITS) }
}
-- INSERT --                                                    75,2          97%
```

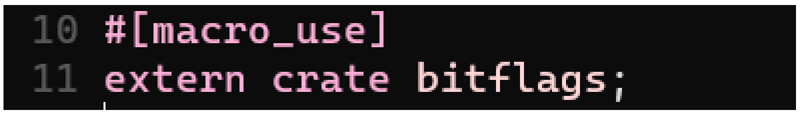
- 实现查询索引等其他内容。



```
    type IntoIter = SimpleRangeIterator<T>;
    fn into_iter(self) -> Self::IntoIter {
        SimpleRangeIterator::new(self.l, self.r)
    }
}
pub struct SimpleRangeIterator<T> where
    T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
    current: T,
    end: T,
}
impl<T> SimpleRangeIterator<T> where
    T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
    pub fn new(l: T, r: T) -> Self {
        Self { current: l, end: r, }
    }
}
impl<T> Iterator for SimpleRangeIterator<T> where
    T: StepByOne + Copy + PartialEq + PartialOrd + Debug, {
    type Item = T;
    fn next(&mut self) -> Option<Self::Item> {
        if self.current == self.end {
            None
        } else {
            let t = self.current;
            self.current.step();
            Some(t)
        }
    }
}
pub type VPNRange = SimpleRange<VirtPageNum>;
-- INSERT --                                                    195,46        98%
```

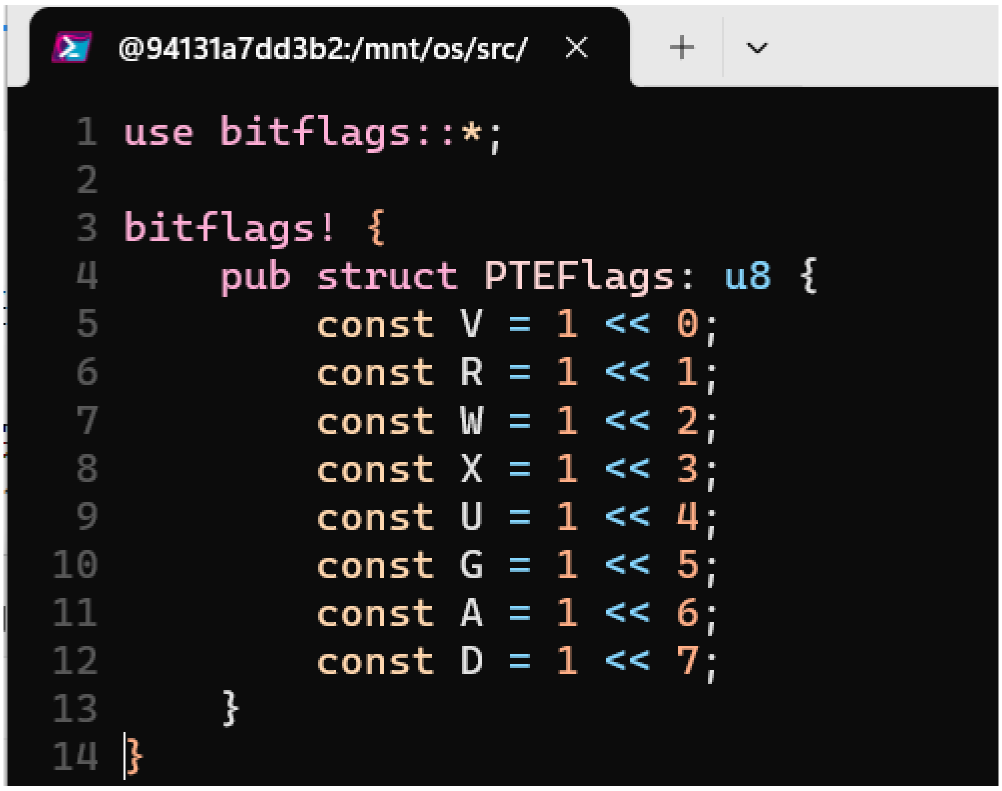
1.3 定义页表项数据结构

- 实现页表项中的标志位PTElages
  - os/src/main.rs



```rust
#[macro_use]
extern crate bitflags;
```

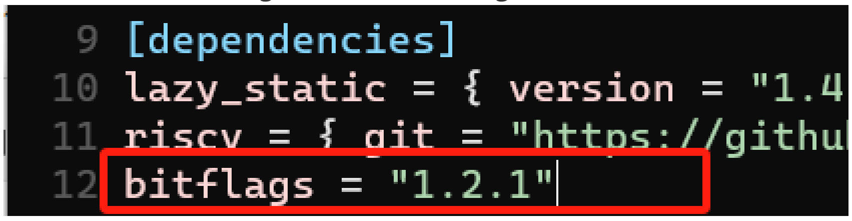
  - os/src/mm/page_table.rs



@94131a7dd3b2:/mnt/os/src/

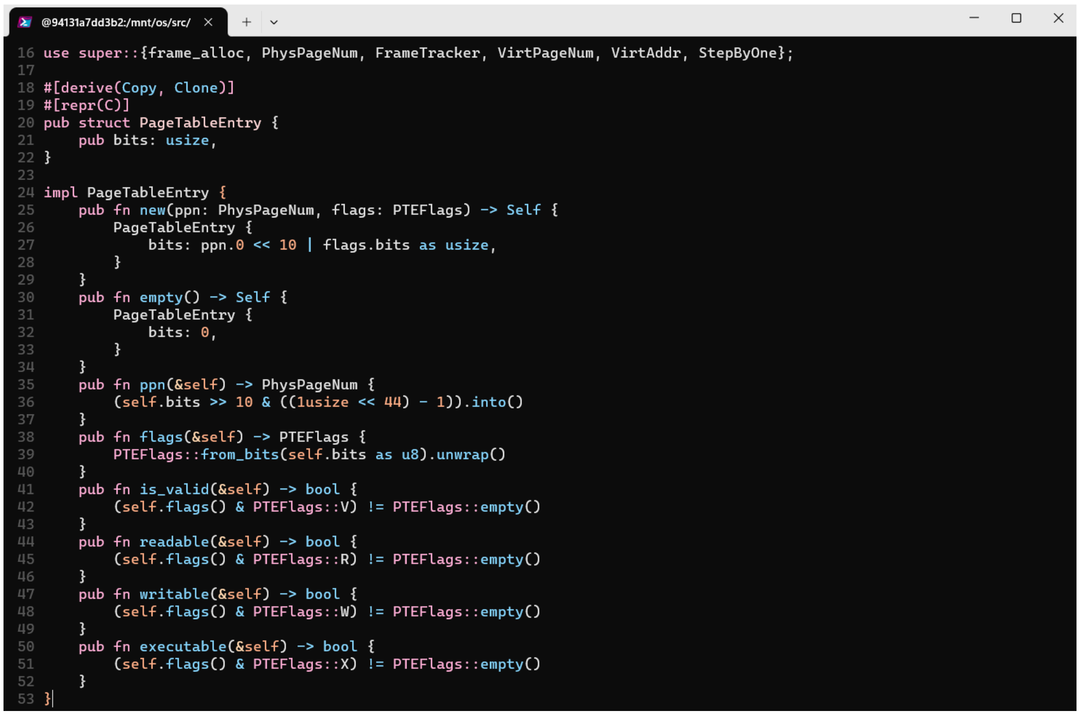
```rust
use bitflags::*;

bitflags! {
    pub struct PTEFlags: u8 {
        const V = 1 << 0;
        const R = 1 << 1;
        const W = 1 << 2;
        const X = 1 << 3;
        const U = 1 << 4;
        const G = 1 << 5;
        const A = 1 << 6;
        const D = 1 << 7;
    }
}
```

  - 在配置文件中增加bitflgs的依赖。（os/Cargo.toml）



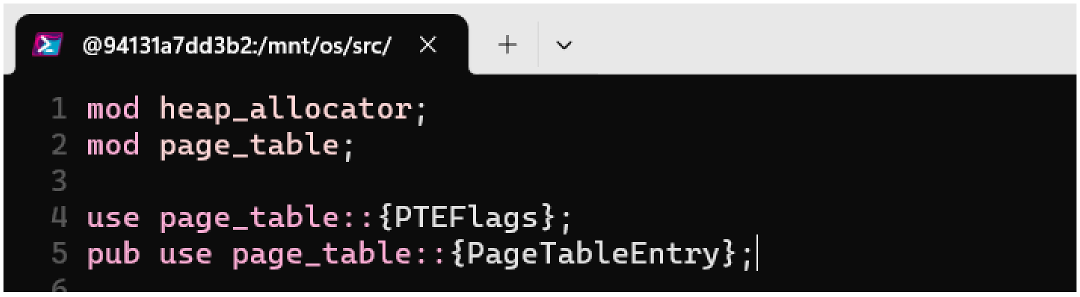
```toml
[dependencies]
lazy_static = { version = "1.4
riscv = { git = "https://githu
bitflags = "1.2.1"
```

- 实现PageTableEntry。（os/src/mm/page_table.rs）



```
use super::{frame_alloc, PhysPageNum, FrameTracker, VirtPageNum, VirtAddr, StepByOne};

#[derive(Copy, Clone)]
#[repr(C)]
pub struct PageTableEntry {
    pub bits: usize,
}

impl PageTableEntry {
    pub fn new(ppn: PhysPageNum, flags: PTEFlags) -> Self {
        PageTableEntry {
            bits: ppn.0 << 10 | flags.bits as usize,
        }
    }
    pub fn empty() -> Self {
        PageTableEntry {
            bits: 0,
        }
    }
    pub fn ppn(&self) -> PhysPageNum {
        (self.bits >> 10 & ((1usize << 44) - 1)).into()
    }
    pub fn flags(&self) -> PTEFlags {
        PTEFlags::from_bits(self.bits as u8).unwrap()
    }
    pub fn is_valid(&self) -> bool {
        (self.flags() & PTEFlags::V) != PTEFlags::empty()
    }
    pub fn readable(&self) -> bool {
        (self.flags() & PTEFlags::R) != PTEFlags::empty()
    }
    pub fn writable(&self) -> bool {
        (self.flags() & PTEFlags::W) != PTEFlags::empty()
    }
    pub fn executable(&self) -> bool {
        (self.flags() & PTEFlags::X) != PTEFlags::empty()
    }
}
```

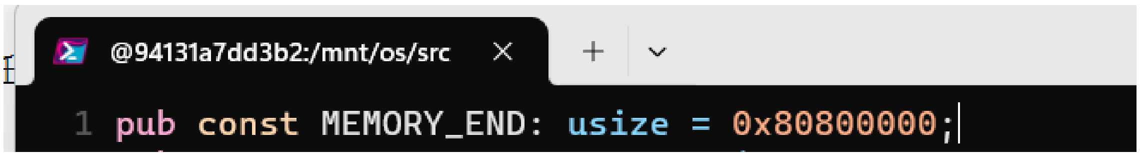
- 修改src/mm/mod/rs



```
mod heap_allocator;
mod page_table;

use page_table::{PTEFlags};
pub use page_table::{PageTableEntry};
```

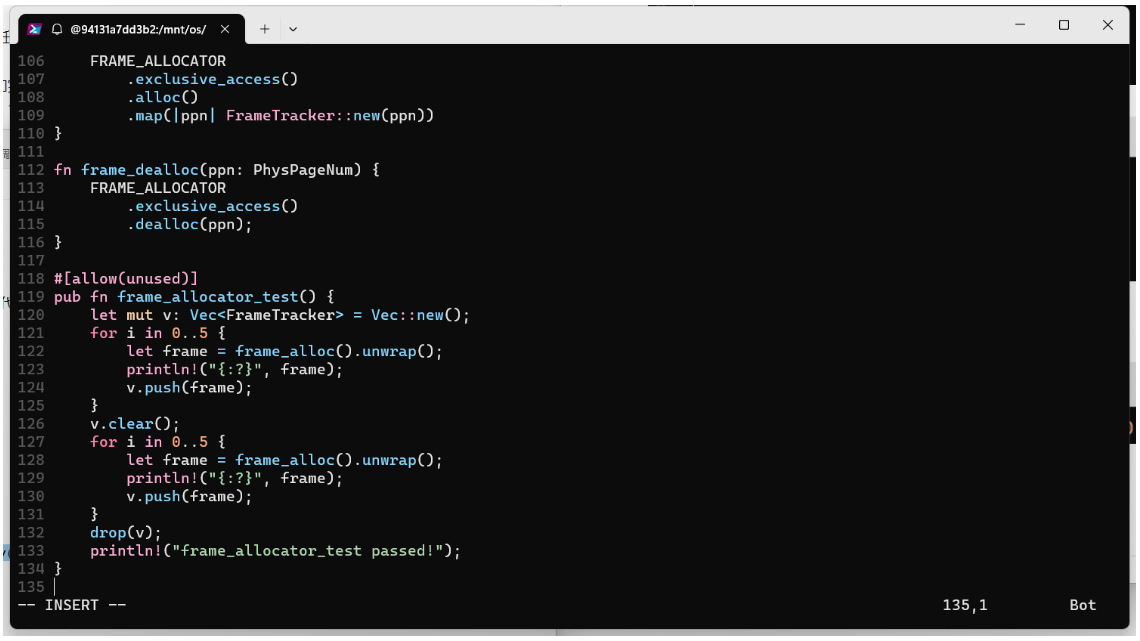
1.4 实现物理帧的管理和分配

- 设置物理内存的终止地址。（os/src/config.rs）



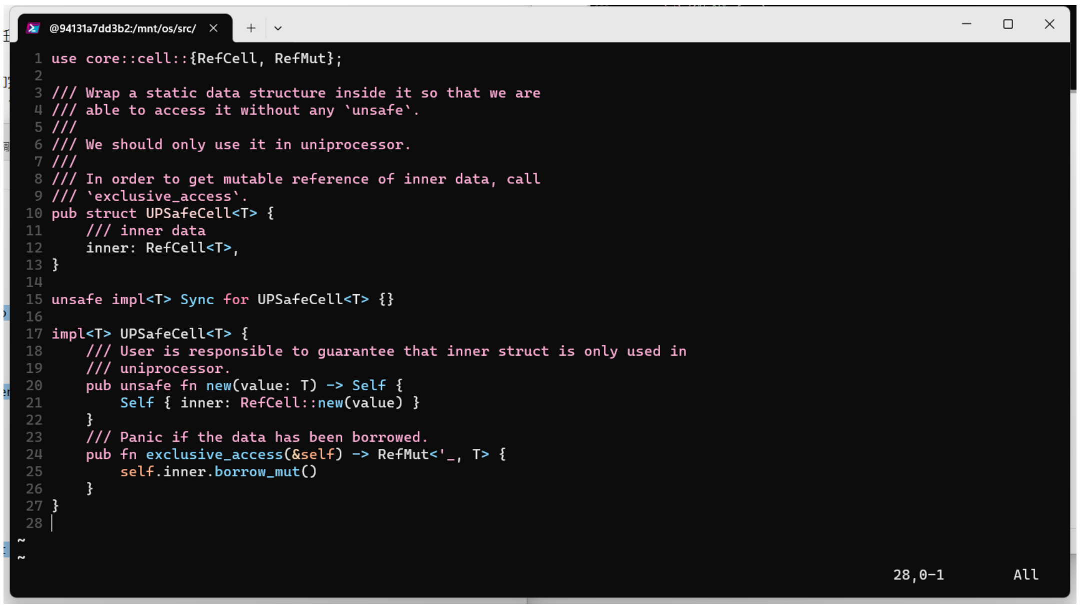
```
pub const MEMORY_END: usize = 0x80800000;
```

- 实现物理帧管理。（os/src/mm/frame_allocator.rs）



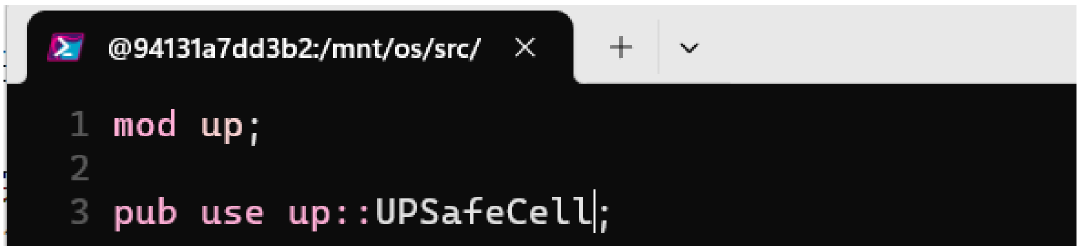
```
    FRAME_ALLOCATOR
        .exclusive_access()
        .alloc()
        .map(|ppn| FrameTracker::new(ppn))
}

fn frame_dealloc(ppn: PhysPageNum) {
    FRAME_ALLOCATOR
        .exclusive_access()
        .dealloc(ppn);
}

#[allow(unused)]
pub fn frame_allocator_test() {
    let mut v: Vec<FrameTracker> = Vec::new();
    for i in 0..5 {
        let frame = frame_alloc().unwrap();
        println!("{:?}", frame);
        v.push(frame);
    }
    v.clear();
    for i in 0..5 {
        let frame = frame_alloc().unwrap();
        println!("{:?}", frame);
        v.push(frame);
    }
    drop(v);
    println!("frame_allocator_test passed!");
}

-- INSERT --                                                    135,1         Bot
```

- 增加sync模块
  - os/src/sync/up.rs



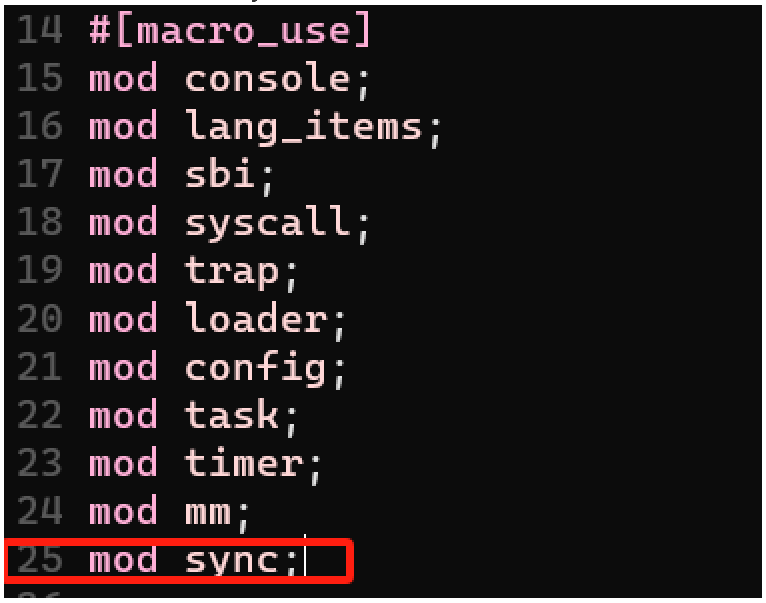
```rust
use core::cell::{RefCell, RefMut};

/// Wrap a static data structure inside it so that we are
/// able to access it without any `unsafe`.
///
/// We should only use it in uniprocessor.
///
/// In order to get mutable reference of inner data, call
/// `exclusive_access`.
pub struct UPSafeCell<T> {
    /// inner data
    inner: RefCell<T>,
}

unsafe impl<T> Sync for UPSafeCell<T> {}

impl<T> UPSafeCell<T> {
    /// User is responsible to guarantee that inner struct is only used in
    /// uniprocessor.
    pub unsafe fn new(value: T) -> Self {
        Self { inner: RefCell::new(value) }
    }
    /// Panic if the data has been borrowed.
    pub fn exclusive_access(&self) -> RefMut<'_, T> {
        self.inner.borrow_mut()
    }
}

~
~
                                                          28,0-1        All
```

  - os/src/sync/mod.rs



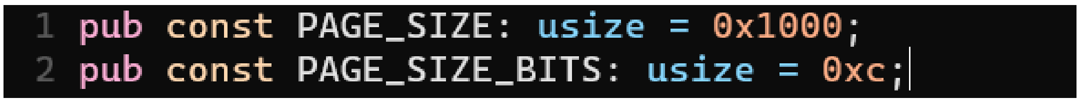
```rust
mod up;

pub use up::UPSafeCell;
```

  - /os/src/main.rs增加sync模块。



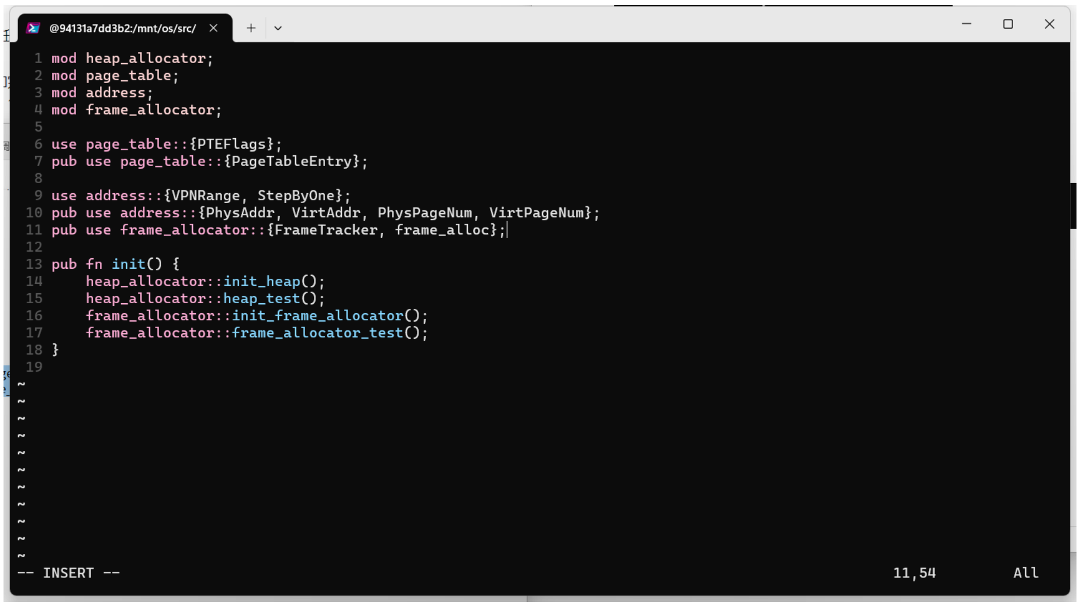
```rust
#[macro_use]
mod console;
mod lang_items;
mod sbi;
mod syscall;
mod trap;
mod loader;
mod config;
mod task;
mod timer;
mod mm;
mod sync;
```

- 物理帧管理测试
  - os/src/config.rs



