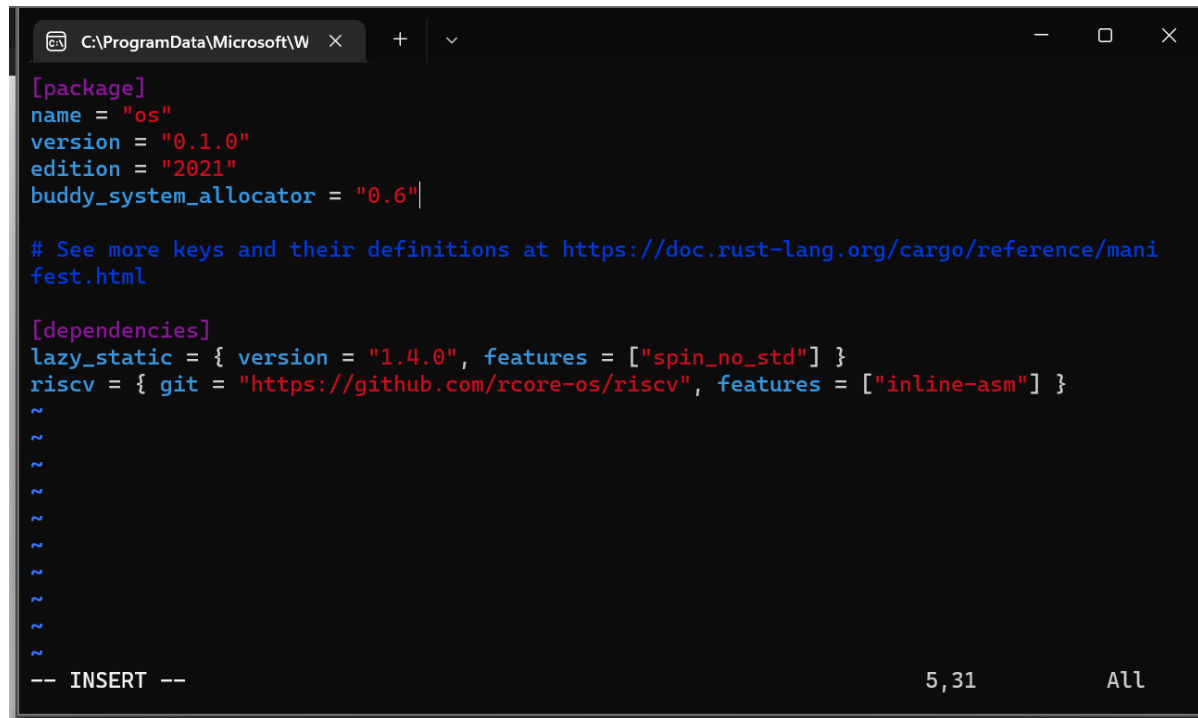


一. 实验步骤

1. 在内核中支持动态内存分配

- 增加依赖

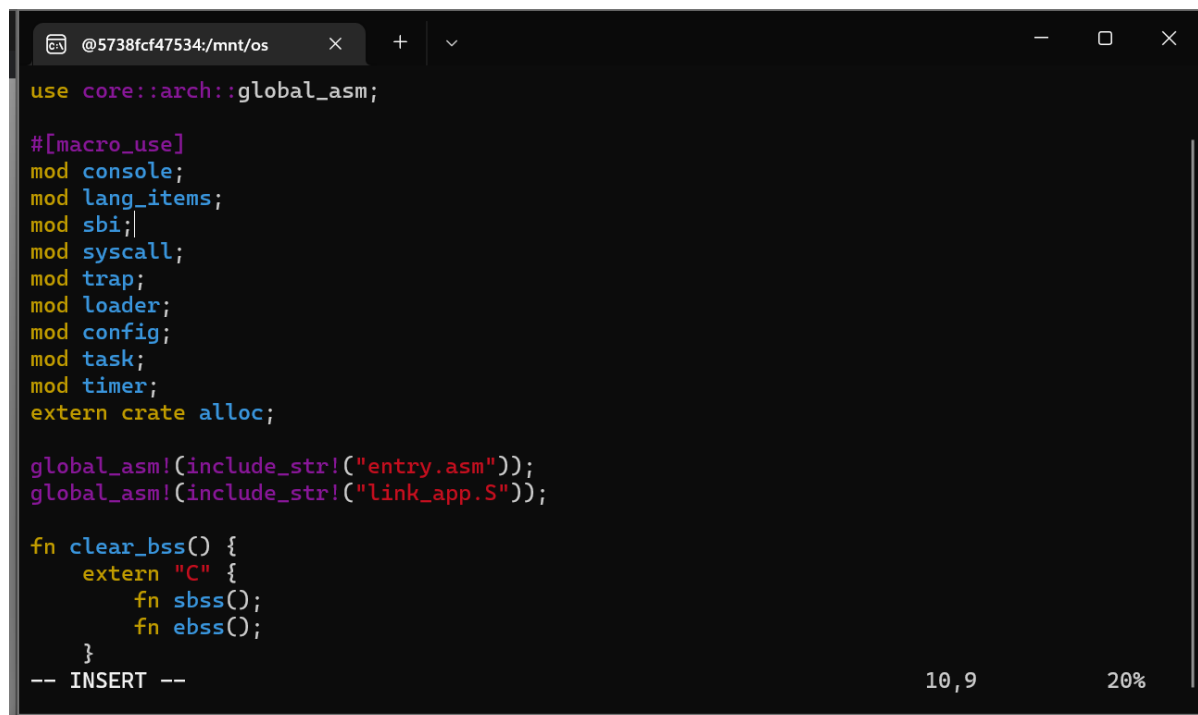


```
[package]
name = "os"
version = "0.1.0"
edition = "2021"
buddy_system_allocator = "0.6"

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

[dependencies]
lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

- 在 main.rs 中引入 alloc 依赖



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

#[macro_use]
mod console;
mod lang_items;
mod sbi;
mod syscall;
mod trap;
mod loader;
mod config;
mod task;
mod timer;
extern crate alloc;

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

fn clear_bss() {
    extern "C" {
        fn sbss();
        fn ebss();
    }
}
-- INSERT --
```

- 实现全局动态内存分配器

```
@5738fcf47534:/mnt/os x + v
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);
    }
}
~
~
~
~
~
~
-- INSERT -- 1,1 All
```

- 处理动态内存分配失败的情况

```
@5738fcf47534:/mnt/os x + v
#![no_std]
#![no_main]
#![feature(panic_info_message)]
#![feature(alloc_error_handler)]

use core::arch::global_asm;

#[macro_use]
mod console;
mod lang_items;
mod sbi;
mod syscall;
mod trap;
mod loader;
mod config;
mod task;
mod timer;
extern crate alloc;

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

-- INSERT -- 5,1 Top
```

```
@5738fcf47534:/mnt/os x + v
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);
    }
}

#[alloc_error_handler]
pub fn handle_alloc_error(layout: core::alloc::Layout) -> ! {
    panic!("Heap allocation error, layout = {:?}", layout);
}

~
~
-- INSERT -- 20,2 All
```

- 测试动态内存分配

```
@5738fcf47534:/mnt/os x + v
    panic!("Heap allocation error, layout = {:?}", layout);
}

#[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);
    }
}

-- INSERT -- 26,2 78%
```

- 修改 main.rs config.rs 中代码


```
@5738fcf47534:/mnt/os x + v
mod heap_allocator;

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

-- INSERT --
6,2 All
```

2. 实现虚拟地址与物理地址的基本定义

总之复制很多代码到 `os/src/mm/address.rs` 中，实现最基本的数据结构和相关操作

```
@5738fcf47534:/mnt/os x + v
    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 --
191,46 Bot
```

3. 定义页表项数据结构

- 实现标志位 `PTEFlags`

```
@5738fcf47534:/mnt/os x + v
use core::arch::global_asm;

#[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;

extern crate alloc;
extern crate bitflags;

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

fn clear_bss() {
```

22,22 19%

```
@5738fcf47534:/mnt/os x + v
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;
    }
}

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

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

-- INSERT --
```

