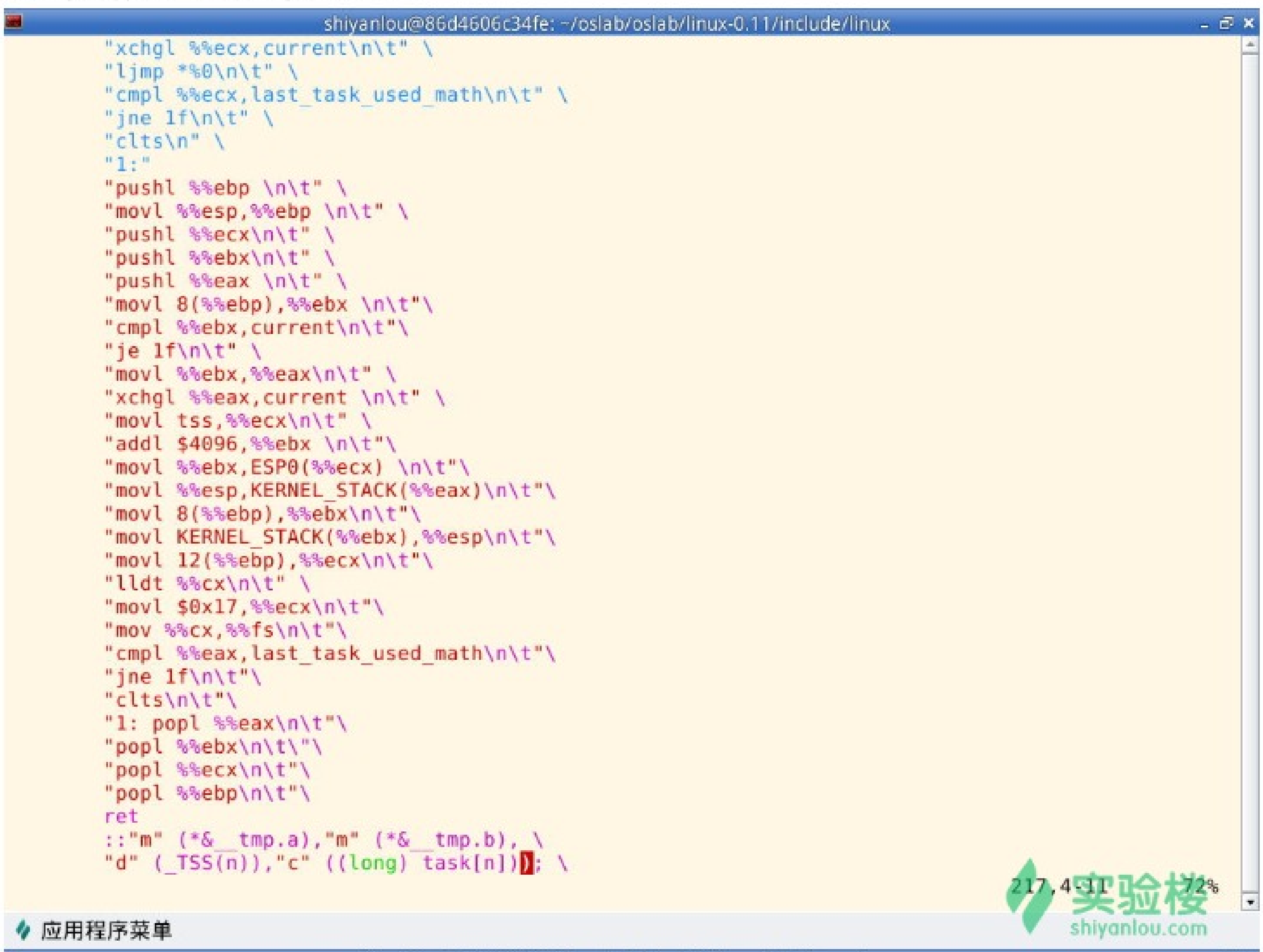
“操作系统原理与实践”实验报告

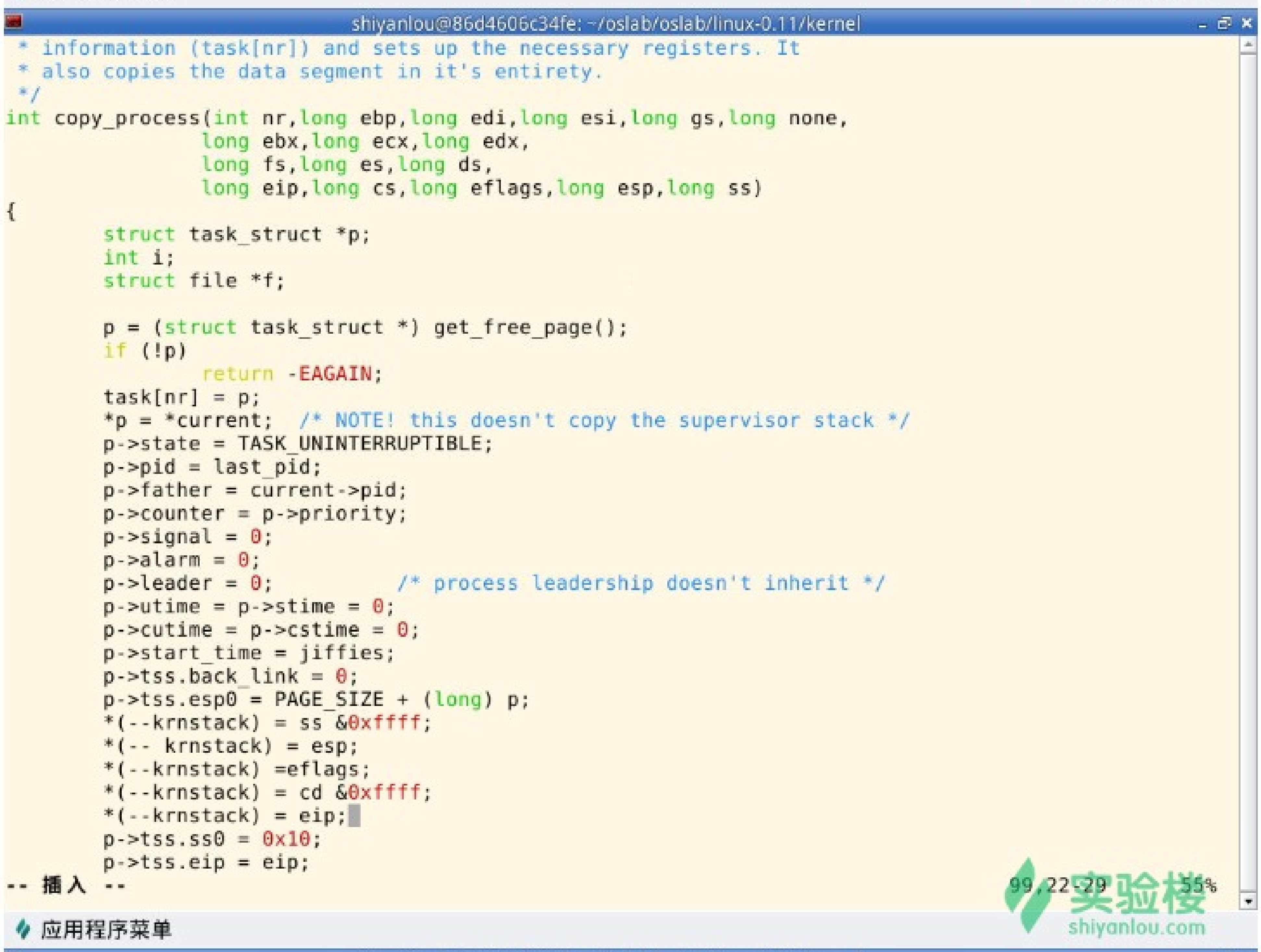
基于内核栈切换的进程切换

问题1： 由于Linux 0.11进程的内核栈和该进程的PCB在同一页内存上（一块4KB大小的内存），其中PCB位于这页内存的低地址，栈位于这页内存的高地址；加4096就可以得到内核栈地址。tss.ss0是内核数据段，现在只用一个tss，因此不需要设置了。 问题2： eax =0 为了与父进程区分开 cpoy_process() 让eax=0 这段代码中的ebx和ecx来自copy_process()的形参，是段寄存器， ebp是用户栈地址，一定要设置，不设置子进程就没有用户栈了 问题3： 这两句代码的含义是重新取一下段寄存器fs的值，这两句话必须要加、也必须要出现在切换完LDT之后，这是因为通过fs访问进程的用户态内存，LDT切换完成就意味着切换了分配给进程的用户态内存地址空间，所以前一个fs指向的是上一个进程的用户态内存，而现在需要执行下一个进程的用户态内存，所以就需要用这两条指令来重取fs。 出现在LDT之前访问的就还是上一个进程的用户态内存