```rust
pub const PAGE_SIZE: usize = 0x1000;
pub const PAGE_SIZE_BITS: usize = 0xc;
```

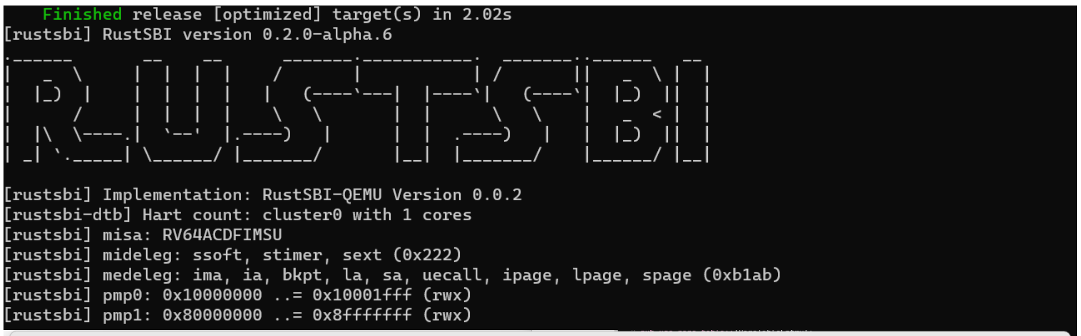
- os/src/mm/mod.rs



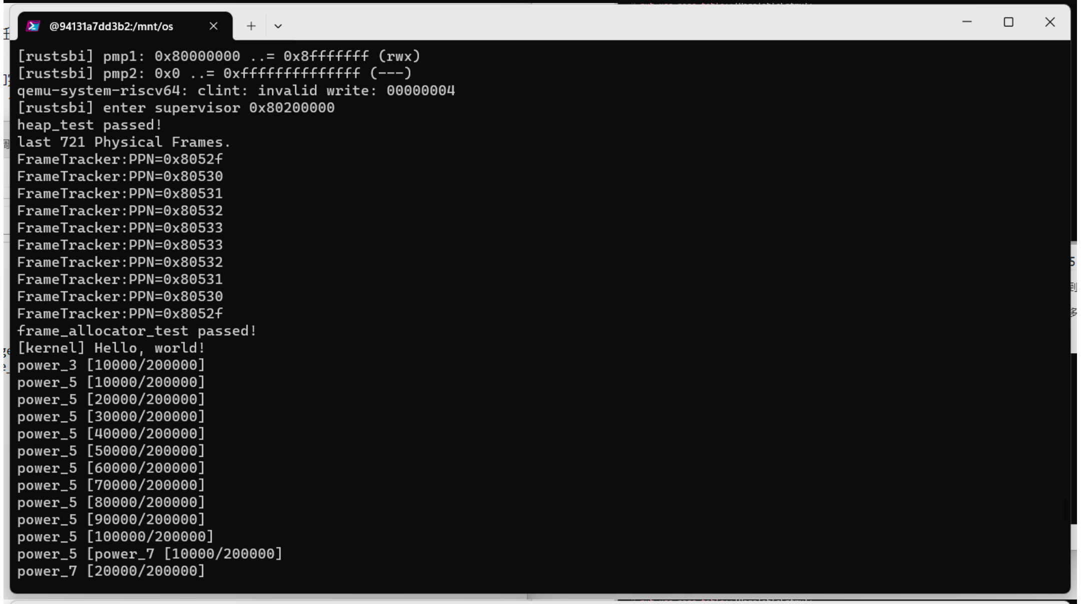
@94131a7dd3b2:/mnt/os/src/

```rust
mod heap_allocator;
mod page_table;
mod address;
mod frame_allocator;

use page_table::{PTEFlags};
pub use page_table::{PageTableEntry};

use address::{VPNRange, StepByOne};
pub use address::{PhysAddr, VirtAddr, PhysPageNum, VirtPageNum};
pub use frame_allocator::{FrameTracker, frame_alloc};

pub fn init() {
    heap_allocator::init_heap();
    heap_allocator::heap_test();
    frame_allocator::init_frame_allocator();
    frame_allocator::frame_allocator_test();
}

~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    11,54         All
```

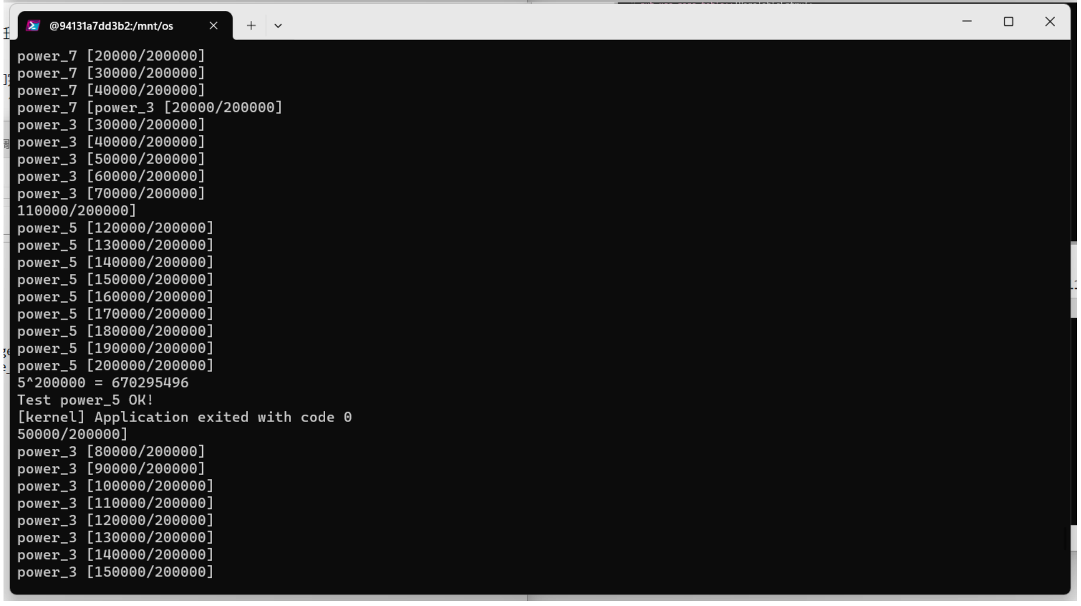
- 测试结果



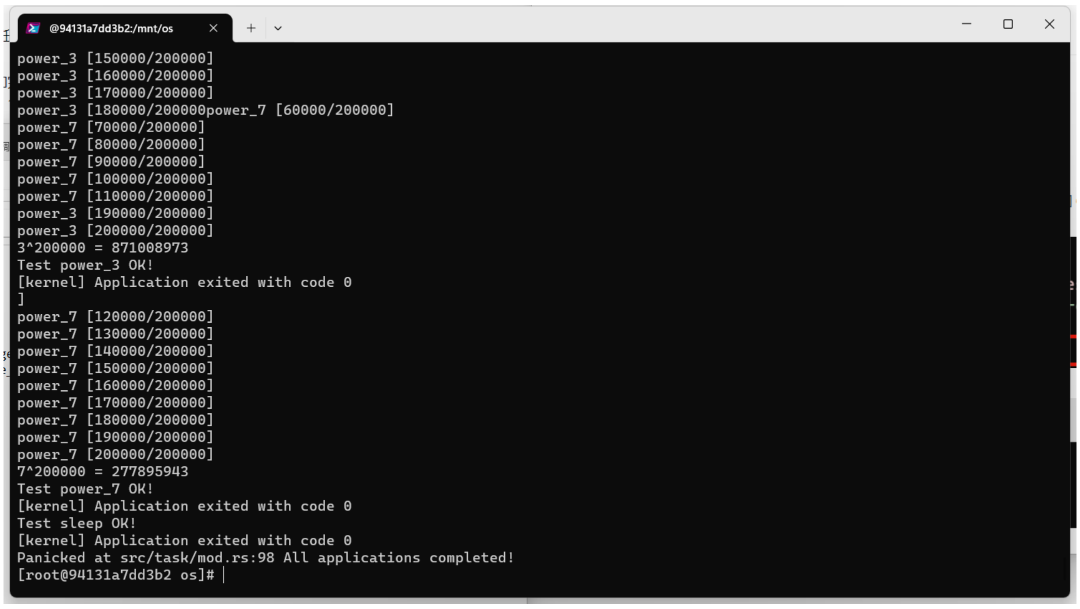
```
    Finished release [optimized] target(s) in 2.02s
[rustsbi] RustSBI version 0.2.0-alpha.6
.______       __    __      _______.___________.  _______..______   __
|   _  \     |  |  |  |    /       |           | /       ||   _  \ |  |
|  |_)  |    |  |  |  |   |   (----`---|  |----`|   (----`|  |_)  ||  |
|      /     |  |  |  |    \   \       |  |      \   \    |   _  < |  |
|  |\  \----.|  `--'  |.----)   |      |  |  .----)   |   |  |_)  ||  |
| _| `._____| \______/ |_______/       |__|  |_______/    |______/ |__|

[rustsbi] Implementation: RustSBI-QEMU Version 0.0.2
[rustsbi-dtb] Hart count: cluster0 with 1 cores
[rustsbi] misa: RV64ACDFIMSU
[rustsbi] mideleg: ssoft, stimer, sext (0x222)
[rustsbi] medeleg: ima, ia, bkpt, la, sa, uecall, ipage, lpage, spage (0xb1ab)
[rustsbi] pmp0: 0x10000000 ..= 0x10001fff (rwx)
[rustsbi] pmp1: 0x80000000 ..= 0x8fffffff (rwx)
```



```
[rustsbi] pmp1: 0x80000000 ..= 0x8fffffff (rwx)
[rustsbi] pmp2: 0x0 ..= 0xffffffffffffff (---)
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
heap_test passed!
last 721 Physical Frames.
FrameTracker:PPN=0x8052f
FrameTracker:PPN=0x80530
FrameTracker:PPN=0x80531
FrameTracker:PPN=0x80532
FrameTracker:PPN=0x80533
FrameTracker:PPN=0x80533
FrameTracker:PPN=0x80532
FrameTracker:PPN=0x80531
FrameTracker:PPN=0x80530
FrameTracker:PPN=0x8052f
frame_allocator_test passed!
[kernel] Hello, world!
power_3 [10000/200000]
power_5 [10000/200000]
power_5 [20000/200000]
power_5 [30000/200000]
power_5 [40000/200000]
power_5 [50000/200000]
power_5 [60000/200000]
power_5 [70000/200000]
power_5 [80000/200000]
power_5 [90000/200000]
power_5 [100000/200000]
power_5 [power_7 [10000/200000]
power_7 [20000/200000]
```



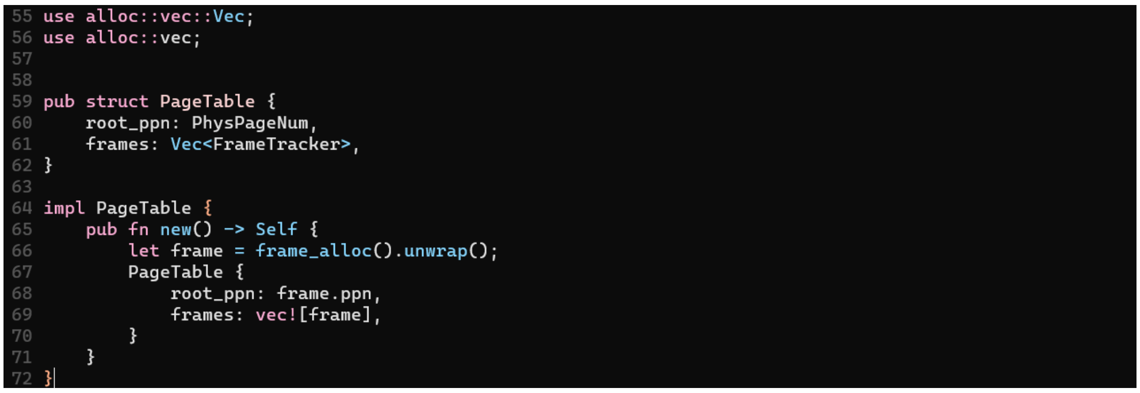
```
power_7 [20000/200000]
power_7 [30000/200000]
power_7 [40000/200000]
power_7 [power_3 [20000/200000]
power_3 [30000/200000]
power_3 [40000/200000]
power_3 [50000/200000]
power_3 [60000/200000]
power_3 [70000/200000]
110000/200000]
power_5 [120000/200000]
power_5 [130000/200000]
power_5 [140000/200000]
power_5 [150000/200000]
power_5 [160000/200000]
power_5 [170000/200000]
power_5 [180000/200000]
power_5 [190000/200000]
power_5 [200000/200000]
5^200000 = 670295496
Test power_5 OK!
[kernel] Application exited with code 0
50000/200000]
power_3 [80000/200000]
power_3 [90000/200000]
power_3 [100000/200000]
power_3 [110000/200000]
power_3 [120000/200000]
power_3 [130000/200000]
power_3 [140000/200000]
power_3 [150000/200000]
```



```
power_3 [150000/200000]
power_3 [160000/200000]
power_3 [170000/200000]
power_3 [180000/200000power_7 [60000/200000]
power_7 [70000/200000]
power_7 [80000/200000]
power_7 [90000/200000]
power_7 [100000/200000]
power_7 [110000/200000]
power_3 [190000/200000]
power_3 [200000/200000]
3^200000 = 871008973
Test power_3 OK!
[kernel] Application exited with code 0
]
power_7 [120000/200000]
power_7 [130000/200000]
power_7 [140000/200000]
power_7 [150000/200000]
power_7 [160000/200000]
power_7 [170000/200000]
power_7 [180000/200000]
power_7 [190000/200000]
power_7 [200000/200000]
7^200000 = 277895943
Test power_7 OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
Panicked at src/task/mod.rs:98 All applications completed!
[root@94131a7dd3b2 os]#
```

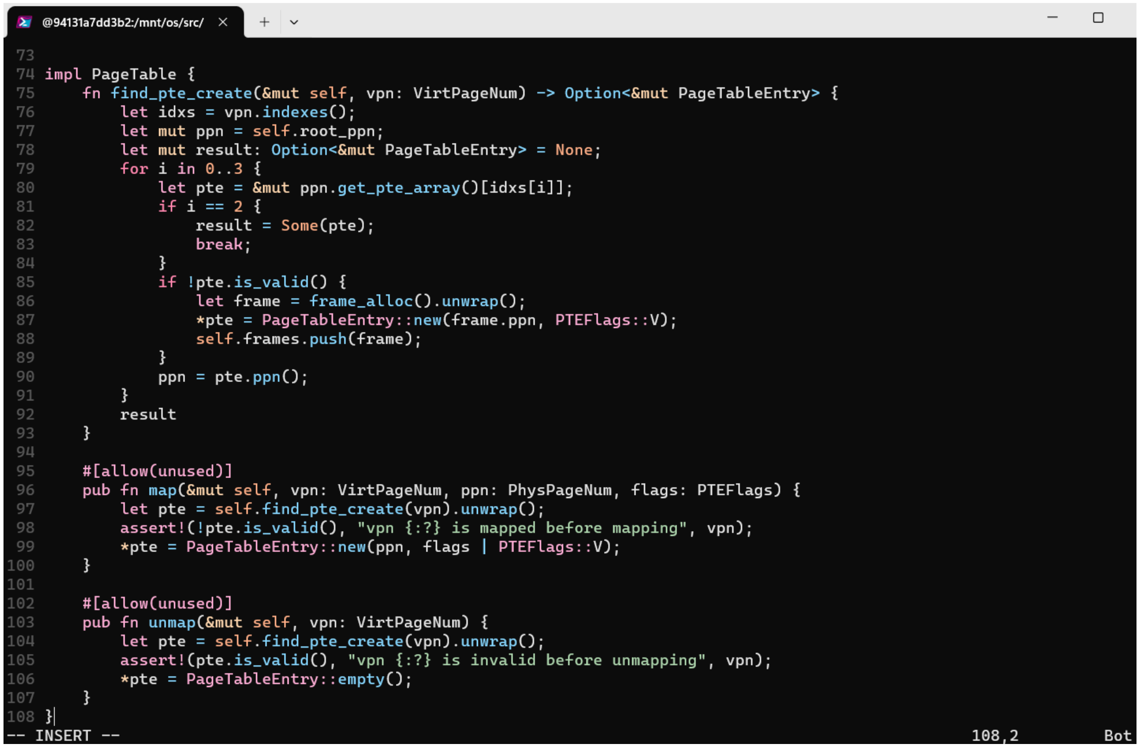
1.5 多级页表管理

- 实现页表的基本数据结构。（os/src/mm/page_table.rs）



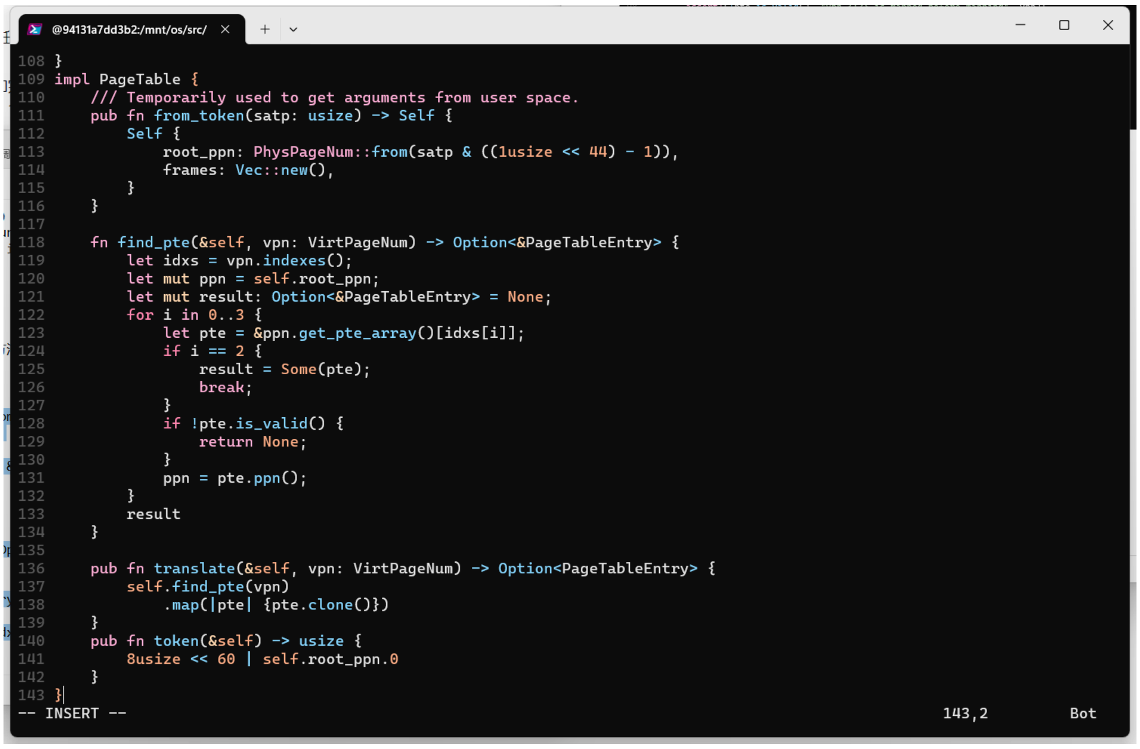
```rust
use alloc::vec::Vec;
use alloc::vec;


pub struct PageTable {
    root_ppn: PhysPageNum,
    frames: Vec<FrameTracker>,
}

impl PageTable {
    pub fn new() -> Self {
        let frame = frame_alloc().unwrap();
        PageTable {
            root_ppn: frame.ppn,
            frames: vec![frame],
        }
    }
}
```

- 实现建立和拆除虚实地址之间的映射关系。（os/src/mm/page_table.rs）



```rust

impl PageTable {
    fn find_pte_create(&mut self, vpn: VirtPageNum) -> Option<&mut PageTableEntry> {
        let idxs = vpn.indexes();
        let mut ppn = self.root_ppn;
        let mut result: Option<&mut PageTableEntry> = None;
        for i in 0..3 {
            let pte = &mut ppn.get_pte_array()[idxs[i]];
            if i == 2 {
                result = Some(pte);
                break;
            }
            if !pte.is_valid() {
                let frame = frame_alloc().unwrap();
                *pte = PageTableEntry::new(frame.ppn, PTEFlags::V);
                self.frames.push(frame);
            }
            ppn = pte.ppn();
        }
        result
    }

    #[allow(unused)]
    pub fn map(&mut self, vpn: VirtPageNum, ppn: PhysPageNum, flags: PTEFlags) {
        let pte = self.find_pte_create(vpn).unwrap();
        assert!(!pte.is_valid(), "vpn {:?} is mapped before mapping", vpn);
        *pte = PageTableEntry::new(ppn, flags | PTEFlags::V);
    }