14,2 Top

- 增加依赖

```
@5738fcf47534:/mnt/os x + v
[package]
name = "os"
version = "0.1.0"
edition = "2021"
buddy_system_allocator = "0.6"
bitflags = "1.2.1"

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

[dependencies]
lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
~
~
~
~
~
~
~
~
-- INSERT -- 6,19 All
```

- 实现 PageTableEntry

```
@5738fcf47534:/mnt/os x + v
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()
    }
}

1,1 Top
```

- 修改 os/src/mm/mod.rs

```
@5738fcf47534:/mnt/os x + v
mod heap_allocator;
mod page_table;

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

pub fn init() {
    heap_allocator::init_heap();
    heap_allocator::heap_test();
}
~
~
~
~
~
~
~
~
~
~
~
-- INSERT (paste) -- 5,38 All
```

4. 实现物理帧的管理与分配

- 设置物理内存终止地址

```
@5738fcf47534:/mnt/os x + v
pub const USER_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
pub const MAX_APP_NUM: usize = 4;
pub const APP_BASE_ADDRESS: usize = 0x80400000;
pub const APP_SIZE_LIMIT: usize = 0x20000;
pub const CLOCK_FREQ: usize = 12500000;
pub const KERNEL_HEAP_SIZE: usize = 0x30_0000;
pub const MEMORY_END: usize = 0x80800000;
~
~
~
~
~
~
~
~
~
~
~
-- INSERT -- 8,1 All
```

- 实现物理帧管理


```
@5738fcf47534:/mnt/os × + v
use super::{PhysAddr, PhysPageNum};
use alloc::vec::Vec;
use crate::sync::UPSafeCell;
use crate::config::MEMORY_END;
use lazy_static::*;
use core::fmt::{self, Debug, Formatter};

pub struct FrameTracker {
    pub ppn: PhysPageNum,
}

impl FrameTracker {
    pub fn new(ppn: PhysPageNum) -> Self {
        // page cleaning
        let bytes_array = ppn.get_bytes_array();
        for i in bytes_array {
            *i = 0;
        }
        Self { ppn }
    }
}

1,1 Top
```

- 增加 sync 模块, 实现 UPSafeCell


```
mod syscall;
mod trap;
mod loader;
mod config;
mod task;
mod timer;
mod mm;
mod sync;

extern crate alloc;
extern crate 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;
    }
}
```

- 物理帧管理测试

```
pub const USER_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
pub const MAX_APP_NUM: usize = 4;
pub const APP_BASE_ADDRESS: usize = 0x80400000;
pub const APP_SIZE_LIMIT: usize = 0x20000;
pub const CLOCK_FREQ: usize = 12500000;
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;|

-- INSERT --
```

```
@5738fcf47534:/mnt/os x + v
mod heap_allocator;
mod page_table;
mod address;
mod frame_allocator;

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

pub fn init() {
    heap_allocator::init_heap();
    heap_allocator::heap_test();
    frame_allocator::init_frame_allocator();
    frame_allocator::frame_allocator_test();
}
~
~
~
~
~
"src/mm/mod.rs" 17L, 467B 10,37 All
```

- 测试成功

```
@5738fcf47534:/mnt/os x + v
qemu-system-riscv64: clint: invalid write: 00000004
[rustsbi] enter supervisor 0x80200000
[kernel] Hello, world!
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!
power_3 [10000/200000]
power_3 [20000power_5 [10000/200000]
power_5 [20000/200000]
power_5 [30000/200000]
power_5 [40000/200000]
power_5 [50000/200000]
power_5 [60000/200000]
```

5. 多级页表管理

依然是粘贴各种代码，实现基本数据结构和操作

```
@5738fcf47534:/mnt/os x + v
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
}
}
:wq|
```

6. 内核与应用的地址空间

- 内核地址空间

粘贴粘贴粘贴粘贴 主要实现了地址空间的抽象和创建内核地址空间

```
@5738fcf47534:/mnt/os x + v
(sdata as usize).into(),
(edata as usize).into(),
MapType::Identical,
MapPermission::R | MapPermission::W,
), None);
println!("mapping .bss section");
memory_set.push(MapArea::new(
  (sbss_with_stack as usize).into(),
  (ebss as usize).into(),
  MapType::Identical,
  MapPermission::R | MapPermission::W,
), None);
println!("mapping physical memory");
memory_set.push(MapArea::new(
  (kernel as usize).into(),
  MEMORY_END.into(),
  MapType::Identical,
  MapPermission::R | MapPermission::W,
), None);
memory_set
}
}
-- INSERT -- 211,2 Bot
```