shiyanlou@86d4606c34fe: ~/oslab/oslab/linux-0.11/include/linux

```
        "xchgl %%ecx,current\n\t" \
        "ljmp *%0\n\t" \
        "cmpl %%ecx,last_task_used_math\n\t" \
        "jne 1f\n\t" \
        "clts\n" \
        "1:"
        "pushl %%ebp \n\t" \
        "movl %%esp,%%ebp \n\t" \
        "pushl %%ecx\n\t" \
        "pushl %%ebx\n\t" \
        "pushl %%eax \n\t" \
        "movl 8(%%ebp),%%ebx \n\t"\
        "cmpl %%ebx,current\n\t"\
        "je 1f\n\t" \
        "movl %%ebx,%%eax\n\t" \
        "xchgl %%eax,current \n\t" \
        "movl tss,%%ecx\n\t" \
        "addl $4096,%%ebx \n\t"\
        "movl %%ebx,ESP0(%%ecx) \n\t"\
        "movl %%esp,KERNEL_STACK(%%eax)\n\t"\
        "movl 8(%%ebp),%%ebx\n\t"\
        "movl KERNEL_STACK(%%ebx),%%esp\n\t"\
        "movl 12(%%ebp),%%ecx\n\t"\
        "lldt %%cx\n\t" \
        "movl $0x17,%%ecx\n\t"\
        "mov %%cx,%%fs\n\t"\
        "cmpl %%eax,last_task_used_math\n\t"\
        "jne 1f\n\t"\
        "clts\n\t"\
        "1: popl %%eax\n\t"\
        "popl %%ebx\n\t\"\
        "popl %%ecx\n\t"\
        "popl %%ebp\n\t"\
        ret
        ::"m" (*&__tmp.a),"m" (*&__tmp.b), \
        "d" (_TSS(n)),"c" ((long) task[n])); \
                                                                217,4-11      72%
```