    #[allow(unused)]
    pub fn unmap(&mut self, vpn: VirtPageNum) {
        let pte = self.find_pte_create(vpn).unwrap();
        assert!(pte.is_valid(), "vpn {:?} is invalid before unmapping", vpn);
        *pte = PageTableEntry::empty();
    }
}
-- INSERT --                                                    108,2          Bot
```

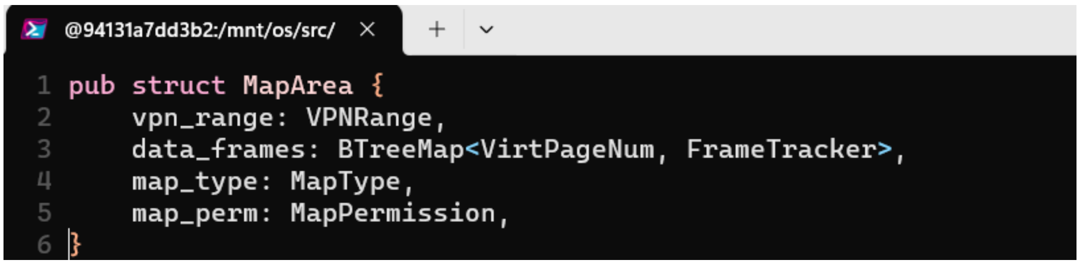
- 实现手动查询页表的方法。（os/src/mm/page_table.rs）



```
}
impl PageTable {
    /// Temporarily used to get arguments from user space.
    pub fn from_token(satp: usize) -> Self {
        Self {
            root_ppn: PhysPageNum::from(satp & ((1usize << 44) - 1)),
            frames: Vec::new(),
        }
    }

    fn find_pte(&self, vpn: VirtPageNum) -> Option<&PageTableEntry> {
        let idxs = vpn.indexes();
        let mut ppn = self.root_ppn;
        let mut result: Option<&PageTableEntry> = None;
        for i in 0..3 {
            let pte = &ppn.get_pte_array()[idxs[i]];
            if i == 2 {
                result = Some(pte);
                break;
            }
            if !pte.is_valid() {
                return None;
            }
            ppn = pte.ppn();
        }
        result
    }

    pub fn translate(&self, vpn: VirtPageNum) -> Option<PageTableEntry> {
        self.find_pte(vpn)
            .map(|pte| {pte.clone()})
    }
    pub fn token(&self) -> usize {
        8usize << 60 | self.root_ppn.0
    }
}
-- INSERT --                                                                    143,2          Bot
```

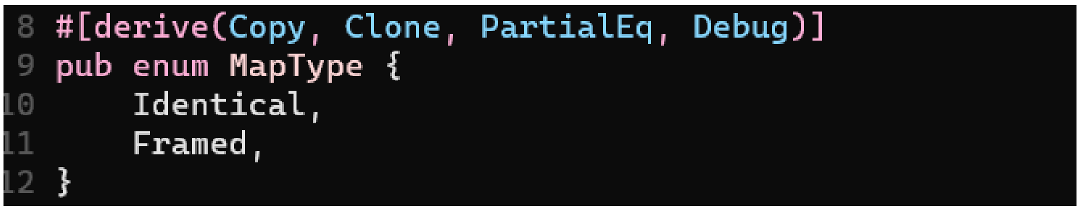
1.6 实现地址空间抽象

- 实现地址空间抽象
    - 首先，以逻辑段MapArea描述一段连续地址的虚拟内存。（os/src/mm/memory_set.rs）



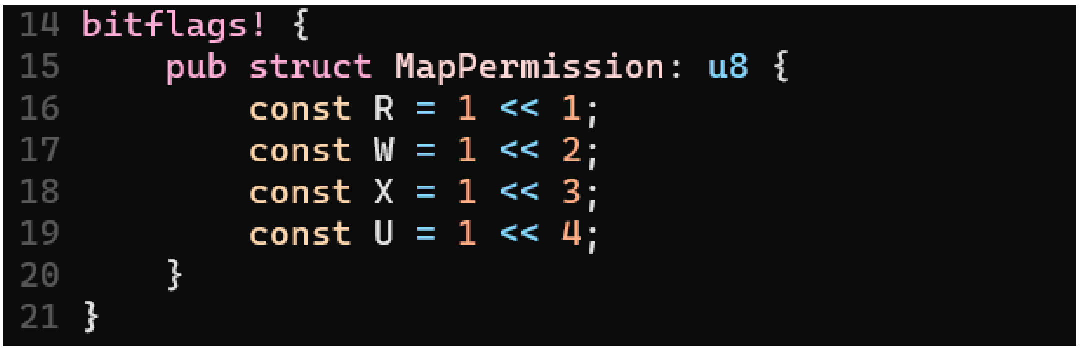
```
pub struct MapArea {
    vpn_range: VPNRange,
    data_frames: BTreeMap<VirtPageNum, FrameTracker>,
    map_type: MapType,
    map_perm: MapPermission,
}
```

    - 接着，用MapType描述逻辑段内所有虚拟页号映射到物理页帧的方式。（os/src/mm/memory_set.rs）



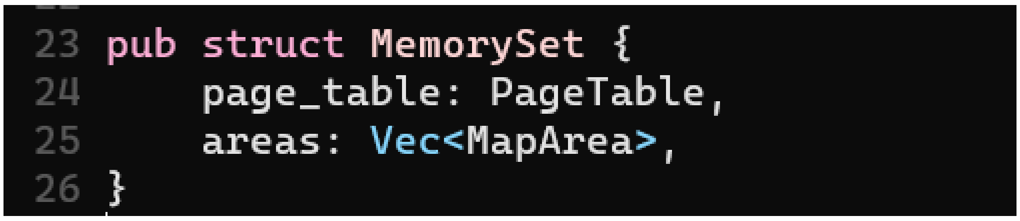
```
#[derive(Copy, Clone, PartialEq, Debug)]
pub enum MapType {
    Identical,
    Framed,
}
```

    - 利用MapPermisssion控制逻辑段的访问方式，其是页表项标志位PTEFlags的子集（os/src/mm/memory_set.rs）



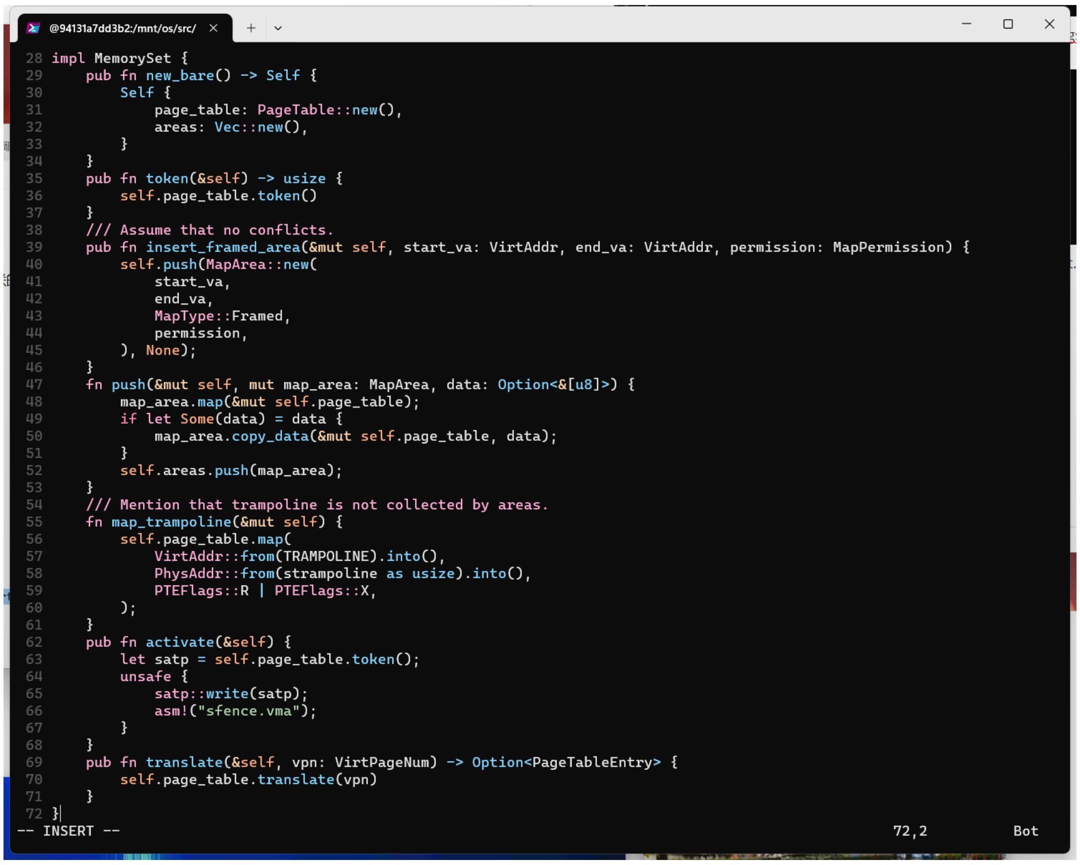
```
bitflags! {
    pub struct MapPermission: u8 {
        const R = 1 << 1;
        const W = 1 << 2;
        const X = 1 << 3;
        const U = 1 << 4;
    }
}
```

- 实现地址空间，也就是一系列有关联的逻辑段（os/src/mm/memory_set.rs）



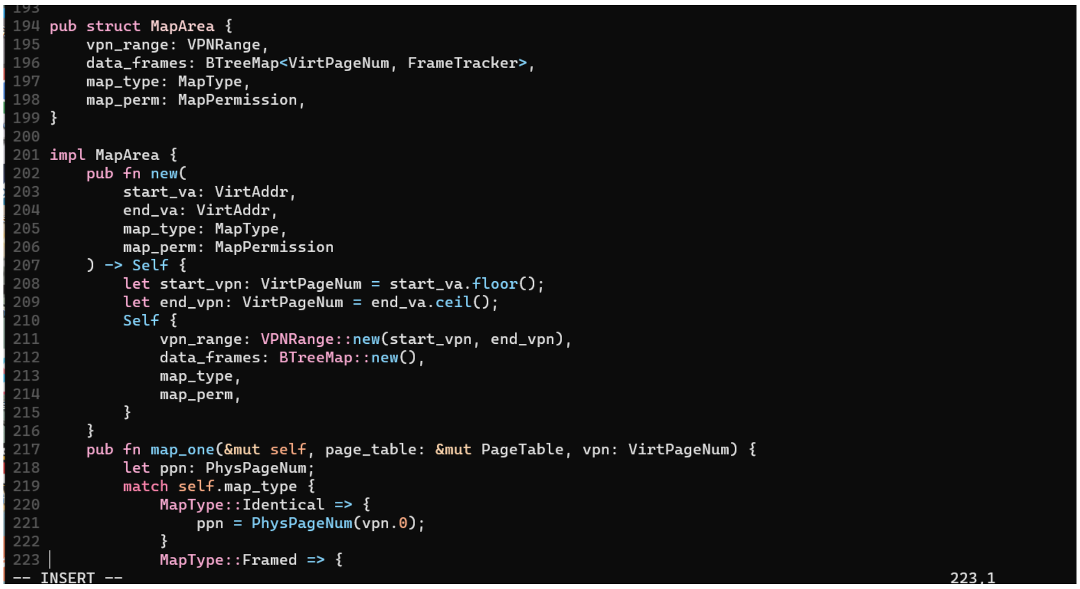
```rust
pub struct MemorySet {
        page_table: PageTable,
        areas: Vec<MapArea>,
}
```

- 实现MemorySet的方法（os/src/mm/memory_set.rs）



```rust
impl MemorySet {
    pub fn new_bare() -> Self {
        Self {
            page_table: PageTable::new(),
            areas: Vec::new(),
        }
    }
    pub fn token(&self) -> usize {
        self.page_table.token()
    }
    /// Assume that no conflicts.
    pub fn insert_framed_area(&mut self, start_va: VirtAddr, end_va: VirtAddr, permission: MapPermission) {
        self.push(MapArea::new(
            start_va,
            end_va,
            MapType::Framed,
            permission,
        ), None);
    }
    fn push(&mut self, mut map_area: MapArea, data: Option<&[u8]>) {
        map_area.map(&mut self.page_table);
        if let Some(data) = data {
            map_area.copy_data(&mut self.page_table, data);
        }
        self.areas.push(map_area);
    }
    /// Mention that trampoline is not collected by areas.
    fn map_trampoline(&mut self) {
        self.page_table.map(
            VirtAddr::from(TRAMPOLINE).into(),
            PhysAddr::from(strampoline as usize).into(),
            PTEFlags::R | PTEFlags::X,
        );
    }
    pub fn activate(&self) {
        let satp = self.page_table.token();
        unsafe {
            satp::write(satp);
            asm!("sfence.vma");
        }
    }
    pub fn translate(&self, vpn: VirtPageNum) -> Option<PageTableEntry> {
        self.page_table.translate(vpn)
    }
}
-- INSERT --                                                                    72,2          Bot
```

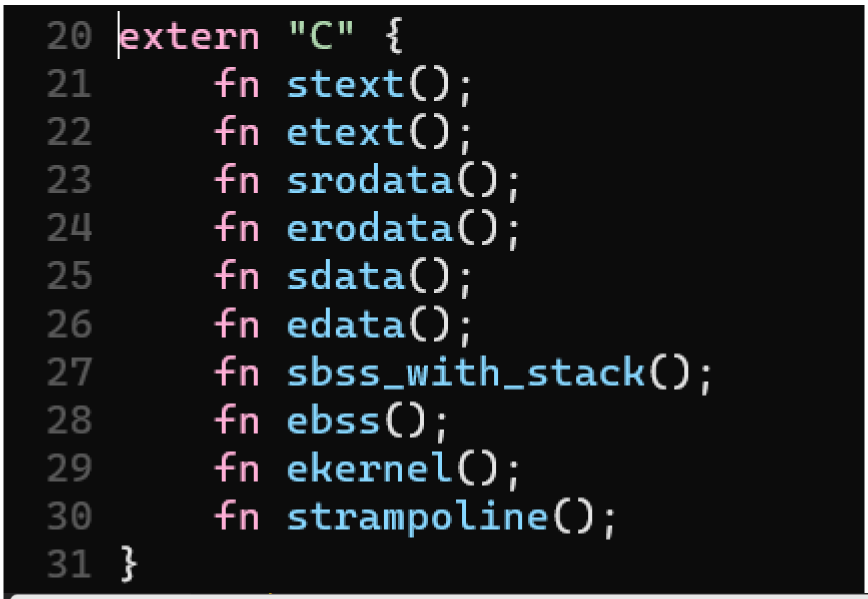
- 实现MapArea（os/src/mm/memory_set.rs）



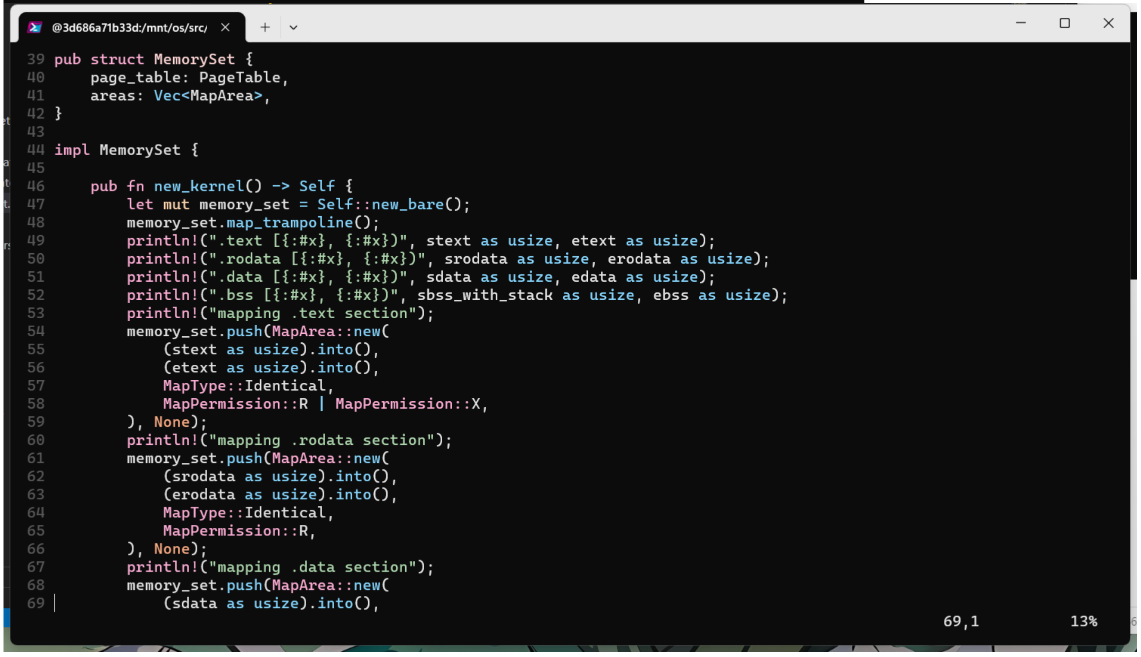
```rust
pub struct MapArea {
    vpn_range: VPNRange,
    data_frames: BTreeMap<VirtPageNum, FrameTracker>,
    map_type: MapType,
    map_perm: MapPermission,
}

impl MapArea {
    pub fn new(
        start_va: VirtAddr,
        end_va: VirtAddr,
        map_type: MapType,
        map_perm: MapPermission
    ) -> Self {
        let start_vpn: VirtPageNum = start_va.floor();
        let end_vpn: VirtPageNum = end_va.ceil();
        Self {
            vpn_range: VPNRange::new(start_vpn, end_vpn),
            data_frames: BTreeMap::new(),
            map_type,
            map_perm,
        }
    }
    pub fn map_one(&mut self, page_table: &mut PageTable, vpn: VirtPageNum) {
        let ppn: PhysPageNum;
        match self.map_type {
            MapType::Identical => {
                ppn = PhysPageNum(vpn.0);
            }
            MapType::Framed => {
-- INSERT --                                                                    223,1
```

- 实现创建内核地址空间的方法new_kernel。(os/src/mm/memory_set.rs)

此处容器id更换了是因为我重装了docker，之前的容器丢失了。所以重新导入镜像，创建了新的容器。

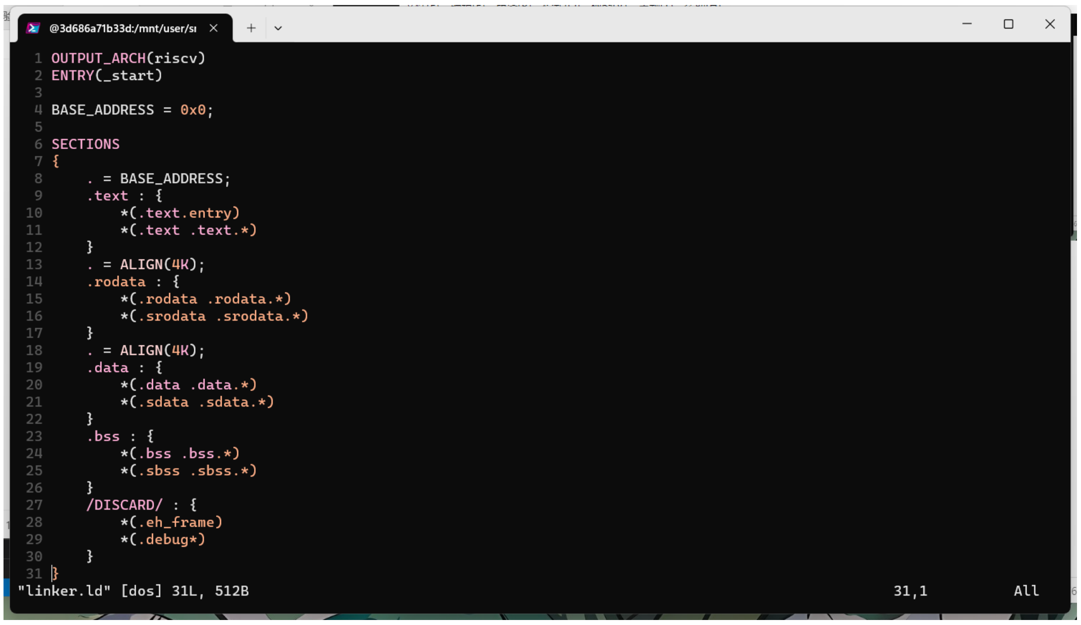


```rust
extern "C" {
    fn stext();
    fn etext();
    fn srodata();
    fn erodata();
    fn sdata();
    fn edata();
    fn sbss_with_stack();
    fn ebss();
    fn ekernel();
    fn strampoline();
}
```



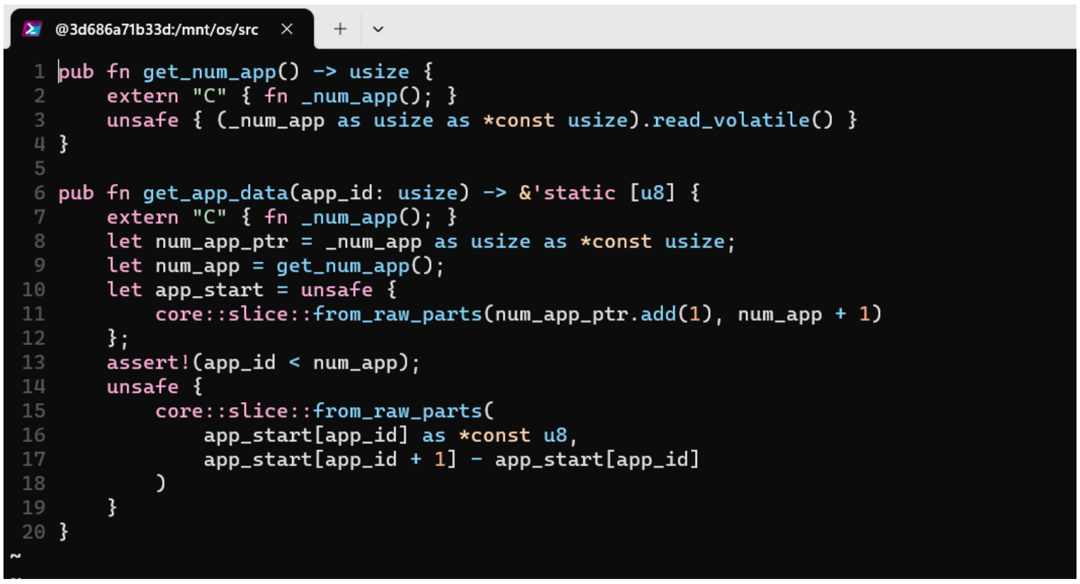
```rust
pub struct MemorySet {
    page_table: PageTable,
    areas: Vec<MapArea>,
}

impl MemorySet {

    pub fn new_kernel() -> Self {
        let mut memory_set = Self::new_bare();
        memory_set.map_trampoline();
        println!(".text [{:#x}, {:#x})", stext as usize, etext as usize);
        println!(".rodata [{:#x}, {:#x})", srodata as usize, erodata as usize);
        println!(".data [{:#x}, {:#x})", sdata as usize, edata as usize);
        println!(".bss [{:#x}, {:#x})", sbss_with_stack as usize, ebss as usize);
        println!("mapping .text section");
        memory_set.push(MapArea::new(
            (stext as usize).into(),
            (etext as usize).into(),
            MapType::Identical,
            MapPermission::R | MapPermission::X,
        ), None);
        println!("mapping .rodata section");
        memory_set.push(MapArea::new(
            (srodata as usize).into(),
            (erodata as usize).into(),
            MapType::Identical,
            MapPermission::R,
        ), None);
        println!("mapping .data section");
        memory_set.push(MapArea::new(
            (sdata as usize).into(),
                                                                69,1          13%
```

- 实现应用地址空间
  - 所有应用使用同一个链接脚本(user/src/linker.ld)



