练习 0：填写已有实验

本实验依赖实验 2/3/4/5/6/7。请把你做的实验 2/3/4/5/6/7 的代码填入本实验中代码中有“LAB2”/“LAB3”/“LAB4”/“LAB5”/“LAB6” /“LAB7”的注释相应部分。并确保编译通过。注意：为了能够正确执行 lab8 的测试应用程序，可能需对已完成的实验 2/3/4/5/6/7 的代码进行进一步改进。

练习 1: 完成读文件操作的实现（需要编码）

首先了解打开文件的处理流程，然后参考本实验后续的文件读写操作的过程分析，填写在 kern/fs/sfs/sfs_inode.c 中 的 sfs_io_nolock()函数，实现读文件中数据的代码。

(1) 如果偏移量不与第一个块对齐，从偏移量到第一个块的末尾读取/写入一些内容
读/写大小 = (nblks != 0) ? (SFS_BLKSIZE - blkoff) : (endpos - offset)

(2) 读/写对齐的块

(3) 如果结束位置不与最后一个块对齐，从开头到最后一个块的 (endpos % SFS_BLKSIZE) 处读取/写入一些内容

```
//LAB8:EXERCISE1 YOUR CODE HINT: call sfs_bmap_load_nolock, sfs_rbuf, sfs_
  /*
   * (1) If offset isn't aligned with the first block, Rd/Wr some content
   *       NOTICE: useful function: sfs_bmap_load_nolock, sfs_buf_op
   *               Rd/Wr size = (nblks != 0) ? (SFS_BLKSIZE - blkoff) : (e
   * (2) Rd/Wr aligned blocks
   *       NOTICE: useful function: sfs_bmap_load_nolock, sfs_block_op
   * (3) If end position isn't aligned with the last block, Rd/Wr some cor
   *       NOTICE: useful function: sfs_bmap_load_nolock, sfs_buf_op
  */
  blkoff=offset%SFS_BLKSIZE;
  if (blkoff != 0) {
      // if(nblks != 0)size=SFS_BLKSIZE-blkoff;
      // else size=endpos-offset;
      size = (nblks != 0) ? (SFS_BLKSIZE - blkoff) : (endpos - offset);
      ret = sfs_bmap_load_nolock(sfs, sin, blkno, &ino);
      if (ret != 0) {
          goto out;
      }

      ret = sfs_buf_op(sfs, buf, size, ino, blkoff);
      if (ret != 0) {
          goto out;
      }
      alen += size;
      buf += size;
      if(nblks == 0)goto out;
      blkno++;
      nblks--;

  }
```

```
    if (nblks>0) {
        ret = sfs_bmap_load_nolock(sfs, sin, blkno, &ino);
        if (ret < 0) {
            goto out;
        }
        ret = sfs_block_op(sfs, buf, blkno, nblks);
        if (ret < 0) {
            goto out;
        }
        alen += nblks * SFS_BLKSIZE;
        buf += nblks * SFS_BLKSIZE;
        blkno += nblks;
        nblks = 0;
    }
    size = endpos % SFS_BLKSIZE;
    if (endpos % SFS_BLKSIZE!=0) {
        ret = sfs_bmap_load_nolock(sfs, sin, blkno, &ino);
        if (ret != 0) {
            goto out;
        }

        ret = sfs_buf_op(sfs, buf, size, ino, 0);
        if (ret != 0) {
            goto out;
        }
        alen += size;
        // }
    }
out:
    *alenp = alen;
    if (offset + alen > sin->din->size) {
```