- 修改链接脚本

这里不知道是实验手册有问题还是我看漏了，选中的这行 `sbss_with_stack` 手册里好像是没有的，如果没有这一行最后build会失败

```
@5738fcf47534:/mnt/os  x  +  v

    *(.data .data.*)
    *(.sdata .sdata.*)
}

. = ALIGN(4K);
edata = .;
sbss_with_stack = .;
.bss : {
    *(.bss.stack)
    sbss = .;
    *(.bss .bss.*)
    *(.sbss .sbss.*)
}

. = ALIGN(4K);
ebss = .;
ekernel = .;

/DISCARD/ : {
    *(.eh_frame)
}
}
-- VISUAL --                2          37,15          Bot
```

- 修改 loader 子模块

```
@5738fcf47534:/mnt  x  +  v

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]
        )
    }
}

~
~
20,1          All
```

- 实现解析ELF格式的数据

```
@5738fcf47534:/mnt
// map user stack with U flags
let max_end_va: VirtAddr = max_end_vpn.into();
let mut user_stack_bottom: usize = max_end_va.into();
// guard page
user_stack_bottom += PAGE_SIZE;
let user_stack_top = user_stack_bottom + USER_STACK_SIZE;
memory_set.push(MapArea::new(
    user_stack_bottom.into(),
    user_stack_top.into(),
    MapType::Framed,
    MapPermission::R | MapPermission::W | MapPermission::U,
), None);
// map TrapContext
memory_set.push(MapArea::new(
    TRAP_CONTEXT.into(),
    TRAMPOLINE.into(),
    MapType::Framed,
    MapPermission::R | MapPermission::W,
), None);
(memory_set, user_stack_top, elf.header.pt2.entry_point() as usize)
}
}
-- INSERT --
222,2 Bot
```

- 增加 xmas-elf 依赖

```
@5738fcf47534:/mnt
[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]
lazy_static = { version = "1.4.0", features = ["spin_no_std"] }
riscv = { git = "https://github.com/rcore-os/riscv", features = ["inline-asm"] }
buddy_system_allocator = "0.6"
bitflags = "1.2.1"
xmas-elf = "0.7.0"
~
~
~
~
~
~
~
-- INSERT --
13,19 All
```

- 实现 memory_set 子模块

```
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
};

pub struct MapArea {
    vpn_range: VPNRange,
    data_frames: BTreeMap<VirtPageNum, FrameTracker>,
}

-- INSERT --
```

- 修改 config.rs

A screenshot of a terminal window with a dark background. The title bar at the top shows '@5738fcf47534:/mnt' and standard window controls. The terminal displays several C preprocessor-style constant definitions. Each line starts with 'pub const' in blue, followed by the variable name in green, its type 'usize' in red, and its value in white. The variables are USER_STACK_SIZE, KERNEL_STACK_SIZE, MAX_APP_NUM, APP_BASE_ADDRESS, APP_SIZE_LIMIT, CLOCK_FREQ, KERNEL_HEAP_SIZE, MEMORY_END, PAGE_SIZE, PAGE_SIZE_BITS, TRAMPOLINE, and TRAP_CONTEXT. The values include arithmetic expressions like '4096 * 2' and hexadecimal literals like '0x80400000'. At the bottom left, there's a prompt '-- INSERT --'. At the bottom right, the page number '12,56' and the word 'All' are visible. On the far left edge, there's a vertical column of small blue tilde symbols (~).

```
pub const USER_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
pub const MAX_APP_NUM: usize = 4;
pub const APP_BASE_ADDRESS: usize = 0x80400000;
pub const APP_SIZE_LIMIT: usize = 0x20000;
pub const CLOCK_FREQ: usize = 12500000;
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;
```