```
OUTPUT_ARCH(riscv)
ENTRY(_start)

BASE_ADDRESS = 0x0;

SECTIONS
{
    . = BASE_ADDRESS;
    .text : {
        *(.text.entry)
        *(.text .text.*)
    }
    . = ALIGN(4K);
    .rodata : {
        *(.rodata .rodata.*)
        *(.srodata .srodata.*)
    }
    . = ALIGN(4K);
    .data : {
        *(.data .data.*)
        *(.sdata .sdata.*)
    }
    .bss : {
        *(.bss .bss.*)
        *(.sbss .sbss.*)
    }
    /DISCARD/ : {
        *(.eh_frame)
        *(.debug*)
    }
}
"linker.ld" [dos] 31L, 512B                                    31,1          All
```

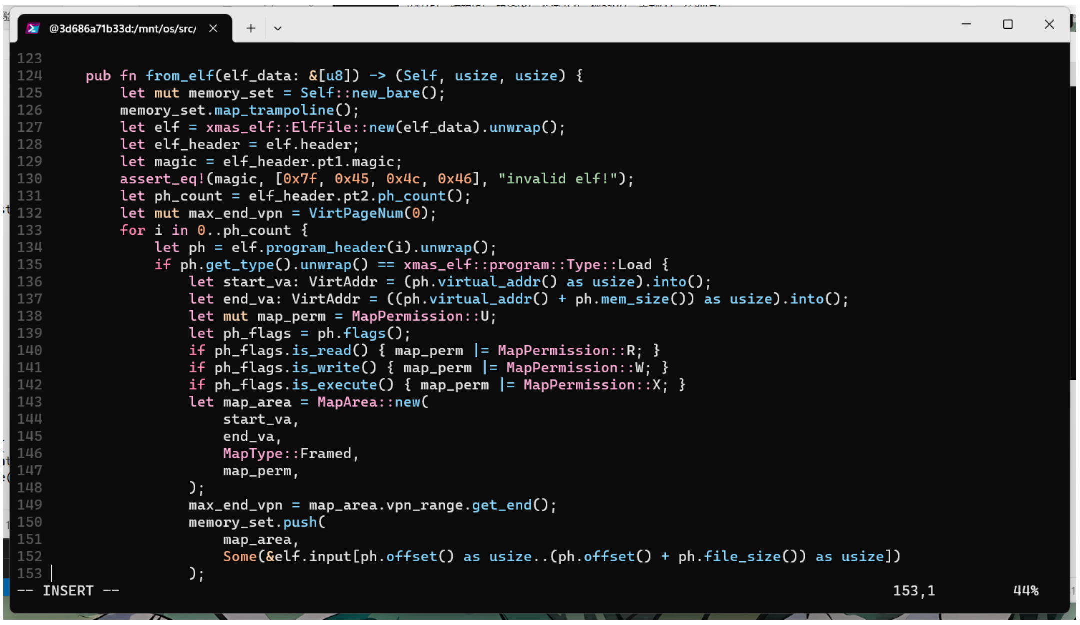
- 修改精简loader子模块(os/src/loader.rs)



@3d686a71b33d:/mnt/os/src

```rust
pub fn get_num_app() -> usize {
    extern "C" { fn _num_app(); }
    unsafe { (_num_app as usize as *const usize).read_volatile() }
}

pub fn get_app_data(app_id: usize) -> &'static [u8] {
    extern "C" { fn _num_app(); }
    let num_app_ptr = _num_app as usize as *const usize;
    let num_app = get_num_app();
    let app_start = unsafe {
        core::slice::from_raw_parts(num_app_ptr.add(1), num_app + 1)
    };
    assert!(app_id < num_app);
    unsafe {
        core::slice::from_raw_parts(
            app_start[app_id] as *const u8,
            app_start[app_id + 1] - app_start[app_id]
        )
    }
}
~
```

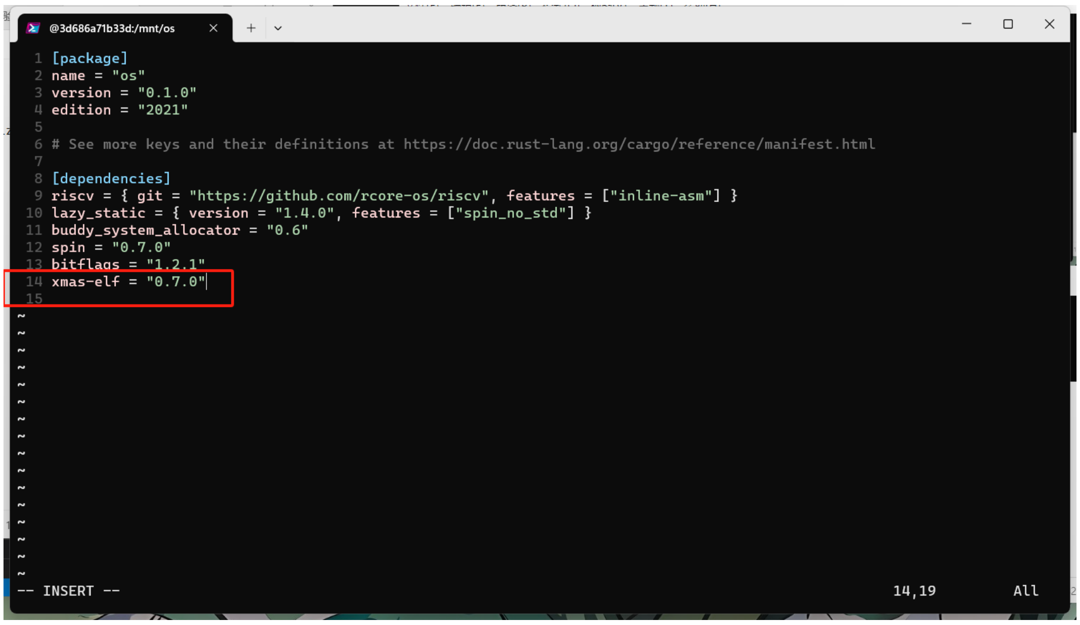
- 还需要解析ELF格式的数据，从而得到一个完整的应用地址空间(os/src/mm/memory_set.rs)



@3d686a71b33d:/mnt/os/src/

```rust

    pub fn from_elf(elf_data: &[u8]) -> (Self, usize, usize) {
        let mut memory_set = Self::new_bare();
        memory_set.map_trampoline();
        let elf = xmas_elf::ElfFile::new(elf_data).unwrap();
        let elf_header = elf.header;
        let magic = elf_header.pt1.magic;
        assert_eq!(magic, [0x7f, 0x45, 0x4c, 0x46], "invalid elf!");
        let ph_count = elf_header.pt2.ph_count();
        let mut max_end_vpn = VirtPageNum(0);
        for i in 0..ph_count {
            let ph = elf.program_header(i).unwrap();
            if ph.get_type().unwrap() == xmas_elf::program::Type::Load {
                let start_va: VirtAddr = (ph.virtual_addr() as usize).into();
                let end_va: VirtAddr = ((ph.virtual_addr() + ph.mem_size()) as usize).into();
                let mut map_perm = MapPermission::U;
                let ph_flags = ph.flags();
                if ph_flags.is_read() { map_perm |= MapPermission::R; }
                if ph_flags.is_write() { map_perm |= MapPermission::W; }
                if ph_flags.is_execute() { map_perm |= MapPermission::X; }
                let map_area = MapArea::new(
                    start_va,
                    end_va,
                    MapType::Framed,
                    map_perm,
                );
                max_end_vpn = map_area.vpn_range.get_end();
                memory_set.push(
                    map_area,
                    Some(&elf.input[ph.offset() as usize..(ph.offset() + ph.file_size()) as usize])
                );
-- INSERT --                                                                         153,1         44%
```

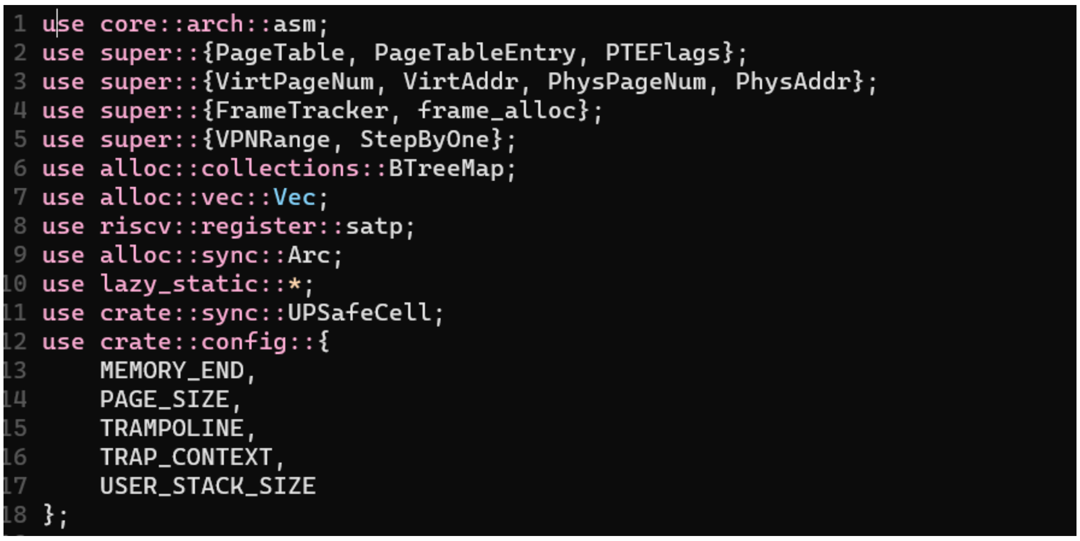
- 在配置文件Cargo.toml增加依赖(os/Cargo.toml)



@3d686a71b33d:/mnt/os

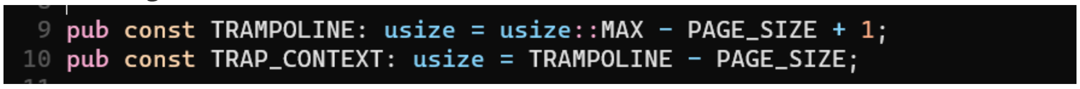
```toml
[package]
name = "os"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
buddy_system_allocator = "0.6"
spin = "0.7.0"
bitflags = "1.2.1"
xmas-elf = "0.7.0"

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                                         14,19         All
```

- 实现memory_set子模块增加代码(os/src/mm/memory_set.rs)



```rust
use core::arch::asm;
use super::{PageTable, PageTableEntry, PTEFlags};
use super::{VirtPageNum, VirtAddr, PhysPageNum, PhysAddr};
use super::{FrameTracker, frame_alloc};
use super::{VPNRange, StepByOne};
use alloc::collections::BTreeMap;
use alloc::vec::Vec;
use riscv::register::satp;
use alloc::sync::Arc;
use lazy_static::*;
use crate::sync::UPSafeCell;
use crate::config::{
    MEMORY_END,
    PAGE_SIZE,
    TRAMPOLINE,
    TRAP_CONTEXT,
    USER_STACK_SIZE
};
```

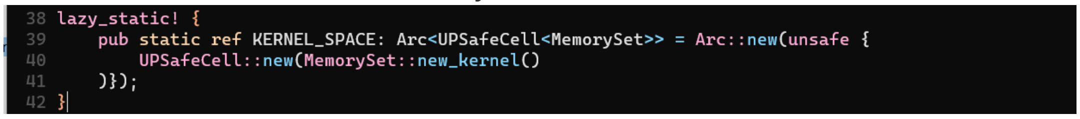
- os/src/config.rs



```rust
pub const TRAMPOLINE: usize = usize::MAX - PAGE_SIZE + 1;
pub const TRAP_CONTEXT: usize = TRAMPOLINE - PAGE_SIZE;
```

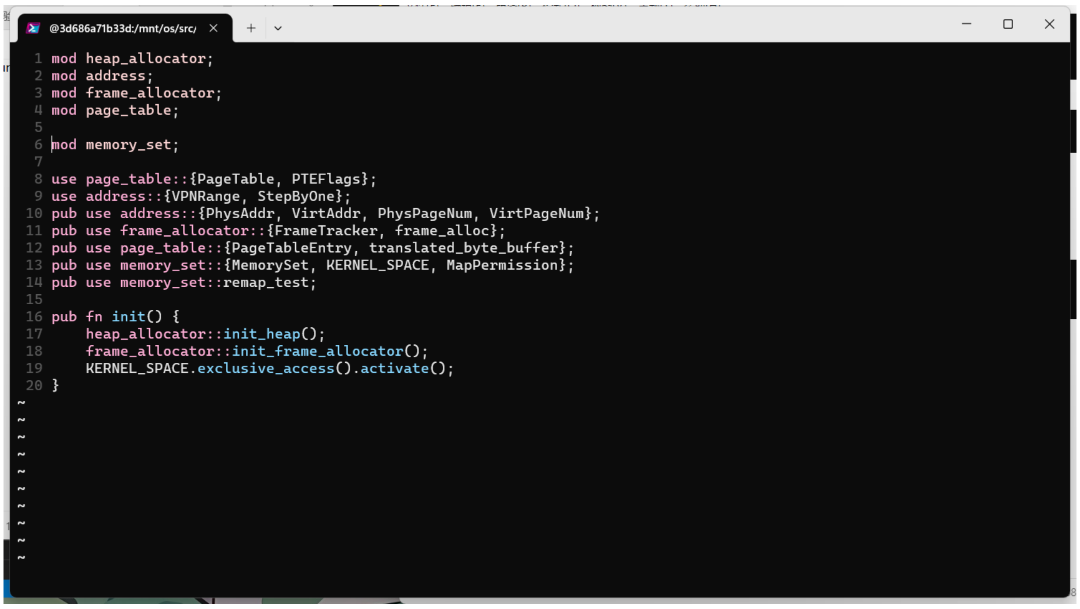
1.7 实现基于地址空间的分时多任务

- 建立基于分页模式的虚拟地址空间
  - 创建内核地址空间。(os/src/mm/memory_set.rs)



```rust
lazy_static! {
    pub static ref KERNEL_SPACE: Arc<UPSafeCell<MemorySet>> = Arc::new(unsafe {
        UPSafeCell::new(MemorySet::new_kernel()
    )});
}
```

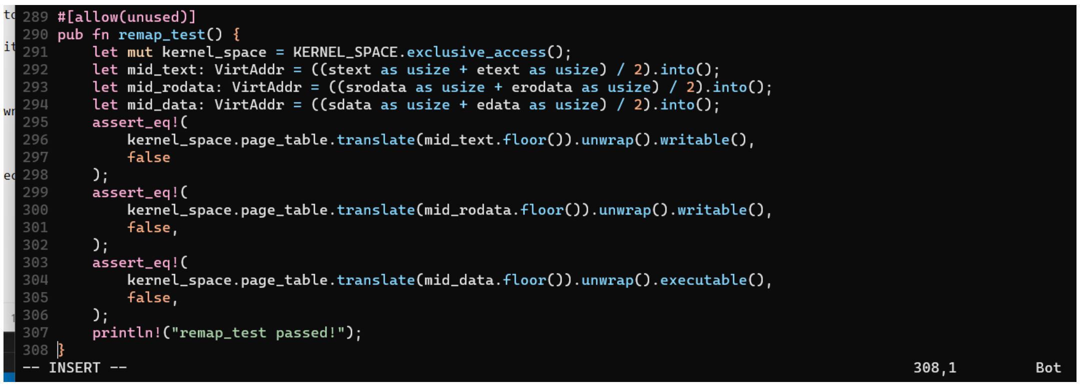
  - 在rust_main中进行内存管理子系统的初始化(os/src/mm/mod.rs)



```
@3d686a71b33d:/mnt/os/src/
```

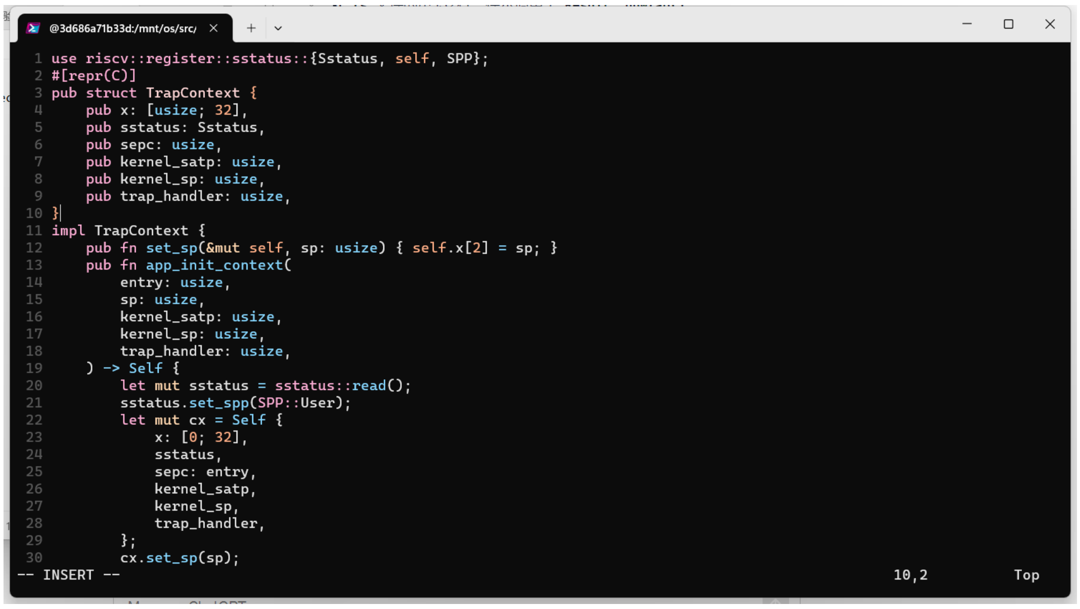
```rust
mod heap_allocator;
mod address;
mod frame_allocator;
mod page_table;

mod memory_set;

use page_table::{PageTable, PTEFlags};
use address::{VPNRange, StepByOne};
pub use address::{PhysAddr, VirtAddr, PhysPageNum, VirtPageNum};
pub use frame_allocator::{FrameTracker, frame_alloc};
pub use page_table::{PageTableEntry, translated_byte_buffer};
pub use memory_set::{MemorySet, KERNEL_SPACE, MapPermission};
pub use memory_set::remap_test;