练习 2: 完成基于文件系统的执行程序机制的实现（需要编码）

改写 proc.c 中的 load_icode 函数和其他相关函数，实现基于文件系统的执行程序机制。执行：make qemu。如果能看看到 sh 用户程序的执行界面，则基本成功了。如果在 sh 用户界面上可以执行”ls”,”hello”等其他放置在 sfs 文件系统中的其他执行程序，则可以认为本实验基本成功。

proc.c/alloc_proc

```
    //LAB8 YOUR CODE : (update LAB6 steps)
     /*
    * below fields(add in LAB6) in proc_struct need to be initialized
    *       struct files_struct * filesp;               file struct point
    */
       proc->state = PROC_UNINIT;
    proc->pid = -1;
    proc->runs = 0;
    proc->kstack = 0;
    proc->need_resched = 0;
    proc->parent = NULL;
    proc->mm = NULL;
    //初始化context结构体
    memset(&(proc->context), 0, sizeof(struct context));
    proc->tf = NULL;
    proc->cr3 = boot_cr3;
    proc->flags = 0;
    proc->wait_state = 0;
    proc->cptr = NULL;
    proc->yptr = NULL;
    proc->optr = NULL;
    proc->rq=NULL;
    list_init(&(proc->run_link));
    list_init(&(proc->hash_link));
    //memset(&(proc->run_link), 0, sizeof(struct context));
    proc->time_slice=0;
    proc->filesp=NULL;
    memset(proc->name, 0, PROC_NAME_LEN+1);
    }
    return proc;
```

proc.c/proc_run

```
void
proc_run(struct proc_struct *proc) {
    if (proc != current) {
        // LAB4:EXERCISE3 YOUR CODE
        /*
        * Some Useful MACROs, Functions and DEFINEs, you can use them in below implementation.
        * MACROs or Functions:
        *   local_intr_save():        Disable interrupts
        *   local_intr_restore():     Enable Interrupts
        *   lcr3():                   Modify the value of CR3 register
        *   switch_to():              Context switching between two processes
        */


    //检查要切换的进程是否与当前正在运行的进程相同，如果相同则不需要切换
        if (proc->pid == current->pid)
            return;
        //禁 用 中 断。 你 可 以 使 用/kern/sync/sync.h 中 定 义 好 的 宏 local_intr_save(x) 和local_intr_restore(x) 来实现关、开中断。
        bool intrstate;
        struct proc_struct *currentpointer = current, *procpointer = proc;
        local_intr_save(intrstate);
        //切换当前进程为要运行的进程。
        current=proc;
        //切换页表，以便使用新进程的地址空间。/libs/riscv.h 中提供了 lcr3(unsigned int cr3)函数，可实现修改 CR3 寄存器值的功能。
        lcr3(procpointer->cr3);
        //实现上下文切换
        flush_tlb();
        switch_to(&(currentpointer->context),&(procpointer->context));
        local_intr_restore(intrstate);
        //LAB8 YOUR CODE : (update LAB4 steps)
        /*
        * below fields(add in LAB6) in proc_struct need to be initialized
        *       before switch_to();you should flush the tlb
        *        MACROs or Functions:
        *       flush_tlb():          flush the tlb
        */
```