-- INSERT --

12,56 All

7. 实现基于地址空间的分时多任务

- 创建内核地址空间


```
@5738fcf47534:/mnt
memory_set.push(MapArea::new(
    user_stack_bottom.into(),
    user_stack_top.into(),
    MapType::Framed,
    MapPermission::R | MapPermission::W | MapPermission::U,
), None);
// map TrapContext
memory_set.push(MapArea::new(
    TRAP_CONTEXT.into(),
    TRAMPOLINE.into(),
    MapType::Framed,
    MapPermission::R | MapPermission::W,
), None);
(memory_set, user_stack_top, elf.header.pt2.entry_point() as usize)
}
}

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

-- INSERT --                                     200,2      Bot
```

- 进行内存管理子系统的初始化

```
@5738fcf47534:/mnt/os
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();
}

~
~
~

"src/mm/mod.rs" [noeol] 19L, 581B                                     18,1      All
```

- 检查内核地址空间的多级页表设置

```

}

#[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 --
221,2 Bot

```

- 实现跳板机制
- 扩展 Trap 上下文

```

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);
    cx
}
}
-- INSERT --
35,2 Bot

```

- 实现地址空间的切换

```
@5738fcf47534:/mnt x + v
# switch to user space
csrw satp, a1
sfence.vma
csrw sscratch, a0
mv sp, a0
# now sp points to TrapContext in user space, start restoring based on it
# restore sstatus/sepc
ld t0, 32*8(sp)
ld t1, 33*8(sp)
csrw sstatus, t0
csrw sepc, t1
# restore general purpose registers except x0/sp/tp
ld x1, 1*8(sp)
ld x3, 3*8(sp)
.set n, 5
.rept 27
    LOAD_GP %n
    .set n, n+1
.endr
# back to user stack
ld sp, 2*8(sp)
sret

69,8 Bot
```

- 建立跳板页面

```
@5738fcf47534:/mnt x + v
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 = .;
-- INSERT (paste) --

17,25 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

```
@5738fcf47534:/mnt/os
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,
}

"src/task/mod.rs" 152L, 4239B 7,1 Top
```

- 更新 config.rs , 在内核初始化时加载所有应用程序

```
@5738fcf47534:/mnt
pub const USER_STACK_SIZE: usize = 4096 * 2;
pub const KERNEL_STACK_SIZE: usize = 4096 * 2;
pub const MAX_APP_NUM: usize = 4;
pub const APP_BASE_ADDRESS: usize = 0x80400000;
pub const APP_SIZE_LIMIT: usize = 0x20000;
pub const CLOCK_FREQ: usize = 12500000;
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;

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)
}

-- INSERT -- 18,2 All
```

- 修改 TaskManager 的实现

```
@5738fcf47534:/mnt x + v
};
}

impl TaskManager {
    fn run_first_task(&self) -> ! {
        let mut inner = self.inner.exclusive_access();
        let next_task = &mut inner.tasks[0];
        next_task.task_status = TaskStatus::Running;
        let next_task_cx_ptr = &next_task.task_cx as *const TaskContext;
        drop(inner);
        let mut _unused = TaskContext::zero_init();
        // before this, we should drop local variables that must be dropped manually
        unsafe {
            __switch(
                &mut _unused as *mut _,
                next_task_cx_ptr,
            );
        }
        panic!("unreachable in run_first_task!");
    }

    fn mark_current_suspended(&self) {

```

55,1 30%

- 修改 /os/src/task/switch.S

```
@5738fcf47534:/mnt x + v
# 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
# restore kernel stack of next task
ld sp, 8(a1)
ret

33,7 Bot
```

- 修改 switch.rs

```
@5738fcf47534:/mnt
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
    );
}

12,1 All
```

- 改进 Trap 的处理

```
@5738fcf47534:/mnt

let user_satp = current_user_token();
extern "C" {
    fn __alltraps();
    fn __restore();
}
let restore_va = __restore as usize - __alltraps as usize + TRAMPOLINE;
unsafe {
    asm!(
        "fence.i",
        "jr {restore_va}",
        restore_va = in(reg) restore_va,
        in("a0") trap_cx_ptr,
        in("a1") user_satp,
        options(noreturn)
    );
}

#[no_mangle]
pub fn trap_from_kernel() -> ! {
    panic!("a trap from kernel!");
}

83,1 Bot
```

- 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

```
@5738fcf47534:/mnt x + v
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],
        }
    }
}