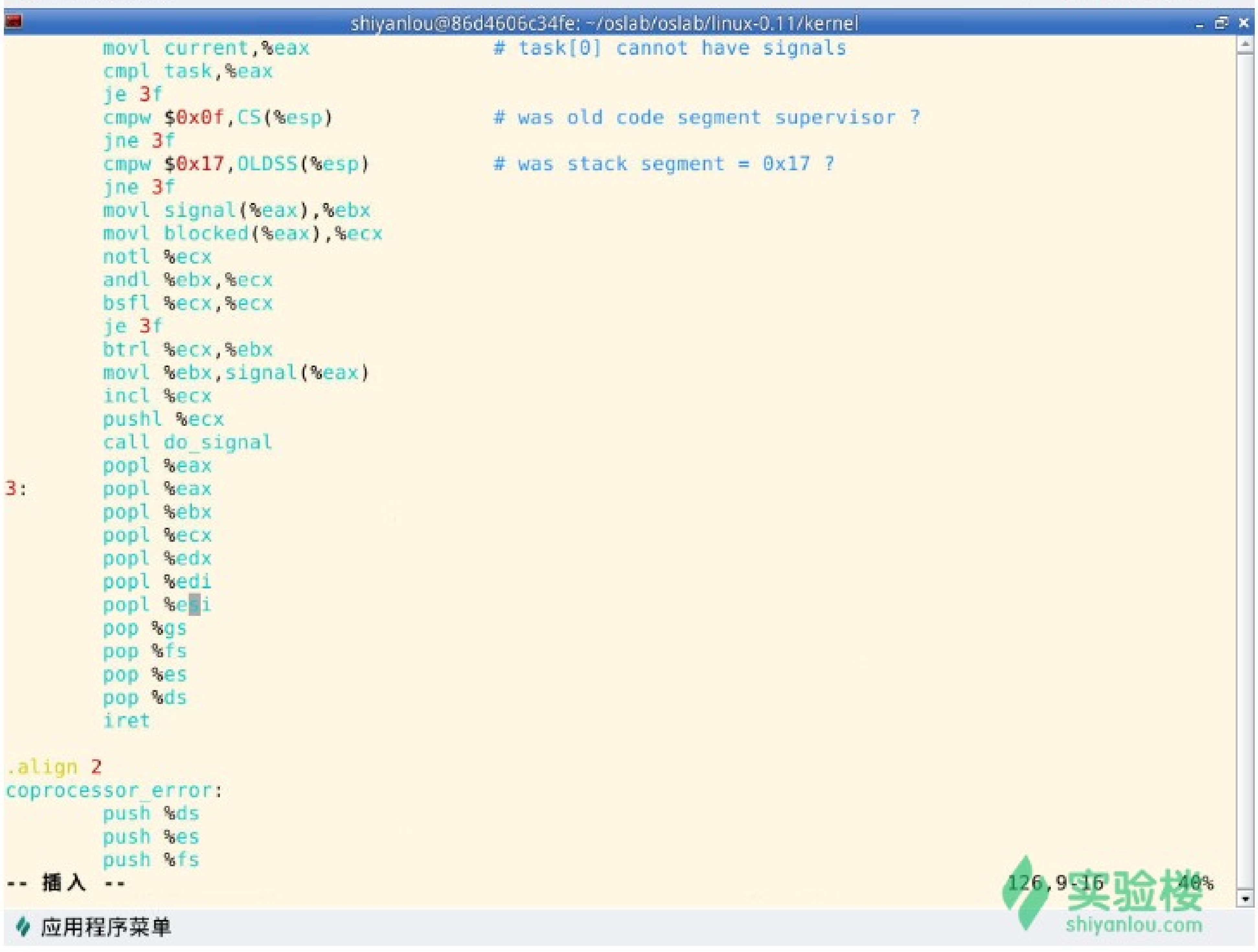
应用程序菜单


shiyanlou@86d4606c34fe: ~/oslab/oslab/linux-0.11/kernel

```
 * information (task[nr]) and sets up the necessary registers. It
 * also copies the data segment in it's entirety.
 */
int copy_process(int nr,long ebp,long edi,long esi,long gs,long none,
                long ebx,long ecx,long edx,
                long fs,long es,long ds,
                long eip,long cs,long eflags,long esp,long ss)
{
        struct task_struct *p;
        int i;
        struct file *f;

        p = (struct task_struct *) get_free_page();
        if (!p)
                return -EAGAIN;
        task[nr] = p;
        *p = *current;  /* NOTE! this doesn't copy the supervisor stack */
        p->state = TASK_UNINTERRUPTIBLE;
        p->pid = last_pid;
        p->father = current->pid;
        p->counter = p->priority;
        p->signal = 0;
        p->alarm = 0;
        p->leader = 0;          /* process leadership doesn't inherit */
        p->utime = p->stime = 0;
        p->cutime = p->cstime = 0;
        p->start_time = jiffies;
        p->tss.back_link = 0;
        p->tss.esp0 = PAGE_SIZE + (long) p;
        *(--krnstack) = ss &0xffff;
        *(-- krnstack) = esp;
        *(--krnstack) =eflags;
        *(--krnstack) = cd &0xffff;
        *(--krnstack) = eip;
        p->tss.ss0 = 0x10;
        p->tss.eip = eip;
-- 插入 --                                                         99,22-29      55%
```

应用程序菜单


shiyanlou@86d4606c34fe: ~/oslab/oslab/linux-0.11/kernel

```
        movl current,%eax               # task[0] cannot have signals
        cmpl task,%eax
        je 3f
        cmpw $0x0f,CS(%esp)             # was old code segment supervisor ?
        jne 3f
        cmpw $0x17,OLDSS(%esp)          # was stack segment = 0x17 ?
        jne 3f
        movl signal(%eax),%ebx
        movl blocked(%eax),%ecx
        notl %ecx
        andl %ebx,%ecx
        bsfl %ecx,%ecx
        je 3f
        btrl %ecx,%ebx
        movl %ebx,signal(%eax)
        incl %ecx
        pushl %ecx
        call do_signal
        popl %eax
3:      popl %eax
        popl %ebx
        popl %ecx
        popl %edx
        popl %edi
        popl %esi
        pop %gs
        pop %fs
        pop %es
        pop %ds
        iret

.align 2
coprocessor_error:
        push %ds
        push %es
        push %fs
-- 插入 --                                                         126,9-16      40%
```

应用程序菜单