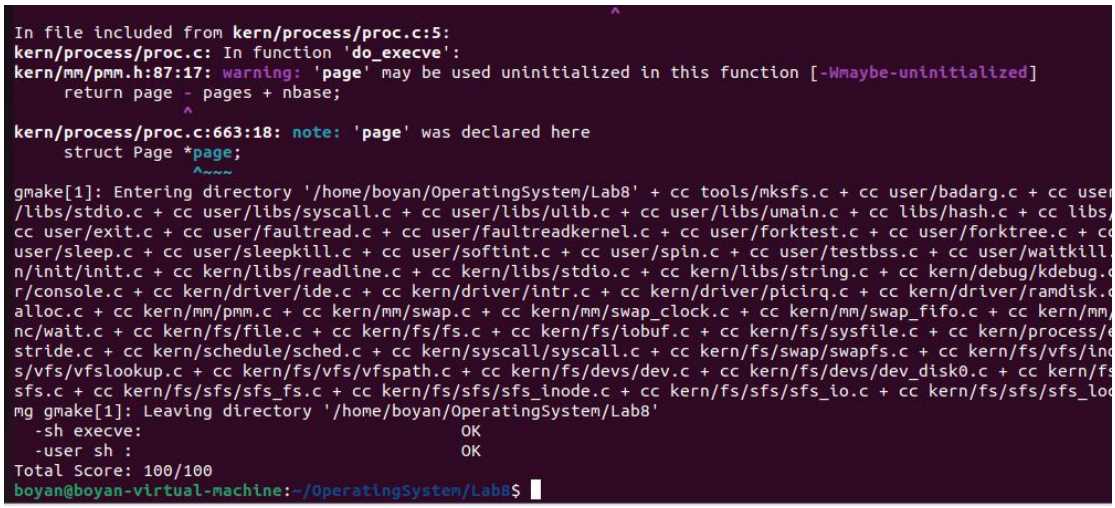
proc.c/load_icode

```
    /
    if (current->mm != NULL) {
        panic("load_icode: current->mm must be empty.\n");
    }

    int ret = -E_NO_MEM;
    struct mm_struct *mm;
    //(1) create a new mm for current process
    if ((mm = mm_create()) == NULL) {
        goto bad_mm;
    }
    //(2) create a new PDT, and mm->pgdir= kernel virtual addr of PDT
    if (setup_pgdir(mm) != 0) {
        goto bad_pgdir_cleanup_mm;
    }
    //(3) copy TEXT/DATA/BSS parts in binary to memory space of process
    struct Page *page;
    struct elfhdr __elf, *elf = &__elf;
    struct proghdr __ph, * ph = &__ph;
    //(3.1) read raw data content in file and resolve elfhdr
    load_icode_read(fd, (void *)elf, sizeof(struct elfhdr), 0);
    // //(3.2) read raw data content in file and resolve proghdr based on info in elfhdr
    //load_icode_read(fd, (void *)ph, sizeof(struct proghdr), elf->e_phoff );
    //(3.3) This program is valid?
    if (elf->e_magic != ELF_MAGIC) {
        ret = -E_INVAL_ELF;
        goto bad_elf_cleanup_pgdir;
    }
```

```
uint32_t vm_flags, perm;
struct proghdr *ph_end = ph + elf->e_phnum;
// for (; ph < ph_end; ph ++) {
for(int index=0; index<elf->e_phnum; index++)
{
    //(3.4) find every program section headers
    off_t ph_off = elf->e_phoff + sizeof(struct proghdr) * index;

    load_icode_read(fd, (void*)ph, sizeof(struct proghdr), ph_off);
    if (ph->p_type != ELF_PT_LOAD) {
        continue ;
    }
    if (ph->p_filesz > ph->p_memsz) {
        ret = -E_INVAL_ELF;
        goto bad_cleanup_mmap;
    }
    if (ph->p_filesz == 0) {
        // continue ;
    }
//(3.5) call mm_map fun to setup the new vma ( ph->p_va, ph->p_memsz)
    vm_flags = 0, perm = PTE_U | PTE_V;
    if (ph->p_flags & ELF_PF_X) vm_flags |= VM_EXEC;
    if (ph->p_flags & ELF_PF_W) vm_flags |= VM_WRITE;
    if (ph->p_flags & ELF_PF_R) vm_flags |= VM_READ;
    // modify the perm bits here for RISC-V
    if (vm_flags & VM_READ) perm |= PTE_R;
    if (vm_flags & VM_WRITE) perm |= (PTE_W | PTE_R);
    if (vm_flags & VM_EXEC) perm |= PTE_X;
    if ((ret = mm_map(mm, ph->p_va, ph->p_memsz, vm_flags, NULL)) != 0) {
        goto bad_cleanup_mmap;
    }
    size_t from = ph->p_offset;
    size_t off, size;
    uintptr_t start = ph->p_va, end, la = ROUNDDOWN(start, PGSIZE);

    ret = -E_NO_MEM;
```

```
//(3.6.1) copy TEXT/DATA section of bianry program
   while (start < end) {
       if ((page = pgdir_alloc_page(mm->pgdir, la, perm)) == NULL) {
           goto bad_cleanup_mmap;
       }
       off = start - la, size = PGSIZE - off, la += PGSIZE;
       if (end < la) {
           size -= la - end;
       }
       load_icode_read(fd, page2kva(page) + off, size, from);
       start += size, from += size;
   }

 //(3.6.2) build BSS section of binary program
   end = ph->p_va + ph->p_memsz;
   if (start < la) {
       /* ph->p_memsz == ph->p_filesz */
       if (start == end) {
           continue ;
       }
       off = start + PGSIZE - la, size = PGSIZE - off;
       if (end < la) {
           size -= la - end;
       }
       memset(page2kva(page) + off, 0, size);
       start += size;
       assert((end < la && start == end) || (end >= la && start == la));
   }
   while (start < end) {
       if ((page = pgdir_alloc_page(mm->pgdir, la, perm)) == NULL) {
           goto bad_cleanup_mmap;
       }
       off = start - la, size = PGSIZE - off, la += PGSIZE;
       if (end < la) {
           size -= la - end;
       }
```