1,1 Top
```

- 改进 sys_write 的实现

```
@5738fcf47534:/mnt x + v
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
}

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();
        start = start + 8;
    }
}

-- INSERT --

142,2 91%
```

- 修改 sys_write 系统调用

```
@5738fcf47534:/mnt x + v
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!");
        }
    }
}

"os/src/syscall/fs.rs" 19L, 533B 19,1 All
```

8. 修改应用程序

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

除了删除 `clear_bss()` 的实现外，注意删除 `_start()` 中调用的 `clear_bss()`。

```
@5738fcf47534:/mnt x + v
use syscall::*;

pub fn write(fd: usize, buf: &[u8]) -> isize { sys_write(fd, buf) }
pub fn exit(exit_code: i32) -> isize { sys_exit(exit_code) }
pub fn get_time() -> isize { sys_get_time() }

fn clear_bss() {
    extern "C" {
        fn start_bss();
        fn end_bss();
    }
    (start_bss as usize..end_bss as usize).for_each(|addr| {
        unsafe { (addr as *mut u8).write_volatile(0); }
    });
}

#[no_mangle]
#[link_section = ".text.entry"]
pub extern "C" fn _start() -> ! {
    clear_bss();
    exit(main());
}

-- VISUAL -- 11 16,0-1 47%
```



```
@5738fcf47534:/mnt x + v

#[macro_use]
pub mod console;
mod syscall;
mod lang_items;

use syscall::*;

pub fn write(fd: usize, buf: &[u8]) -> isize { sys_write(fd, buf) }
pub fn exit(exit_code: i32) -> isize { sys_exit(exit_code) }
pub fn get_time() -> isize { sys_get_time() }

#[no_mangle]
#[link_section = ".text.entry"]
pub extern "C" fn _start() -> ! {
    clear_bss();
    exit(main());
    panic!("unreachable after sys_exit!");
}

#[linkage = "weak"]
#[no_mangle]
-- VISUAL -- 2 18,34 37%
```

- 删除 build.py

```
@5738fcf47534:/mnt/user x + v

[root@5738fcf47534 mnt]# vim os/src/syscall/fs.rs
[root@5738fcf47534 mnt]# vim os/src/syscall/fs.rs
[root@5738fcf47534 mnt]# vim user/src/lib.rs
[root@5738fcf47534 mnt]# cd os
[root@5738fcf47534 os]# ls
Cargo.lock Cargo.toml Makefile build.rs src target
[root@5738fcf47534 os]# rm build.rs
rm: remove regular file 'build.rs'? y
[root@5738fcf47534 os]# ls
Cargo.lock Cargo.toml Makefile src target
[root@5738fcf47534 os]# vim Makefile
[root@5738fcf47534 os]# vim src/main.rs
[root@5738fcf47534 os]# vim src/build.rs
[root@5738fcf47534 os]# vim build.rs
[root@5738fcf47534 os]# vim build.rs
[root@5738fcf47534 os]# cd ../user
[root@5738fcf47534 user]# ls
Cargo.lock Cargo.toml Makefile build.py src target
[root@5738fcf47534 user]# rm build.py
rm: remove regular file 'build.py'? y
[root@5738fcf47534 user]# ls
Cargo.lock Cargo.toml Makefile src target
[root@5738fcf47534 user]# |
```

- 修改 Makefile 文件

把 user/Makefile 文件中的 build.py 替换为 cargo build

```
@5738fcf47534:/mnt/user X + v
TARGET := riscv64gc-unknown-none-elf
MODE := release
APP_DIR := src/bin
TARGET_DIR := target/${TARGET}/${MODE}
APPS := $(wildcard ${APP_DIR}/*.rs)
ELFS := $(patsubst ${APP_DIR}/%.rs, ${TARGET_DIR}/%, $(APPS))
BINS := $(patsubst ${APP_DIR}/%.rs, ${TARGET_DIR}/%.bin, $(APPS))