pub fn init() {
    heap_allocator::init_heap();
    frame_allocator::init_frame_allocator();
    KERNEL_SPACE.exclusive_access().activate();
}
~
~
~
~
~
~
~
~
~
~
```

  - 检查内核地址空间的多级页表设置(os/src/mm/memory_set.rs)



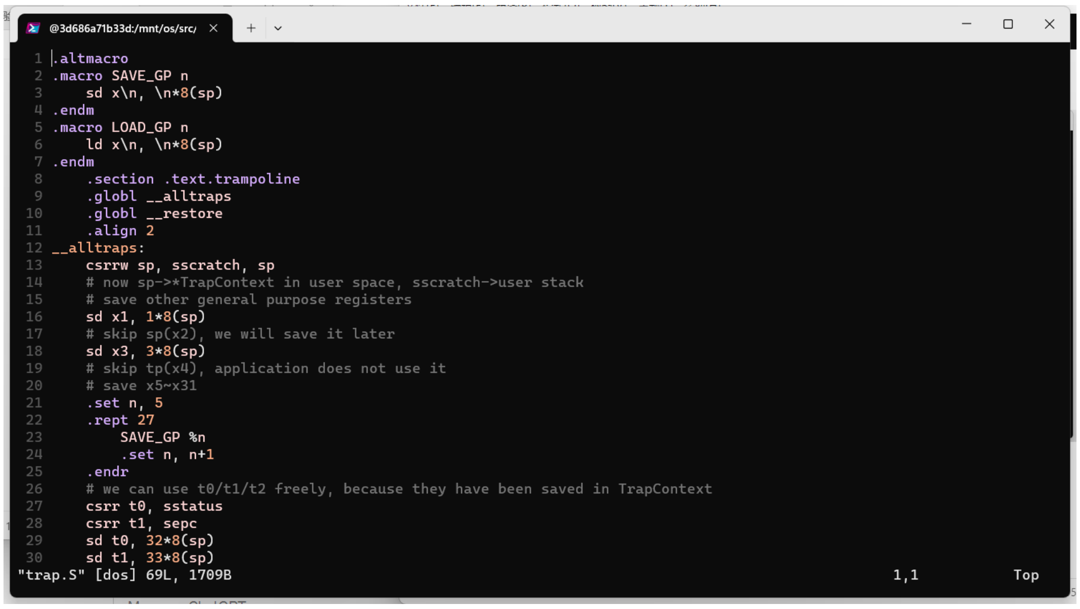
```rust
#[allow(unused)]
pub fn remap_test() {
    let mut kernel_space = KERNEL_SPACE.exclusive_access();
    let mid_text: VirtAddr = ((stext as usize + etext as usize) / 2).into();
    let mid_rodata: VirtAddr = ((srodata as usize + erodata as usize) / 2).into();
    let mid_data: VirtAddr = ((sdata as usize + edata as usize) / 2).into();
    assert_eq!(
        kernel_space.page_table.translate(mid_text.floor()).unwrap().writable(),
        false
    );
    assert_eq!(
        kernel_space.page_table.translate(mid_rodata.floor()).unwrap().writable(),
        false,
    );
    assert_eq!(
        kernel_space.page_table.translate(mid_data.floor()).unwrap().executable(),
        false,
    );
    println!("remap_test passed!");
}
-- INSERT --                                                        308,1         Bot
```

- 实现跳板机制
    - 扩展Trap上下文(os/src/trap/context.rs)



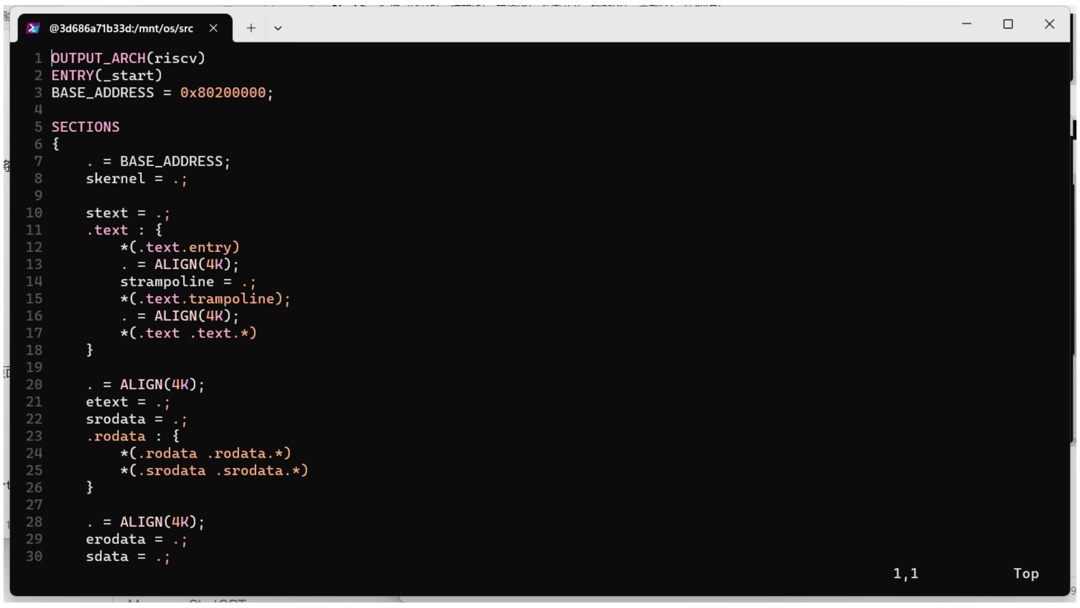
```rust
use riscv::register::sstatus::{Sstatus, self, SPP};
#[repr(C)]
pub struct TrapContext {
    pub x: [usize; 32],
    pub sstatus: Sstatus,
    pub sepc: usize,
    pub kernel_satp: usize,
    pub kernel_sp: usize,
    pub trap_handler: usize,
}
impl TrapContext {
    pub fn set_sp(&mut self, sp: usize) { self.x[2] = sp; }
    pub fn app_init_context(
        entry: usize,
        sp: usize,
        kernel_satp: usize,
        kernel_sp: usize,
        trap_handler: usize,
    ) -> Self {
        let mut sstatus = sstatus::read();
        sstatus.set_spp(SPP::User);
        let mut cx = Self {
            x: [0; 32],
            sstatus,
            sepc: entry,
            kernel_satp,
            kernel_sp,
            trap_handler,
        };
        cx.set_sp(sp);
-- INSERT --                                                                  10,2          Top
```

    - 实现地址空间的切换(os/src/trap/trap.S)



```asm
.altmacro
.macro SAVE_GP n
    sd x\n, \n*8(sp)
.endm
.macro LOAD_GP n
    ld x\n, \n*8(sp)
.endm
    .section .text.trampoline
    .globl __alltraps
    .globl __restore
    .align 2
__alltraps:
    csrrw sp, sscratch, sp
    # now sp->*TrapContext in user space, sscratch->user stack
    # save other general purpose registers
    sd x1, 1*8(sp)
    # skip sp(x2), we will save it later
    sd x3, 3*8(sp)
    # skip tp(x4), application does not use it
    # save x5~x31
    .set n, 5
    .rept 27
        SAVE_GP %n
        .set n, n+1
    .endr
    # we can use t0/t1/t2 freely, because they have been saved in TrapContext
    csrr t0, sstatus
    csrr t1, sepc
    sd t0, 32*8(sp)
    sd t1, 33*8(sp)
"trap.S" [dos] 69L, 1709B                                                    1,1           Top
```

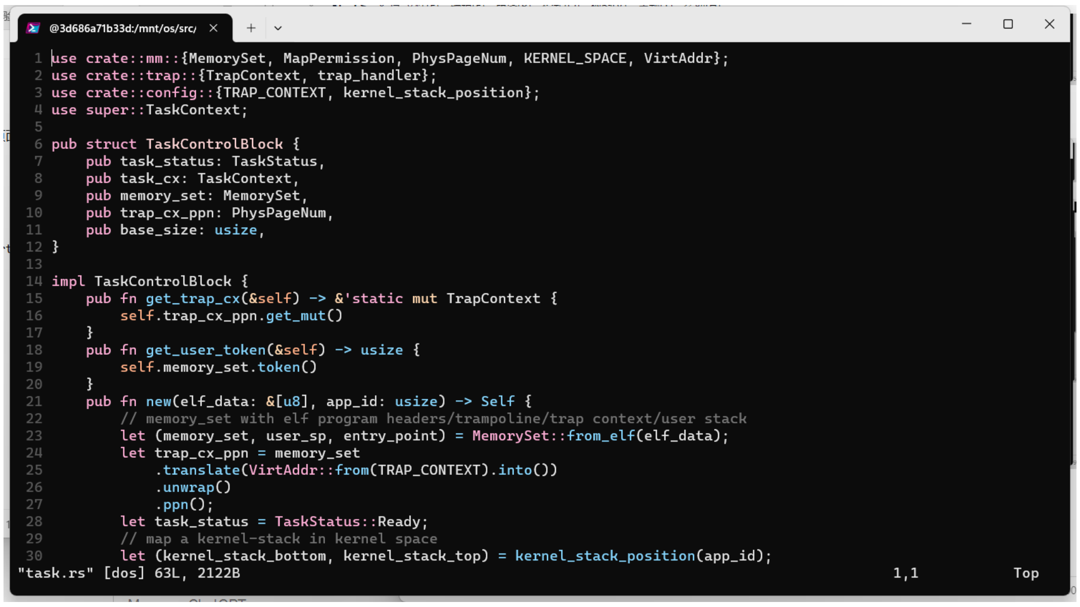
    - 将 trap.S 中的整段汇编代码放置在 .text.trampoline 段，并在调整内存布局的时候将它对齐到代码段的一个页面中。(os/src/linker.ld)



```ld
OUTPUT_ARCH(riscv)
ENTRY(_start)
BASE_ADDRESS = 0x80200000;

SECTIONS
{
    . = BASE_ADDRESS;
    skernel = .;

    stext = .;
    .text : {
        *(.text.entry)
        . = ALIGN(4K);
        strampoline = .;
        *(.text.trampoline);
        . = ALIGN(4K);
        *(.text .text.*)
    }

    . = ALIGN(4K);
    etext = .;
    srodata = .;
    .rodata : {
        *(.rodata .rodata.*)
        *(.srodata .srodata.*)
    }

    . = ALIGN(4K);
    erodata = .;
    sdata = .;
                                                                             1,1           Top
```

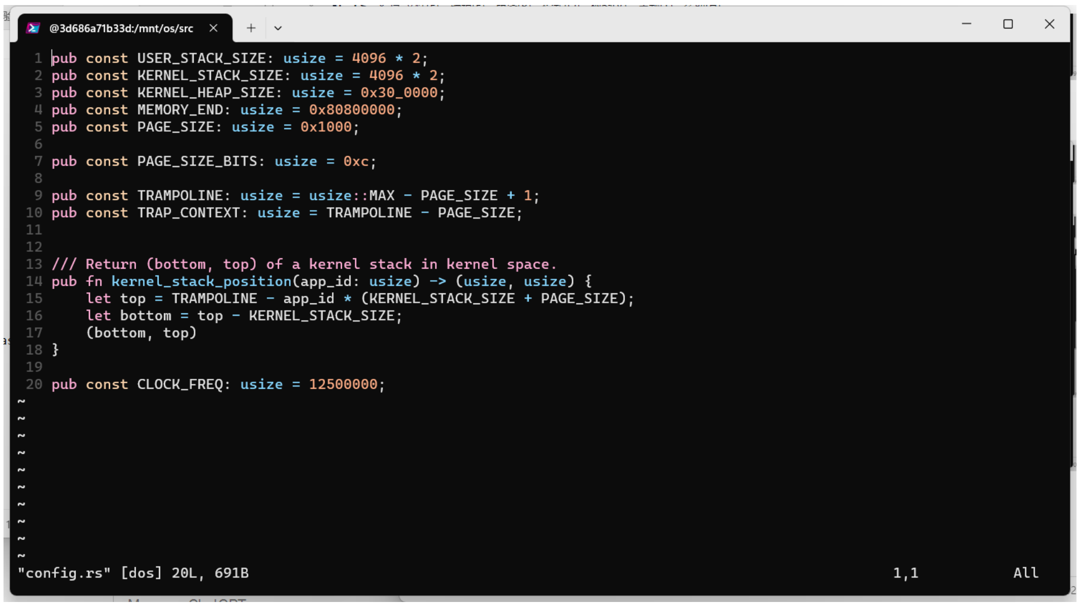
- 加载和执行应用程序

- 修改任务子模块，并更新任务控制块的管理。(os/src/task/task.rs)



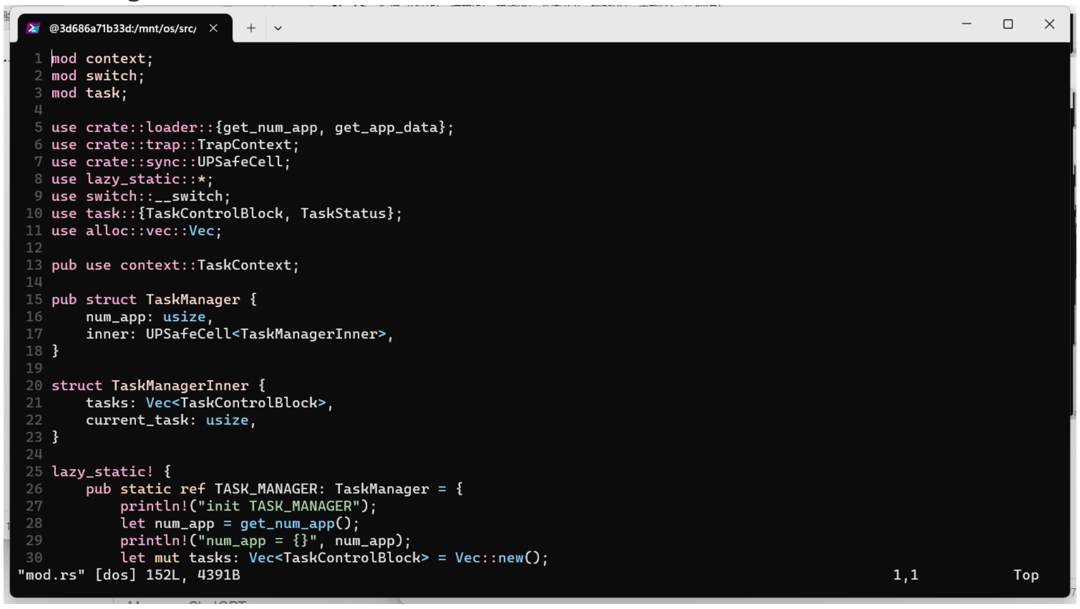
```
use crate::mm::{MemorySet, MapPermission, PhysPageNum, KERNEL_SPACE, VirtAddr};
use crate::trap::{TrapContext, trap_handler};
use crate::config::{TRAP_CONTEXT, kernel_stack_position};
use super::TaskContext;

pub struct TaskControlBlock {
    pub task_status: TaskStatus,
    pub task_cx: TaskContext,
    pub memory_set: MemorySet,
    pub trap_cx_ppn: PhysPageNum,
    pub base_size: usize,
}

impl TaskControlBlock {
    pub fn get_trap_cx(&self) -> &'static mut TrapContext {
        self.trap_cx_ppn.get_mut()
    }
    pub fn get_user_token(&self) -> usize {
        self.memory_set.token()
    }
    pub fn new(elf_data: &[u8], app_id: usize) -> Self {
        // memory_set with elf program headers/trampoline/trap context/user stack
        let (memory_set, user_sp, entry_point) = MemorySet::from_elf(elf_data);
        let trap_cx_ppn = memory_set
            .translate(VirtAddr::from(TRAP_CONTEXT).into())
            .unwrap()
            .ppn();
        let task_status = TaskStatus::Ready;
        // map a kernel-stack in kernel space
        let (kernel_stack_bottom, kernel_stack_top) = kernel_stack_position(app_id);
"task.rs" [dos] 63L, 2122B                                                  1,1           Top
```

- os/src/config.rs



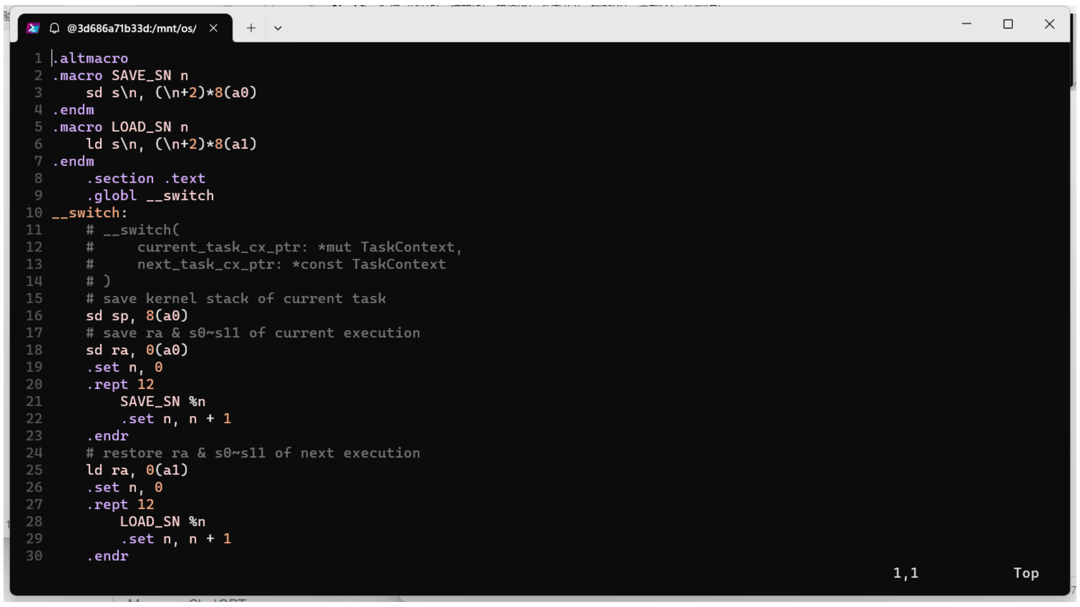
```
pub const USER_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_HEAP_SIZE: usize = 0x30_0000;
pub const MEMORY_END: usize = 0x80800000;
pub const PAGE_SIZE: usize = 0x1000;

pub const PAGE_SIZE_BITS: usize = 0xc;

pub const TRAMPOLINE: usize = usize::MAX - PAGE_SIZE + 1;
pub const TRAP_CONTEXT: usize = TRAMPOLINE - PAGE_SIZE;


/// Return (bottom, top) of a kernel stack in kernel space.
pub fn kernel_stack_position(app_id: usize) -> (usize, usize) {
    let top = TRAMPOLINE - app_id * (KERNEL_STACK_SIZE + PAGE_SIZE);
    let bottom = top - KERNEL_STACK_SIZE;
    (bottom, top)
}

pub const CLOCK_FREQ: usize = 12500000;
~
~
~
~
~
~
~
~
~
~
"config.rs" [dos] 20L, 691B                                                 1,1           All
```

- 在内核初始化的时候，需要将所有的应用程序加载到全局应用管理器中。同时，也要修改TaskManager的实现。(os/src/task/mod.rs)



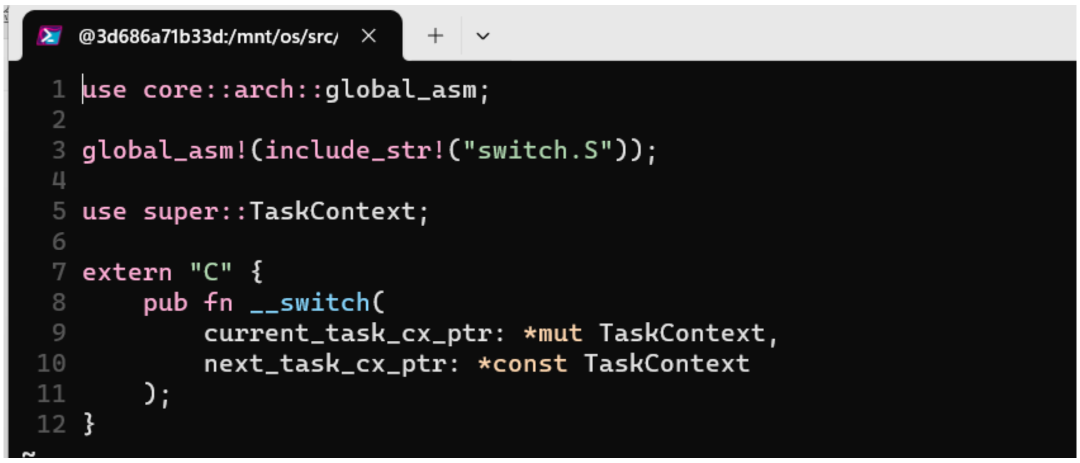
```
mod context;
mod switch;
mod task;

use crate::loader::{get_num_app, get_app_data};
use crate::trap::TrapContext;
use crate::sync::UPSafeCell;
use lazy_static::*;
use switch::__switch;
use task::{TaskControlBlock, TaskStatus};
use alloc::vec::Vec;

pub use context::TaskContext;

pub struct TaskManager {
    num_app: usize,
    inner: UPSafeCell<TaskManagerInner>,
}

struct TaskManagerInner {
    tasks: Vec<TaskControlBlock>,
    current_task: usize,
}

lazy_static! {
    pub static ref TASK_MANAGER: TaskManager = {
        println!("init TASK_MANAGER");
        let num_app = get_num_app();
        println!("num_app = {}", num_app);
        let mut tasks: Vec<TaskControlBlock> = Vec::new();
"mod.rs" [dos] 152L, 4391B                                                  1,1           Top
```

- 修改/os/src/task/switch.S



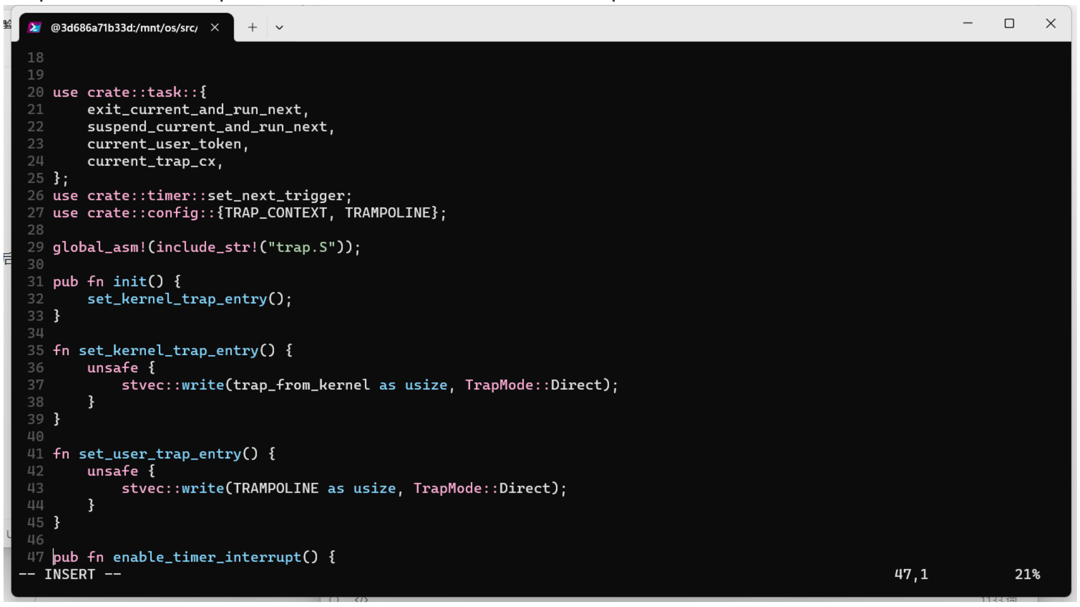
```
.altmacro
.macro SAVE_SN n
    sd s\n, (\n+2)*8(a0)
.endm
.macro LOAD_SN n
    ld s\n, (\n+2)*8(a1)
.endm
    .section .text
    .globl __switch
__switch:
    # __switch(
    #     current_task_cx_ptr: *mut TaskContext,
    #     next_task_cx_ptr: *const TaskContext
    # )
    # save kernel stack of current task
    sd sp, 8(a0)
    # save ra & s0~s11 of current execution
    sd ra, 0(a0)
    .set n, 0
    .rept 12
        SAVE_SN %n
        .set n, n + 1
    .endr
    # restore ra & s0~s11 of next execution
    ld ra, 0(a1)
    .set n, 0
    .rept 12
        LOAD_SN %n
        .set n, n + 1
    .endr
                                                                1,1           Top
```

- 修改switch.rs



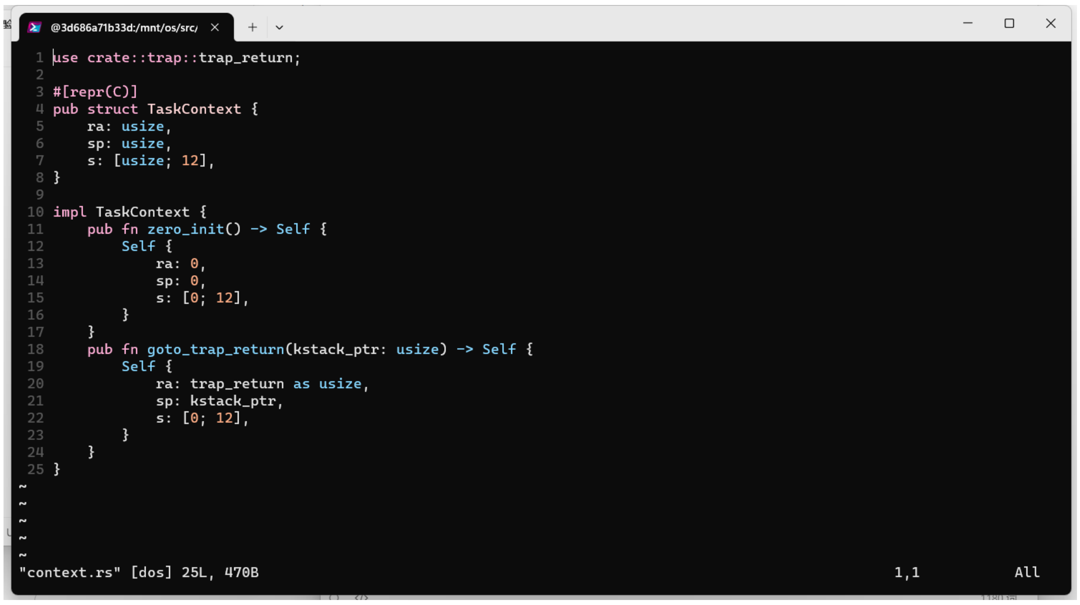
```
use core::arch::global_asm;