为当前进程创建一个新的内存管理结构（mm）；创建一个新的页目录表（PDT），并将 mm->pgdir 设置为页目录表的内核虚拟地址 ，将二进制文件中的 TEXT/DATA/BSS 部分复制到进程的内存空间中 ；读取文件中的原始数据内容并解析 ELF 头部（elfhdr）， 根据 ELF 头部信息，读取文件中的原始数据内容并解析程序头部（proghdr），调用 mm_map 构建与 TEXT/DATA 相关的虚拟内存区域（VMA），调用 pgdir_alloc_page 为 TEXT/DATA 分配页面，读取文件内容并复制到新分配的页面中，调用 pgdir_alloc_page 为 BSS 分配页面，在这些页面中填充零 ；调用 mm_map 设置用户栈，并将参数放入用户栈； 设置当前进程的内存管理结构、cr3 寄存器，重置页目录表（使用 lcr3 宏） ； 在用户栈中设置 uargc 和 uargv ； 为用户环境设置陷阱帧（trapframe） ；如果前面的步骤失败，应清理环境 */


^
In file included from kern/process/proc.c:5:
kern/process/proc.c: In function 'do_execve':
kern/mm/pmm.h:87:17: warning: 'page' may be used uninitialized in this function [-Wmaybe-uninitialized]
return page - pages + nbase;
^
kern/process/proc.c:663:18: note: 'page' was declared here
struct Page *page;
^~~~
gmake[1]: Entering directory '/home/boyan/OperatingSystem/Lab8' + cc tools/mksfs.c + cc user/badarg.c + cc user
/libs/stdio.c + cc user/libs/syscall.c + cc user/libs/ulib.c + cc user/libs/umain.c + cc libs/hash.c + cc libs/
cc user/exit.c + cc user/faultread.c + cc user/faultreadkernel.c + cc user/forktest.c + cc user/forktree.c + cc
user/sleep.c + cc user/sleepkill.c + cc user/softint.c + cc user/spin.c + cc user/testbss.c + cc user/waitkill.
n/init/init.c + cc kern/libs/readline.c + cc kern/libs/stdio.c + cc kern/libs/string.c + cc kern/debug/kdebug.c
r/console.c + cc kern/driver/ide.c + cc kern/driver/intr.c + cc kern/driver/picirq.c + cc kern/driver/ramdisk.c
alloc.c + cc kern/mm/pmm.c + cc kern/mm/swap.c + cc kern/mm/swap_clock.c + cc kern/mm/swap_fifo.c + cc kern/mm/
nc/wait.c + cc kern/fs/file.c + cc kern/fs/fs.c + cc kern/fs/iobuf.c + cc kern/fs/sysfile.c + cc kern/process/e
stride.c + cc kern/schedule/sched.c + cc kern/syscall/syscall.c + cc kern/fs/swap/swapfs.c + cc kern/fs/vfs/inc
s/vfs/vfslookup.c + cc kern/fs/vfs/vfspath.c + cc kern/fs/devs/dev.c + cc kern/fs/devs/dev_disk0.c + cc kern/fs
sfs.c + cc kern/fs/sfs/sfs_fs.c + cc kern/fs/sfs/sfs_inode.c + cc kern/fs/sfs/sfs_io.c + cc kern/fs/sfs/sfs_loc
mg gmake[1]: Leaving directory '/home/boyan/OperatingSystem/Lab8'
-sh execve: OK
-user sh : OK
Total Score: 100/100
boyan@boyan-virtual-machine:~/OperatingSystem/Lab8$