OBJDUMP := rust-objdump --arch-name=riscv64
OBJCOPY := rust-objcopy --binary-architecture=riscv64

elf: $(APPS)
    @cargo build --release

binary: elf
    $(foreach elf, $(ELFS), $(OBJCOPY) $(elf) --strip-all -O binary $(patsubst ${TARGET_DIR}/%, ${TARGET_DIR}/%.bin, $(elf));)

build: binary

clean:
    @cargo clean
"Makefile" 23L, 618B 1,1 Top
```

- 修改 main.rs

```
@5738fcf47534:/mnt/os X + v
    }
    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 build.rs

```
@5738fcf47534:/mnt/os x + v
.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_{}_start
.global app_{}_end
.align 3
app_{}_start:
.incbin "{}{1}"
app_{}_end:"#, idx, app, TARGET_PATH)?;
}
Ok(())
}
```

49,1 Bot

8. 运行成功

应该算成功了吧，虽然有很多警告，好像是因为删代码没删干净

```
@5738fcf47534:/mnt/os x + v
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_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]
```

```
@5738fcf47534:/mnt/os x + v
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_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
Panicked at src/task/mod.rs:115 All applications completed!
[root@5738fcf47534 os]#
```

二. 思考题

1. 虚拟地址和物理地址的设计与实现

- **设计:** 虚拟地址 (`VirtAddr`) 和物理地址 (`PhysAddr`) 是为了方便内存管理和实现虚拟内存系统而设计的。在操作系统中, 虚拟地址提供了一种抽象, 允许每个程序感觉像是在使用一个大的、连续的内存空间, 而实际上这些虚拟地址被映射到物理内存的不同部分。
- **实现:** 在您的代码中, 这两种地址都被封装为含有一个 `usize` 字段的结构体。提供了从 `usize` 到这些类型的转换, 以及这些类型之间的相互转换 (例如, 通过页号和页表来转换)。这种实现方式简化了地址的处理, 并允许在物理和虚拟地址空间之间轻松转换。

2. 物理帧的管理与分配

- **管理:** 物理帧 (内存的物理页) 通过 `StackFrameAllocator` 来管理。这个分配器负责跟踪哪些物理页被使用, 哪些是空闲的。
- **分配:** 通过 `frame_alloc` 函数分配物理帧。如果有可用的帧, 它会返回一个包含物理页号的 `FrameTracker` 实例。当 `FrameTracker` 被丢弃时, 对应的物理帧会被释放回分配器。

3. 内核和应用程序的地址空间实现

- **内核地址空间:** 通过 `MemorySet` 结构表示。在启动时, 内核地址空间包括代码段、数据段、BSS段等。此地址空间始终激活并被所有进程共享。
- **应用程序地址空间:** 每个应用程序都有自己的 `MemorySet` 实例, 表示其私有的地址空间。这些空间包括应用程序的代码、数据、堆栈等。在上下文切换时, 会切换活跃的地址空间, 确保每个应用程序只能访问自己的内存。

4. 基于地址空间的分时多任务实现

- **多任务:** 利用任务控制块 (`TaskControlBlock`), 系统维护了多个任务的状态。每个任务都有自己的虚拟地址空间 (`MemorySet`), 堆栈, 以及其他必要的上下文信息。

- **上下文切换:** 当一个任务的时间片用完时，系统会保存其上下文（包括CPU寄存器等），并加载下一个任务的上下文。这个过程通过保存和恢复任务上下文来完成，确保每个任务在适当的时候运行。

5. 编写新的应用程序并测试验证结果

- 实现了一个新的计算2次幂的程序

```
@5738fcf47534:/mnt/user  ×  +  ∨

#![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 = 2u64;
    let m = 998244353u64;
    let iter: usize = 240000;
    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 {
            "src/bin/04power_2.rs" [dos] 29L, 642B
            8,0-1 Top
        }
    }
}
```

- 运行结果:

看起来没什么问题

```
@5738fcf47534:/mnt/os  ×  +  ∨

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
210000/210000]
5^210000 = 527227302(MOD 998244353)
Test power_5 OK!
[kernel] Application exited with code 0
240000/power_2 [230000/240000]
power_2 [240000/240000]
2^240000 = 746486105(MOD 998244353)
Test power_2 OK!
[kernel] Application exited with code 0
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
Panicked at src/task/mod.rs:115 All applications completed!
[root@5738fcf47534 os]#
```

三. Git提交截图