global_asm!(include_str!("switch.S"));

use super::TaskContext;

extern "C" {
    pub fn __switch(
        current_task_cx_ptr: *mut TaskContext,
        next_task_cx_ptr: *const TaskContext
    );
}
```

- 改进Trap的处理
  - 首先修改init函数。然后再trap_handler的开头增加set_kernel_trap_entry。同时，在处理完trap后还要调用trap_return 返回用户态。(os/src/trap/mod.rs)



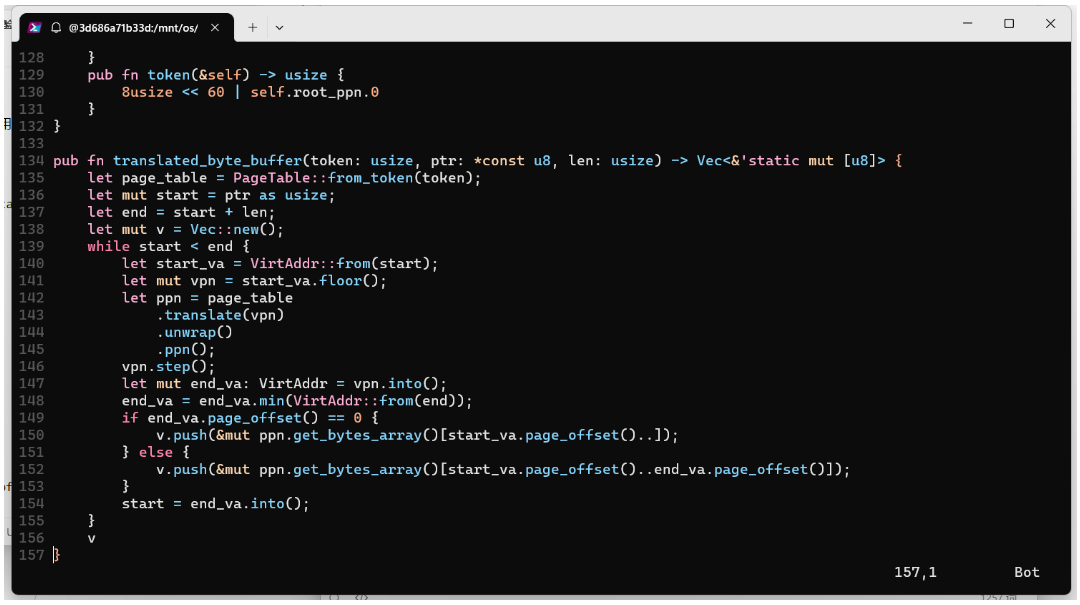
```


use crate::task::{
    exit_current_and_run_next,
    suspend_current_and_run_next,
    current_user_token,
    current_trap_cx,
};
use crate::timer::set_next_trigger;
use crate::config::{TRAP_CONTEXT, TRAMPOLINE};

global_asm!(include_str!("trap.S"));

pub fn init() {
    set_kernel_trap_entry();
}

fn set_kernel_trap_entry() {
    unsafe {
        stvec::write(trap_from_kernel as usize, TrapMode::Direct);
    }
}

fn set_user_trap_entry() {
    unsafe {
        stvec::write(TRAMPOLINE as usize, TrapMode::Direct);
    }
}

pub fn enable_timer_interrupt() {
-- INSERT --                                                    47,1          21%
```

- 在每一个应用程序第一次获得CPU权限时，内核栈顶放置在内核加载应用的时候构造的一个任务上下文。(os/src/task/context.rs)



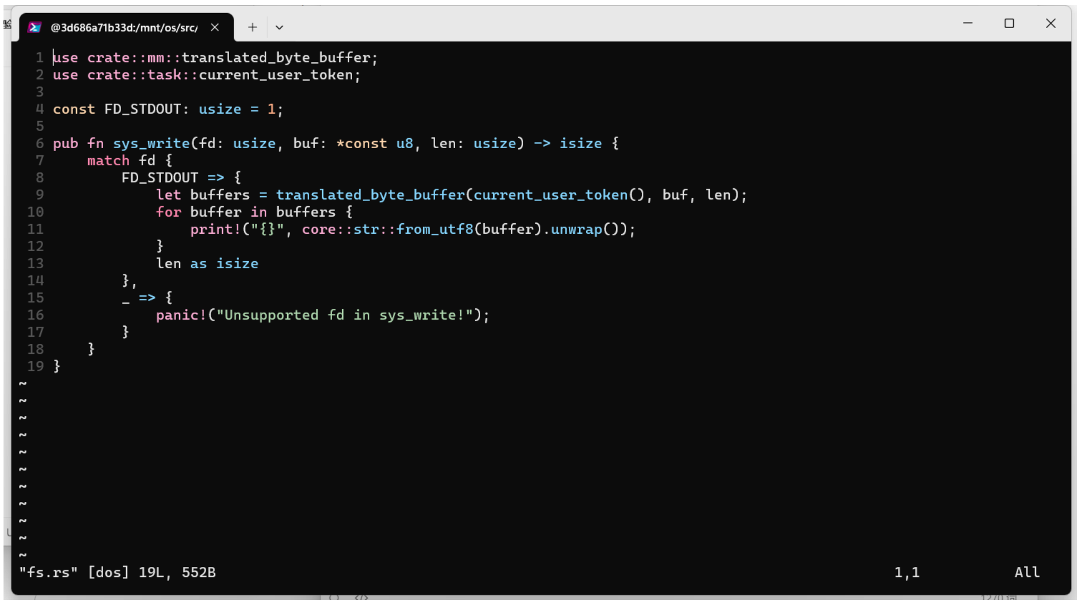
```
use crate::trap::trap_return;

#[repr(C)]
pub struct TaskContext {
    ra: usize,
    sp: usize,
    s: [usize; 12],
}

impl TaskContext {
    pub fn zero_init() -> Self {
        Self {
            ra: 0,
            sp: 0,
            s: [0; 12],
        }
    }
    pub fn goto_trap_return(kstack_ptr: usize) -> Self {
        Self {
            ra: trap_return as usize,
            sp: kstack_ptr,
            s: [0; 12],
        }
    }
}
~
~
~
~
~
"context.rs" [dos] 25L, 470B                                        1,1           All
```

- 改进sys_write的实现
  - 由于地址空间的隔离，sys_write无法直接方法应用空间的数据。为此，页表page_table提供一个将应用地址空间的缓冲区转化为内核地址空间可以直接访问的辅助函数。(os/src/mm/page_table.rs)



```
    }
    pub fn token(&self) -> usize {
        8usize << 60 | self.root_ppn.0
    }
}

pub fn translated_byte_buffer(token: usize, ptr: *const u8, len: usize) -> Vec<&'static mut [u8]> {
    let page_table = PageTable::from_token(token);
    let mut start = ptr as usize;
    let end = start + len;
    let mut v = Vec::new();
    while start < end {
        let start_va = VirtAddr::from(start);
        let mut vpn = start_va.floor();
        let ppn = page_table
            .translate(vpn)
            .unwrap()
            .ppn();
        vpn.step();
        let mut end_va: VirtAddr = vpn.into();
        end_va = end_va.min(VirtAddr::from(end));
        if end_va.page_offset() == 0 {
            v.push(&mut ppn.get_bytes_array()[start_va.page_offset()..]);
        } else {
            v.push(&mut ppn.get_bytes_array()[start_va.page_offset()..end_va.page_offset()]);
        }
        start = end_va.into();
    }
    v
}
                                                                    157,1         Bot
```

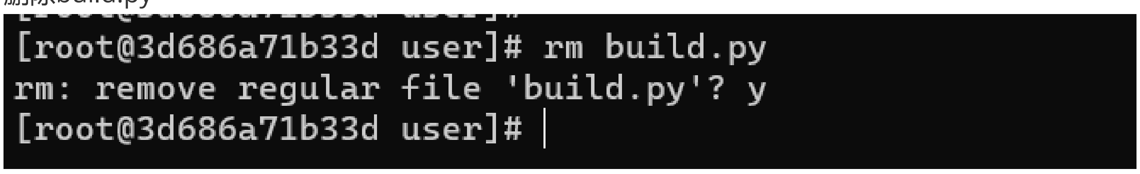
- 修改sys_write系统调用。(os/src/syscall/fs.rs)



```
use crate::mm::translated_byte_buffer;
use crate::task::current_user_token;

const FD_STDOUT: usize = 1;

pub fn sys_write(fd: usize, buf: *const u8, len: usize) -> isize {
    match fd {
        FD_STDOUT => {
            let buffers = translated_byte_buffer(current_user_token(), buf, len);
            for buffer in buffers {
                print!("{}", core::str::from_utf8(buffer).unwrap());
            }
            len as isize
        },
        _ => {
            panic!("Unsupported fd in sys_write!");
        }
    }
}
~
"fs.rs" [dos] 19L, 552B                                                  1,1           All
```

1.8 修改应用程序

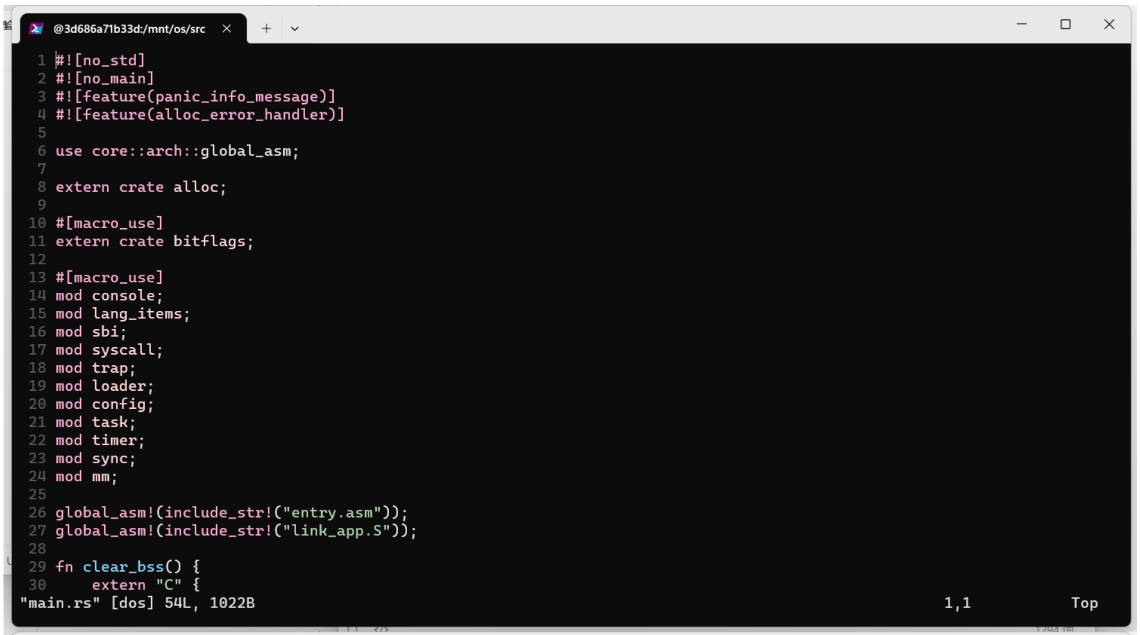
- 删除user/src/lib.rs中的clear_bss()
- 删除build.py



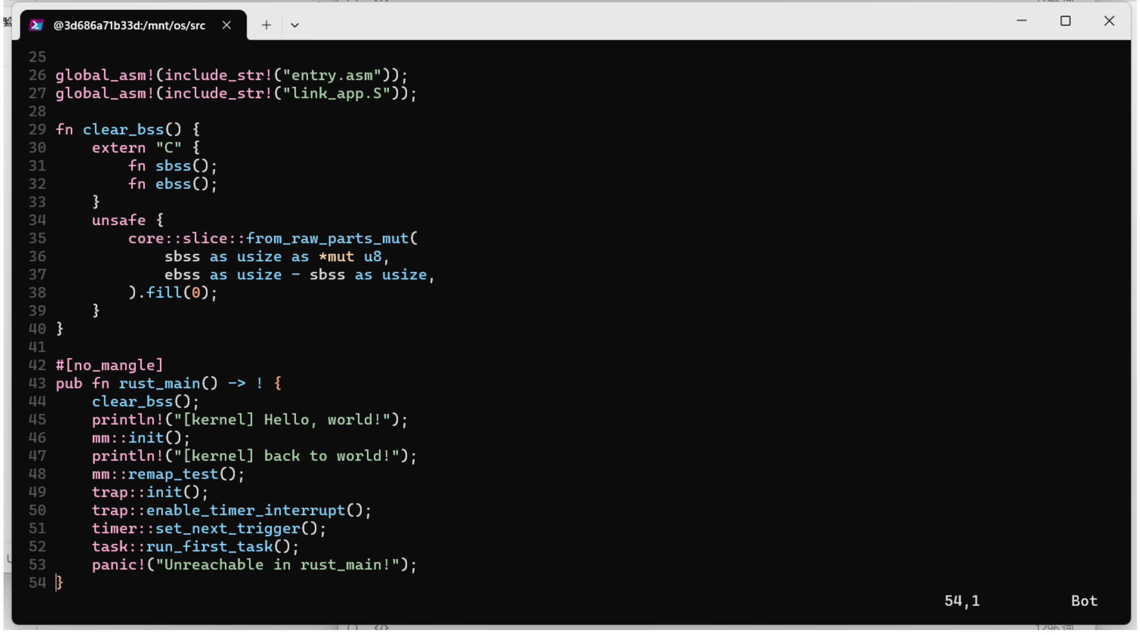
```
[root@3d686a71b33d user]# rm build.py
rm: remove regular file 'build.py'? y
[root@3d686a71b33d user]#
```

1.9 修改main.rs

- os/src/main.rs

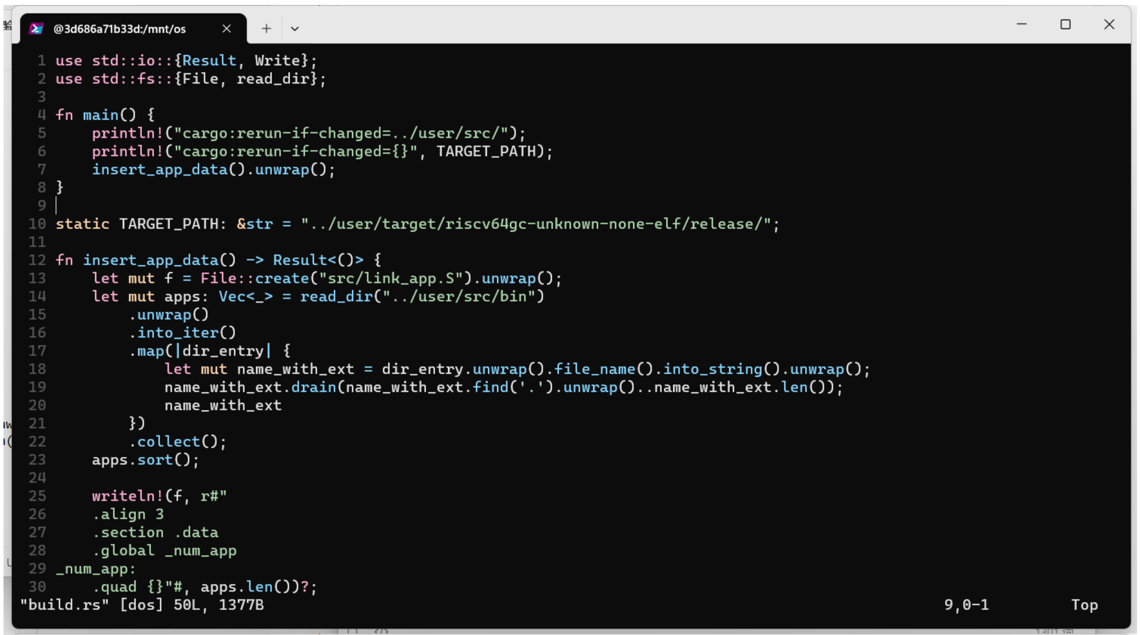


```
#![no_std]
#![no_main]
#![feature(panic_info_message)]
#![feature(alloc_error_handler)]

use core::arch::global_asm;

extern crate alloc;

#[macro_use]
extern crate bitflags;

#[macro_use]
mod console;
mod lang_items;
mod sbi;
mod syscall;
mod trap;
mod loader;
mod config;
mod task;
mod timer;
mod sync;
mod mm;

global_asm!(include_str!("entry.asm"));
global_asm!(include_str!("link_app.S"));

fn clear_bss() {
    extern "C" {
"main.rs" [dos] 54L, 1022B                                        1,1           Top
```

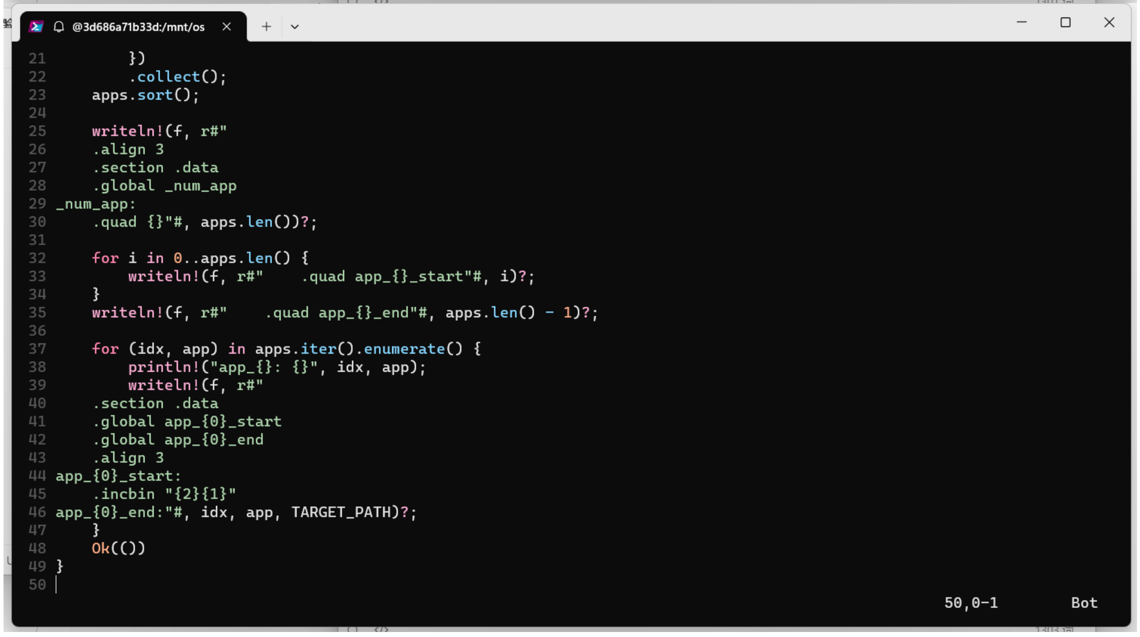


```

global_asm!(include_str!("entry.asm"));
global_asm!(include_str!("link_app.S"));

fn clear_bss() {
    extern "C" {
        fn sbss();
        fn ebss();
    }
    unsafe {
        core::slice::from_raw_parts_mut(
            sbss as usize as *mut u8,
            ebss as usize - sbss as usize,
        ).fill(0);
    }
}

#[no_mangle]
pub fn rust_main() -> ! {
    clear_bss();
    println!("[kernel] Hello, world!");
    mm::init();
    println!("[kernel] back to world!");
    mm::remap_test();
    trap::init();
    trap::enable_timer_interrupt();
    timer::set_next_trigger();
    task::run_first_task();
    panic!("Unreachable in rust_main!");
}
                                                                  54,1          Bot
```

- 修改os/build.rs

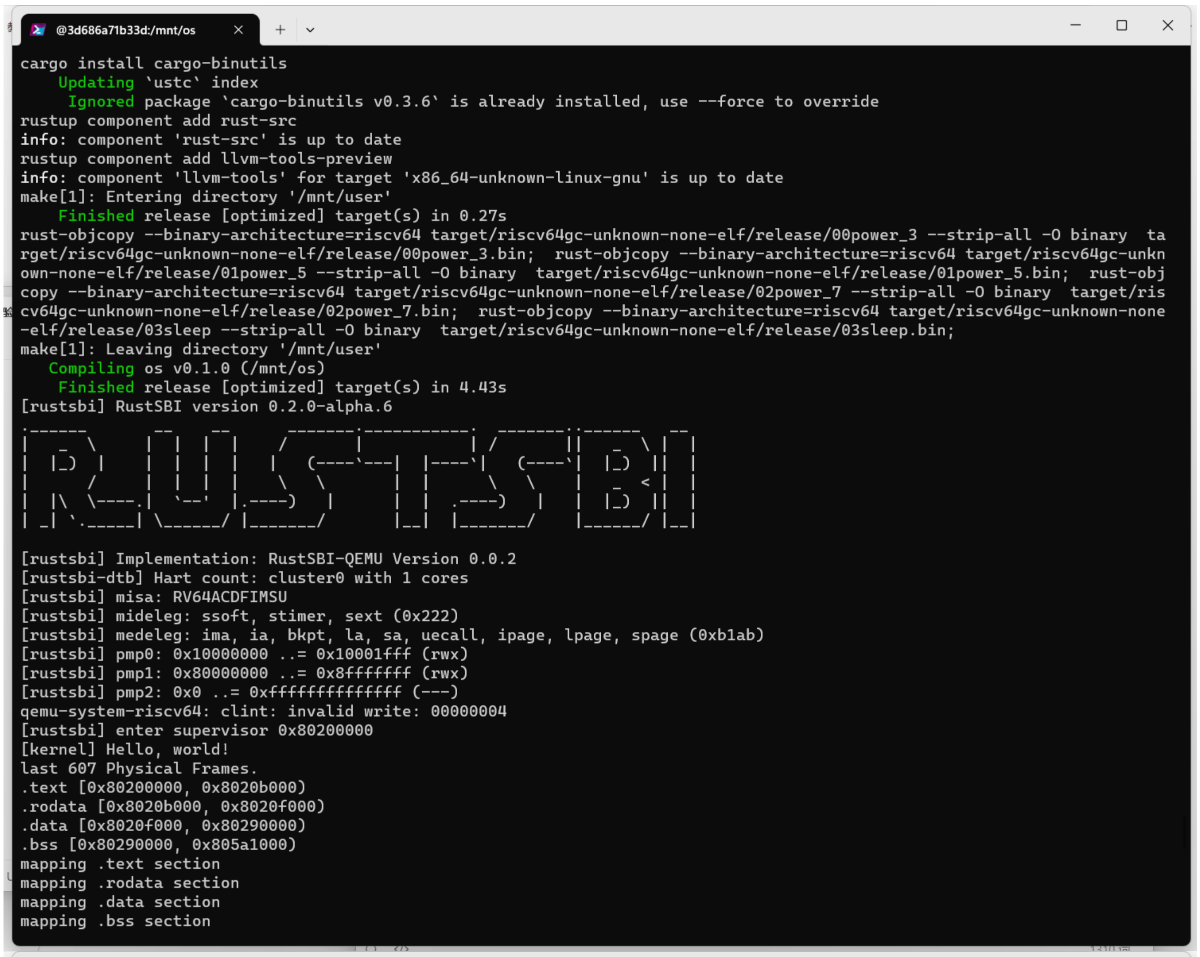


```
use std::io::{Result, Write};
use std::fs::{File, read_dir};

fn main() {
    println!("cargo:rerun-if-changed=../user/src/");
    println!("cargo:rerun-if-changed={}", TARGET_PATH);
    insert_app_data().unwrap();
}

static TARGET_PATH: &str = "../user/target/riscv64gc-unknown-none-elf/release/";

fn insert_app_data() -> Result<()> {
    let mut f = File::create("src/link_app.S").unwrap();
    let mut apps: Vec<_> = read_dir("../user/src/bin")
        .unwrap()
        .into_iter()
        .map(|dir_entry| {
            let mut name_with_ext = dir_entry.unwrap().file_name().into_string().unwrap();
            name_with_ext.drain(name_with_ext.find('.').unwrap()..name_with_ext.len());
            name_with_ext
        })
        .collect();
    apps.sort();

    writeln!(f, r#"
    .align 3
    .section .data
    .global _num_app
_num_app:
    .quad {}"#, apps.len())?;
"build.rs" [dos] 50L, 1377B                                                          9,0-1         Top
```

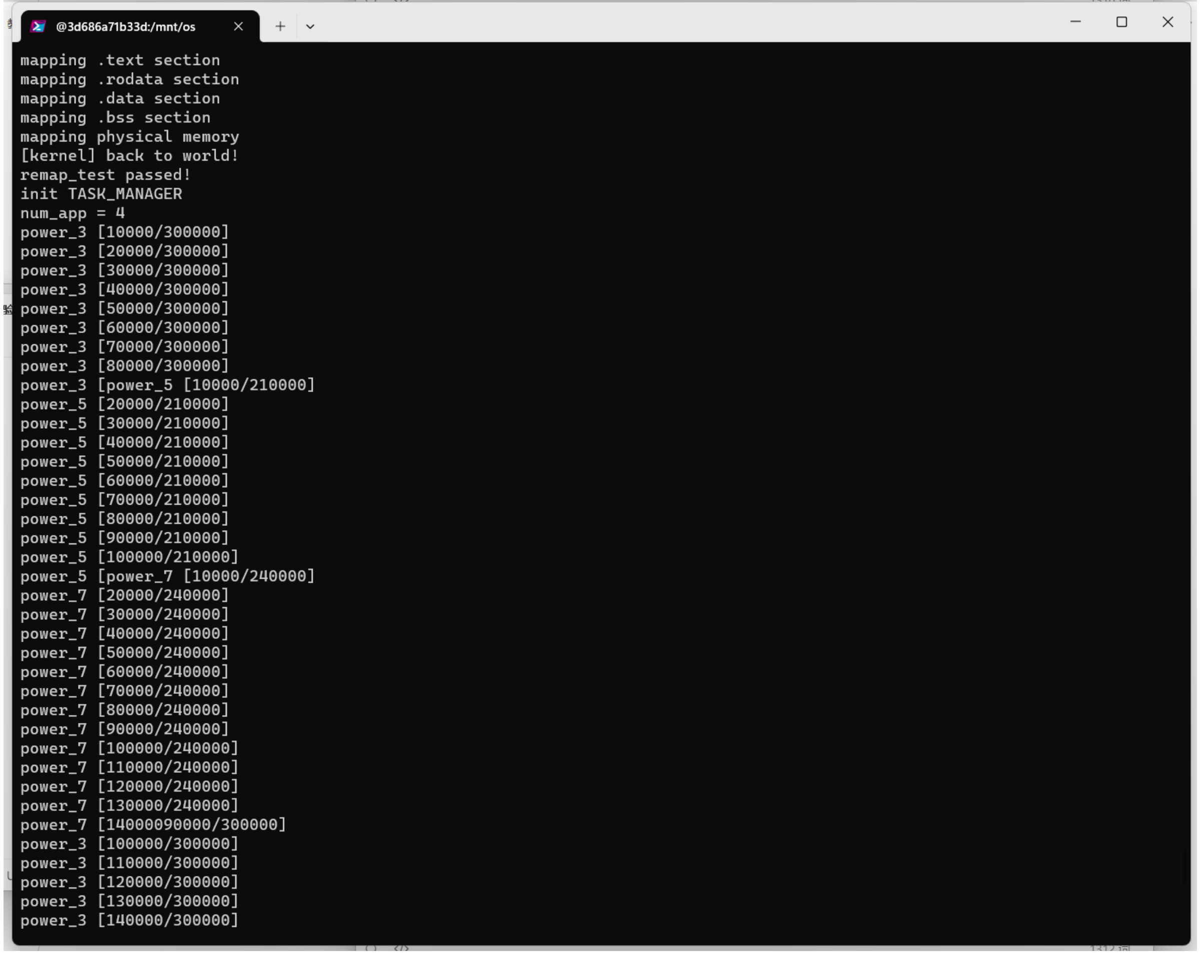


```
        })
        .collect();
    apps.sort();

    writeln!(f, r#"
    .align 3
    .section .data
    .global _num_app
_num_app:
    .quad {}"#, apps.len())?;

    for i in 0..apps.len() {
        writeln!(f, r#"    .quad app_{}_start"#, i)?;
    }
    writeln!(f, r#"    .quad app_{}_end"#, apps.len() - 1)?;

    for (idx, app) in apps.iter().enumerate() {
        println!("app_{}: {}", idx, app);
        writeln!(f, r#"
    .section .data
    .global app_{0}_start
    .global app_{0}_end
    .align 3
app_{0}_start:
    .incbin "{2}{1}"
app_{0}_end:"#, idx, app, TARGET_PATH)?;
    }
    Ok(())
}

                                                                                     50,0-1        Bot
```

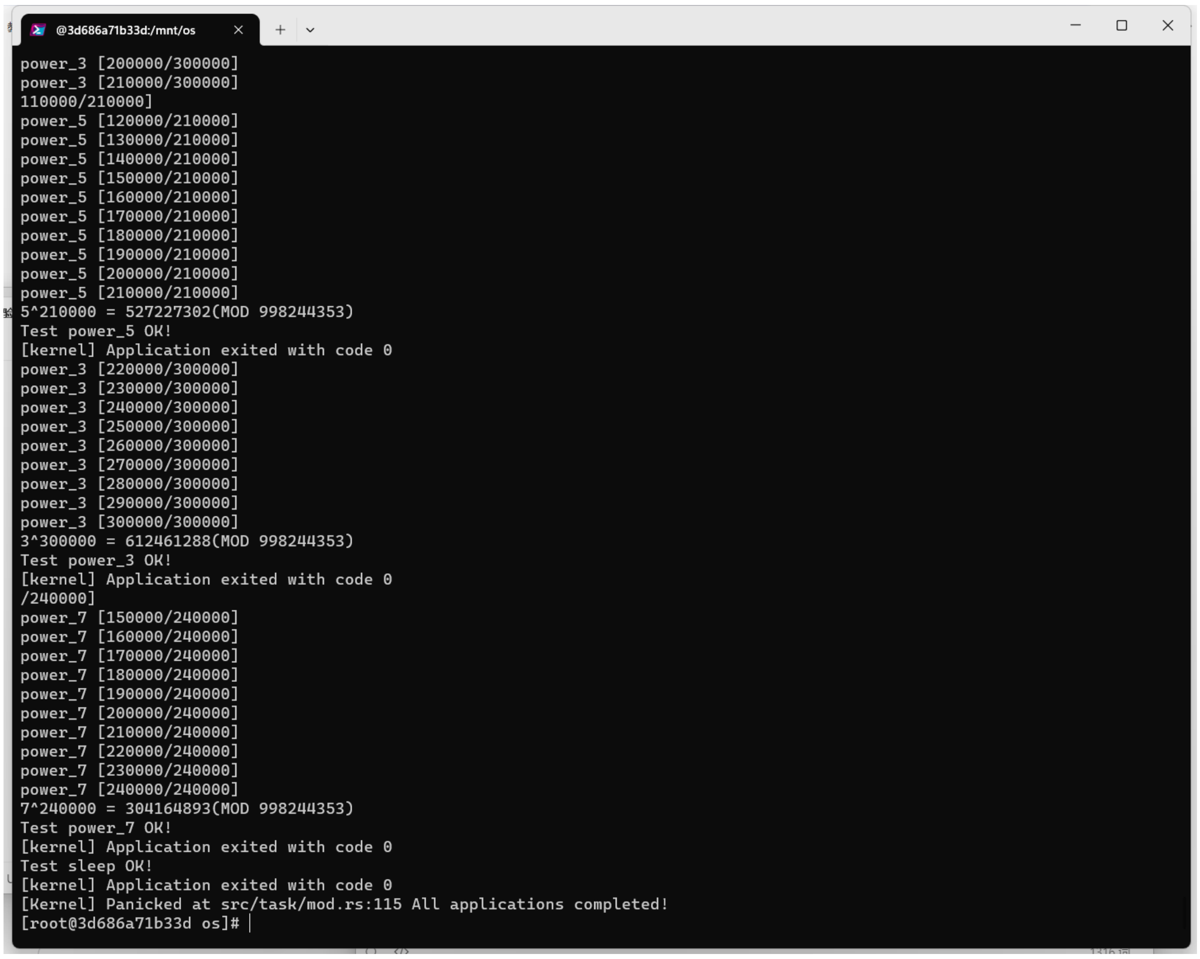
1.10 运行结果



```
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' for target 'x86_64-unknown-linux-gnu' is up to date
make[1]: Entering directory '/mnt/user'
    Finished release [optimized] target(s) in 0.27s
rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/00power_3 --strip-all -O binary  ta
rget/riscv64gc-unknown-none-elf/release/00power_3.bin;  rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unkn
own-none-elf/release/01power_5 --strip-all -O binary  target/riscv64gc-unknown-none-elf/release/01power_5.bin;  rust-obj
copy --binary-architecture=riscv64 target/riscv64gc-unknown-none-elf/release/02power_7 --strip-all -O binary  target/ris
cv64gc-unknown-none-elf/release/02power_7.bin;  rust-objcopy --binary-architecture=riscv64 target/riscv64gc-unknown-none
-elf/release/03sleep --strip-all -O binary  target/riscv64gc-unknown-none-elf/release/03sleep.bin;
make[1]: Leaving directory '/mnt/user'
   Compiling os v0.1.0 (/mnt/os)
    Finished release [optimized] target(s) in 4.43s
[rustsbi] RustSBI version 0.2.0-alpha.6
.______       __    __      _______.___________.  _______..______   __
|   _  \     |  |  |  |    /       |           | /       ||   _  \ |  |
|  |_)  |    |  |  |  |   |   (----`---|  |----`|   (----`|  |_)  ||  |
|      /     |  |  |  |    \   \       |  |      \   \    |   _  < |  |
|  |\  \----.|  `--'  |.----)   |      |  |  .----)   |   |  |_)  ||  |
| _| `._____| \______/ |_______/       |__|  |_______/    |______/ |__|

[rustsbi] Implementation: RustSBI-QEMU Version 0.0.2
[rustsbi-dtb] Hart count: cluster0 with 1 cores
[rustsbi] misa: RV64ACDFIMSU
[rustsbi] mideleg: ssoft, stimer, sext (0x222)
[rustsbi] medeleg: ima, ia, bkpt, la, sa, uecall, ipage, lpage, spage (0xb1ab)
[rustsbi] pmp0: 0x10000000 ..= 0x10001fff (rwx)
[rustsbi] pmp1: 0x80000000 ..= 0x8fffffff (rwx)
[rustsbi] pmp2: 0x0 ..= 0xffffffffffffff (---)
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
[kernel] Hello, world!
last 607 Physical Frames.
.text [0x80200000, 0x8020b000)
.rodata [0x8020b000, 0x8020f000)
.data [0x8020f000, 0x80290000)
.bss [0x80290000, 0x805a1000)
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section
```



```
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 [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 [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 [14000090000/300000]
power_3 [100000/300000]
power_3 [110000/300000]
power_3 [120000/300000]
power_3 [130000/300000]
power_3 [140000/300000]
```



```
power_3 [200000/300000]
power_3 [210000/300000]
110000/210000]
power_5 [120000/210000]
power_5 [130000/210000]
power_5 [140000/210000]
power_5 [150000/210000]
power_5 [160000/210000]
power_5 [170000/210000]
power_5 [180000/210000]
power_5 [190000/210000]
power_5 [200000/210000]
power_5 [210000/210000]
5^210000 = 527227302(MOD 998244353)
Test power_5 OK!
[kernel] Application exited with code 0
power_3 [220000/300000]
power_3 [230000/300000]
power_3 [240000/300000]
power_3 [250000/300000]
power_3 [260000/300000]
power_3 [270000/300000]
power_3 [280000/300000]
power_3 [290000/300000]
power_3 [300000/300000]
3^300000 = 612461288(MOD 998244353)
Test power_3 OK!
[kernel] Application exited with code 0
/240000]
power_7 [150000/240000]
power_7 [160000/240000]
power_7 [170000/240000]
power_7 [180000/240000]
power_7 [190000/240000]
power_7 [200000/240000]
power_7 [210000/240000]
power_7 [220000/240000]
power_7 [230000/240000]
power_7 [240000/240000]
7^240000 = 304164893(MOD 998244353)
Test power_7 OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
[Kernel] Panicked at src/task/mod.rs:115 All applications completed!
[root@3d686a71b33d os]#
```

二、思考问题

2.1 分析虚拟地址和物理地址的设计与实现；

- 虚拟地址和物理地址的数据结构的基本定义在os/src/mm/address.rs。
- 在os/src/mm/address.rs中实现和usize之间的相互转换、地址和页号之间的相互转换。
- 在os/src/mm/page_table.rs中通过map方法建立虚实地址之间的映射关系，通过unmap方式实现了拆除映射关系。

2.2 分析物理帧是如何管理与分配的；

- 在os/src/mm/frame_allocator.rs中，FrameTracker记录了物理帧的信息，包括页号，实现了FrameAllocator，用于分配和回收物理帧(alloc和dealloc)。
- 利用StackFrameAllocator类型的FrameAllocatorImpl实现栈管理物理页。实现frame_alloc和frame_dealloc分配和回收物理帧。
- 物理帧的分配先通过recycled检查回收的物理帧，优先从已经回收的列表中取出，如果没有回收的物理帧， 就将当前位置设为新的物理帧。
- 物理帧的回收，首先检测物理帧是否合法，并放入recycled中。

2.3 分析内核的地址空间以及应用程序的地址空间是如何实现的；

- 实现在os/src/mm/memory_set子模块
- 定义MapArea结构体，存储连续空间的虚拟页的范围、映射方式、映射权限。
- 实现memory_set方法管理操作体统的地址空间。
- 内核地址空间

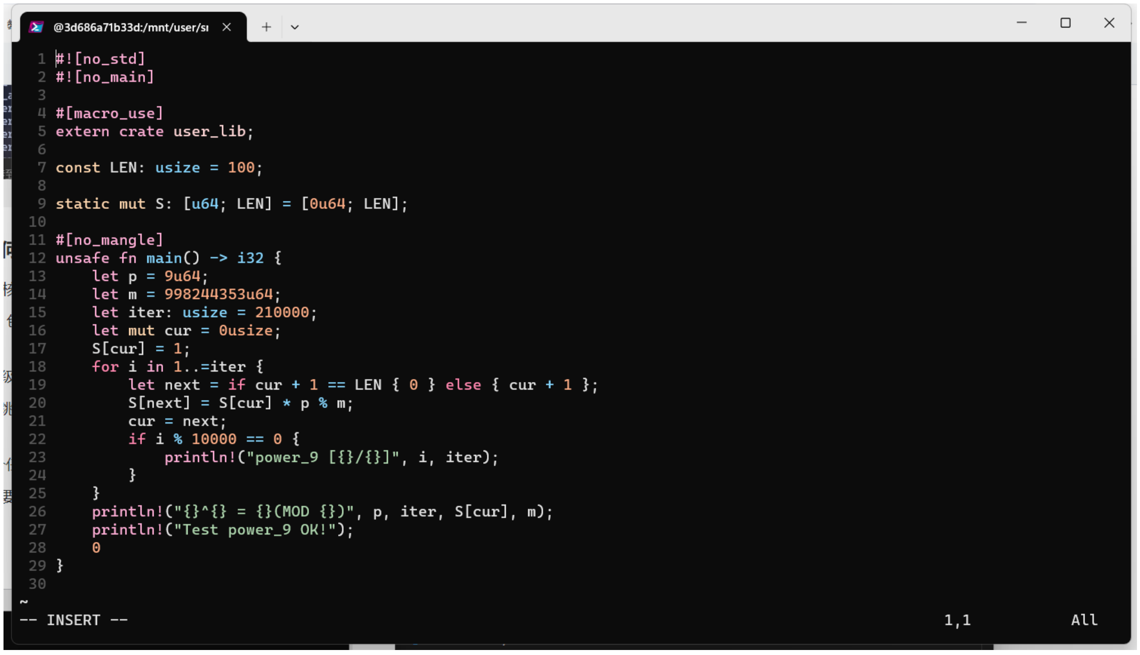
    - 实现new_kernel方法，用于创建内核地址空间。
    - 定义一个memory_set用于管理内核地址空间。
    - 将多个MapArea 放进memory_set，实现内核地址空间的分配和管理。
- 应用地址空间
    - user/src/linker.ld 中定义定义应用程序在内存中的布局，基础地址BASE_ADDRESS设为0。
    - 在MemorySet中定义from_elf方法，用于读取elf格式的可执行文件。解析页的大小和起始终止位置，放入map_area中。将map_area放入memory_set中管理。

2.4 分析基于地址空间的分时多任务是如何实现的；

- 在os/src/mm/memory_set.rs中创建统一的内核地址空间。
- 在rust_main中进行内存管理子系统的初始化，包括heap_allocator和frame_allocator并激活内核空间。
- 实现remap_test方法，检查内核地址空间的多级页表设置
- 扩展Trap上下文，实现地址空间的切换，建立跳板页面。在调整内存布局的时候将trap对齐到代码段的一个页面中。
- 通过 TaskControlBlock 任务管理器来管理每个任务的信息，并实行上下文切换。
- 实现TaskManager，在内核初始化的时候，需要将所有的应用程序加载到全局应用管理器中。

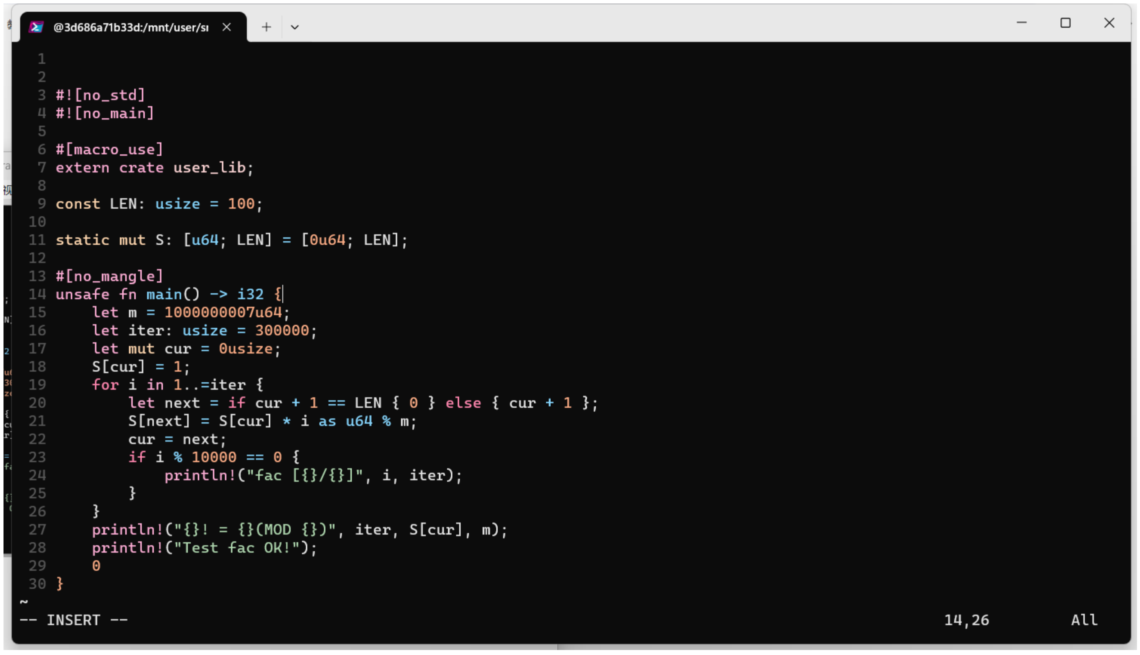
2.5 编写新的应用程序并测试验证结果。

- 实现了04power_9.rs，计算9的210000次方mod998244353



```
#![no_std]
#![no_main]

#[macro_use]
extern crate user_lib;

const LEN: usize = 100;

static mut S: [u64; LEN] = [0u64; LEN];

#[no_mangle]
unsafe fn main() -> i32 {
    let p = 9u64;
    let m = 998244353u64;
    let iter: usize = 210000;
    let mut cur = 0usize;
    S[cur] = 1;
    for i in 1..=iter {
        let next = if cur + 1 == LEN { 0 } else { cur + 1 };
        S[next] = S[cur] * p % m;
        cur = next;
        if i % 10000 == 0 {
            println!("power_9 [{}/{}]", i, iter);
        }
    }
    println!("{}^{} = {}(MOD {})", p, iter, S[cur], m);
    println!("Test power_9 OK!");
    0
}

~
-- INSERT --                                                  1,1           All
```

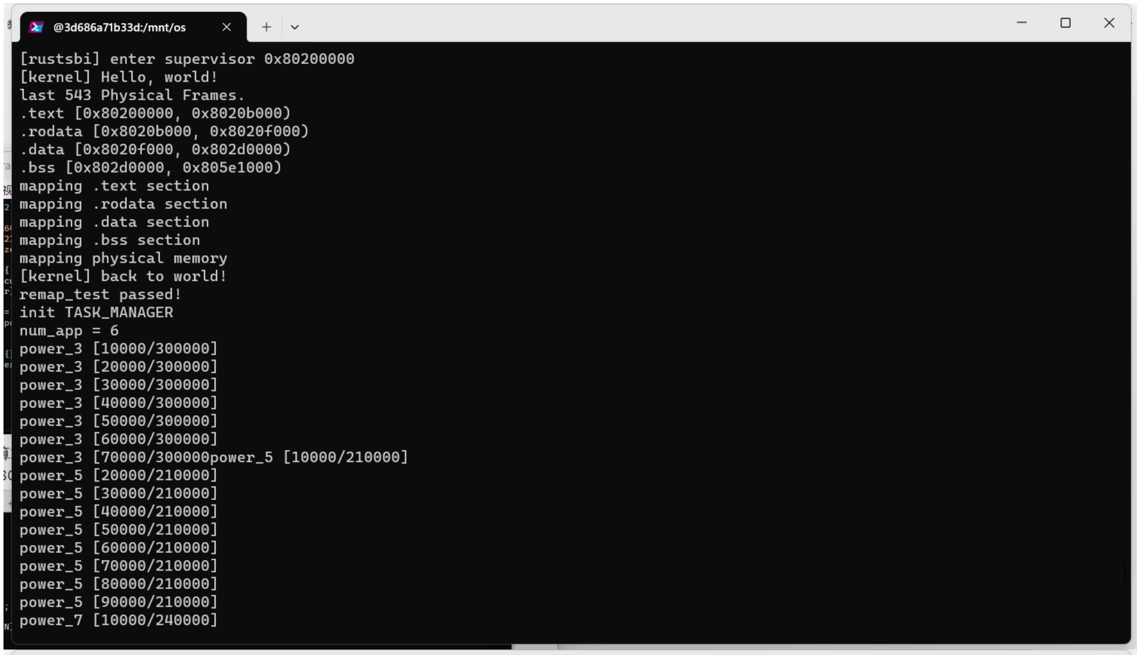
- 实现了05_fac.rs，计算300000的阶层



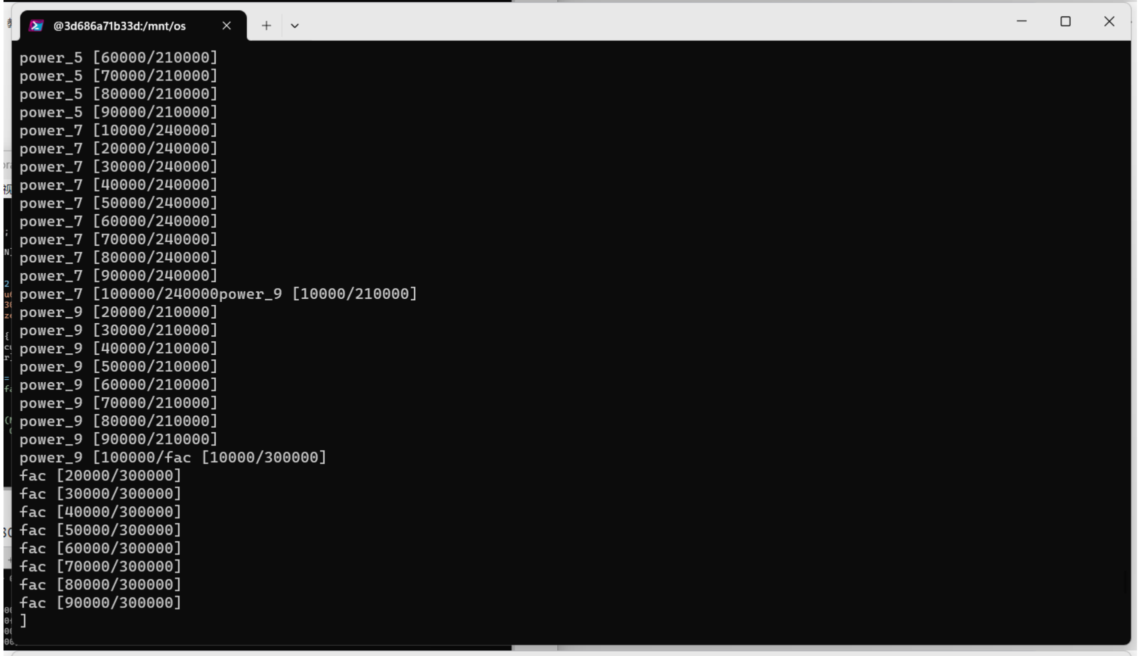
@3d686a71b33d:/mnt/user/s

```



#![no_std]
#![no_main]

#[macro_use]
extern crate user_lib;

const LEN: usize = 100;

static mut S: [u64; LEN] = [0u64; LEN];

#[no_mangle]
unsafe fn main() -> i32 {
    let m = 1000000007u64;
    let iter: usize = 300000;
    let mut cur = 0usize;
    S[cur] = 1;
    for i in 1..=iter {
        let next = if cur + 1 == LEN { 0 } else { cur + 1 };
        S[next] = S[cur] * i as u64 % m;
        cur = next;
        if i % 10000 == 0 {
            println!("fac [{}/{}]", i, iter);
        }
    }
    println!("{}! = {}(MOD {})", iter, S[cur], m);
    println!("Test fac OK!");
    0
}
~
-- INSERT --                                                  14,26         All
```

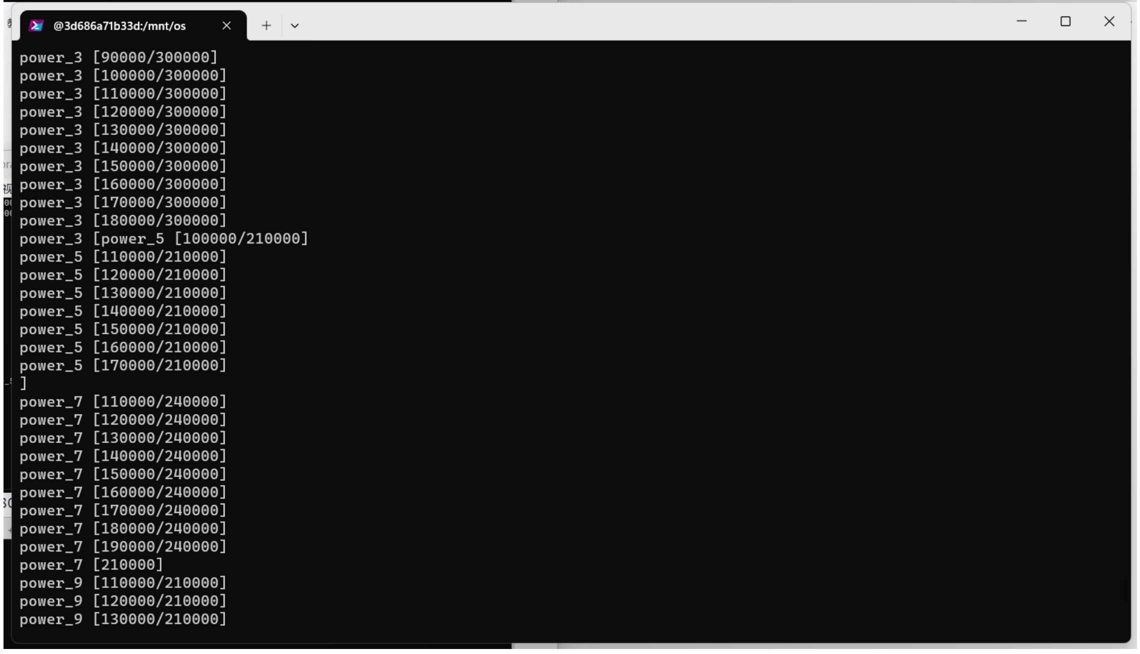
- 运行结果



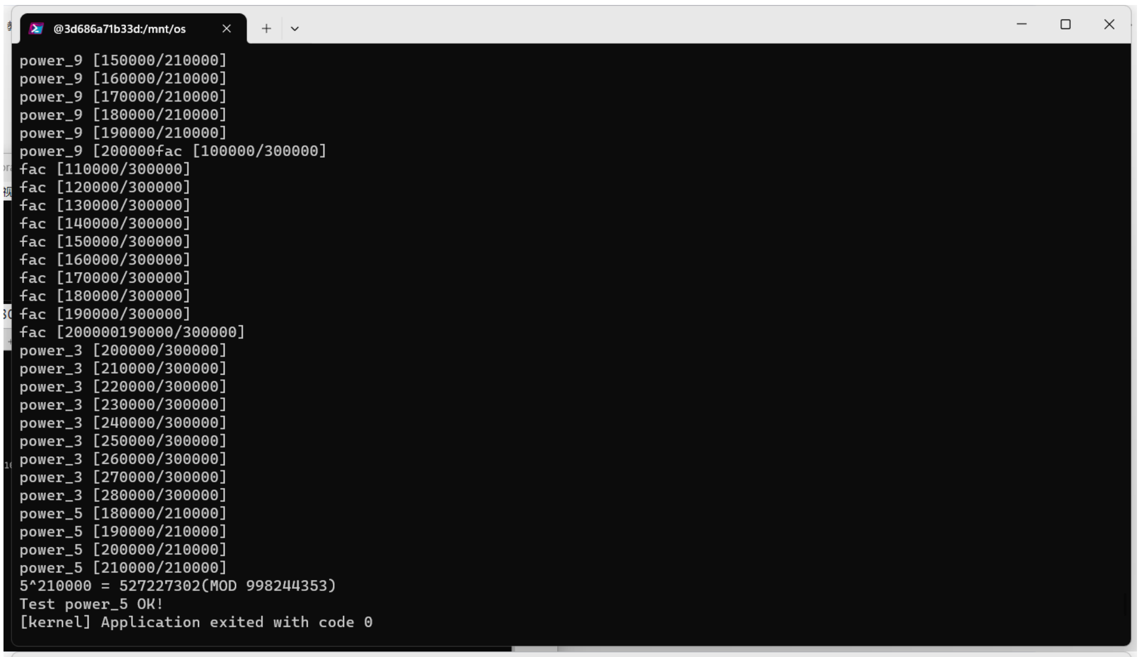
```
[rustsbi] enter supervisor 0x80200000
[kernel] Hello, world!
last 543 Physical Frames.
.text [0x80200000, 0x8020b000)
.rodata [0x8020b000, 0x8020f000)
.data [0x8020f000, 0x802d0000)
.bss [0x802d0000, 0x805e1000)
mapping .text section
mapping .rodata section
mapping .data section
mapping .bss section
mapping physical memory
[kernel] back to world!
remap_test passed!
init TASK_MANAGER
num_app = 6
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 [70000/300000power_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_7 [10000/240000]
```



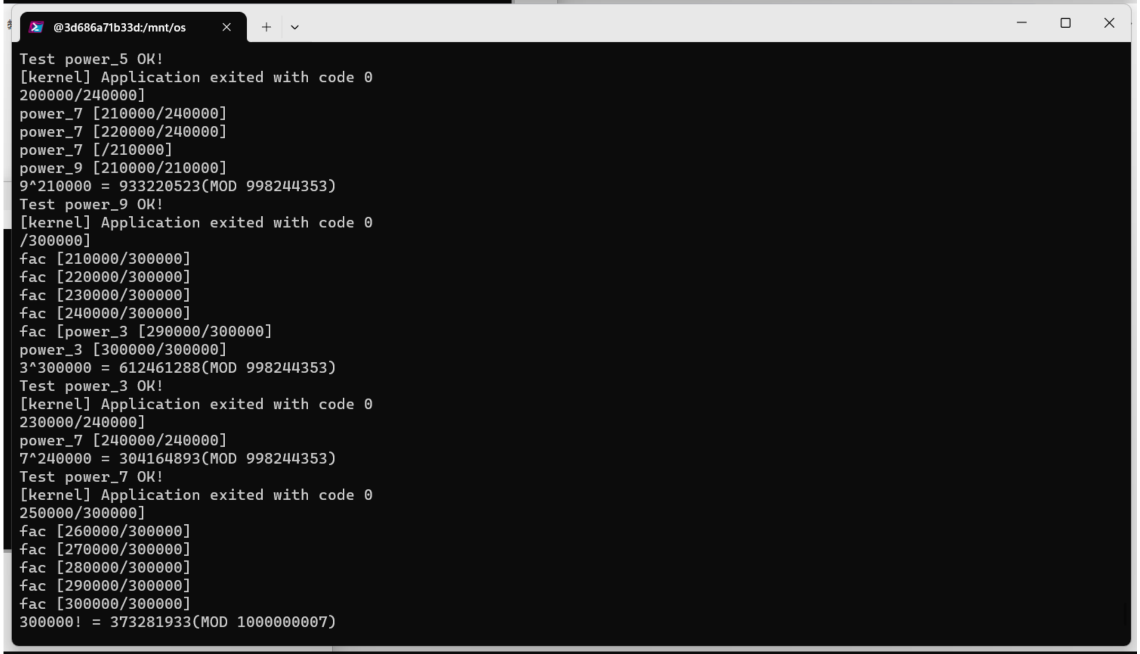
```
power_5 [60000/210000]
power_5 [70000/210000]
power_5 [80000/210000]
power_5 [90000/210000]
power_7 [10000/240000]
power_7 [20000/240000]
power_7 [30000/240000]
power_7 [40000/240000]
power_7 [50000/240000]
power_7 [60000/240000]
power_7 [70000/240000]
power_7 [80000/240000]
power_7 [90000/240000]
power_7 [100000/240000power_9 [10000/210000]
power_9 [20000/210000]
power_9 [30000/210000]
power_9 [40000/210000]
power_9 [50000/210000]
power_9 [60000/210000]
power_9 [70000/210000]
power_9 [80000/210000]
power_9 [90000/210000]
power_9 [100000/fac [10000/300000]
fac [20000/300000]
fac [30000/300000]
fac [40000/300000]
fac [50000/300000]
fac [60000/300000]
fac [70000/300000]
fac [80000/300000]
fac [90000/300000]
]
```



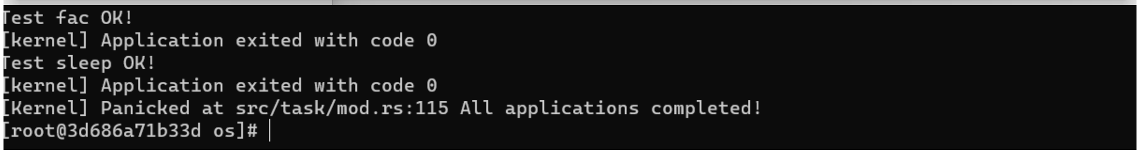
```
power_3 [90000/300000]
power_3 [100000/300000]
power_3 [110000/300000]
power_3 [120000/300000]
power_3 [130000/300000]
power_3 [140000/300000]
power_3 [150000/300000]
power_3 [160000/300000]
power_3 [170000/300000]
power_3 [180000/300000]
power_3 [power_5 [100000/210000]
power_5 [110000/210000]
power_5 [120000/210000]
power_5 [130000/210000]
power_5 [140000/210000]
power_5 [150000/210000]
power_5 [160000/210000]
power_5 [170000/210000]
]
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_7 [180000/240000]
power_7 [190000/240000]
power_7 [210000]
power_9 [110000/210000]
power_9 [120000/210000]
power_9 [130000/210000]
```



```
power_9 [150000/210000]
power_9 [160000/210000]
power_9 [170000/210000]
power_9 [180000/210000]
power_9 [190000/210000]
power_9 [200000fac [100000/300000]
fac [110000/300000]
fac [120000/300000]
fac [130000/300000]
fac [140000/300000]
fac [150000/300000]
fac [160000/300000]
fac [170000/300000]
fac [180000/300000]
fac [190000/300000]
fac [200000190000/300000]
power_3 [200000/300000]
power_3 [210000/300000]
power_3 [220000/300000]
power_3 [230000/300000]
power_3 [240000/300000]
power_3 [250000/300000]
power_3 [260000/300000]
power_3 [270000/300000]
power_3 [280000/300000]
power_5 [180000/210000]
power_5 [190000/210000]
power_5 [200000/210000]
power_5 [210000/210000]
5^210000 = 527227302(MOD 998244353)
Test power_5 OK!
[kernel] Application exited with code 0
```



```
Test power_5 OK!
[kernel] Application exited with code 0
200000/240000]
power_7 [210000/240000]
power_7 [220000/240000]
power_7 [/210000]
power_9 [210000/210000]
9^210000 = 933220523(MOD 998244353)
Test power_9 OK!
[kernel] Application exited with code 0
/300000]
fac [210000/300000]
fac [220000/300000]
fac [230000/300000]
fac [240000/300000]
fac [power_3 [290000/300000]
power_3 [300000/300000]
3^300000 = 612461288(MOD 998244353)
Test power_3 OK!
[kernel] Application exited with code 0
230000/240000]
power_7 [240000/240000]
7^240000 = 304164893(MOD 998244353)
Test power_7 OK!
[kernel] Application exited with code 0
250000/300000]
fac [260000/300000]
fac [270000/300000]
fac [280000/300000]
fac [290000/300000]
fac [300000/300000]
300000! = 373281933(MOD 1000000007)
```

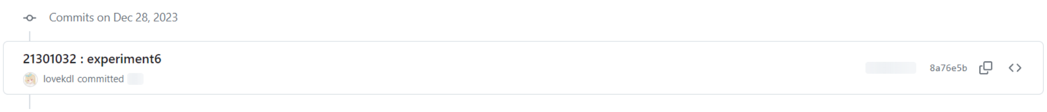


```
Test fac OK!
[kernel] Application exited with code 0
Test sleep OK!
[kernel] Application exited with code 0
[Kernel] Panicked at src/task/mod.rs:115 All applications completed!
[root@3d686a71b33d os]# |
```

- 验证成功

三、git截图

- git截图(https://github.com/lovekdl/GardenerOS)



Commits on Dec 28, 2023

21301032 : experiment6
lovekdl committed

8a76e5b

四、 其它说明

- 中途容器id更换是因为重装了docker，之前的容器丢失了。所以重新导入镜像，创建了新的容器。