[TOC]

lab 4 □□□□

□□□□□□□□□□□□□□□□

□□1□□□□□□□□□□□□□□

□□□□

```
// □□□□□□□□□ PROC_UNINIT□□□□□□□□□□□□□
proc->state = PROC_UNINIT;
// □□□□□ID□ -1□□□□□□□□□□ID
proc->pid = -1;
// □□□□□□□□□□□ 0□□□□□□□□□
proc->runs = 0;
// □□□□□□□ 0□□□□□□□□□□□□□□
proc->kstack = 0;
// □□□□□□□□□□□□□□ 0□□□□□□□□□□
proc->need_resched = 0;
// □□□□□□□□□□ NULL□□□□□□□□□□□□ init □□□
proc->parent = NULL;
// □□□□□□□□□□ (mm_struct) □□□□ NULL□□□□□□□□□□□
proc->mm = NULL;
// □□□□□□□ (context) □□□□□□□□□□□□□□
memset(&(proc->context), 0, sizeof(struct context));
// □□□□□□□□□ (trapframe) □ NULL□□□□□□□□□□□□□□□□□□
proc->tf = NULL;
// □□□□□ CR3 □□□□ boot_cr3□□□□□□□□□□□□□□□□□
proc->cr3 = boot_cr3;
// □□□□□□□□□□ 0□□□□□□□□□□□□
proc->flags = 0;
// □□□□□□□□□□□□□□□□□□□□□□□□ PROC_NAME_LEN + 1
memset(proc->name, 0, PROC_NAME_LEN + 1);
```

`struct context context` □ `struct trapframe *tf` □□□

□ `proc_struct` □□□□□`struct context context` □ `struct trapframe *tf` □□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□/□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1. `struct context context`

`context` □□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□ □□□ `context` □□□□□□□□□□ □□□□□□□□□□□

- □□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ context □□□□□□□□□□□□□□□□ □□□□□□□□□□

□□ context □□□□□□□□ □ context □□□□□□□□□□□□□□□□□□□□□

2. **`struct trapframe *tf`**

`trapframe` □□□ □□□□□□□□□□□□□□□□□□ `trapframe` □□□□

- □□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□CPU □□□□□□□□□□□□□□□□□□□□□□□ `trapframe` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `trapframe` □□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `trapframe`□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `trapframe` □□□□□□□□□□□□□□□□

□□□

- **`struct context context`** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- **`struct trapframe *tf`** □□□

□□□□□`context` □□□□□□□□□□□□□□□□□ `trapframe` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□

□□2□□□□□□□□□□□□□□□□□□□□□□

> □□□□□□□□□□□□□□□□□□□□□kernel_thread□□□□□□**do_fork**□□□□□□□□□□□□□□□□do_kernel□□□□□ alloc_proc□□□□□□□□□□□□□□□□□□□alloc_proc□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ucore□□□□ do_fork□□□□□□□□□□□do_fork□□ □□□**"fork"**□□□□□**stack**□**trapframe**□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ kern/process/proc.c□□do_fork□□□□□□□□□□□□□□□□□□□□□
>
> - □□alloc_proc□□□□□□□□□□□□□□
> - □□□□□□□□□□□□□
> - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
> - □□□□□□□□□□□□
> - □□□□□□□□□□□□□
> - □□□□□□
> - □□□□□□□
>
> □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
>
> - □□□ucore□□□□□□□□□fork□□□□□□□□id□□□□□□□□□□□□

2.1 □□do_fork□□

`do_fork` □□□□□□□□□□□□□□□□□□,□□□ □□□□□□□□□□□□□

- □□ `alloc_proc` □□□□□□□□□□□□□
- □□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□
- □□□□□
- □□□□□□

```
    proc = alloc_proc();                   // □□□□□□□□□□□□□□□
    proc->parent = current;                // □□□□□□□□□□□□□□
    setup_kstack(proc);                    // □□□□□□□□□
    copy_mm(clone_flags, proc);            // □□□□□□□□□□□□□□□□
    copy_thread(proc, stack, tf);          // □□□□□□□□□□□□□□□□
    int pid = get_pid();                   // □□□□□□□□ID
    proc->pid = pid;                       // □□□□□□□□ID
    hash_proc(proc);                       // □□□□□□□□□□□□
    list_add(&proc_list, &(proc->list_link)); // □□□□□□□□□□
    nr_process++;                          // □□□□□□□□□□
    proc->state = PROC_RUNNABLE;           // □□□□□□□□□□□□
    ret = proc->pid;                       // □□□□□□□□ID
```

2.1 ucore□□□□□□□□□fork□□□□□□□□□id

ucore□□□□□□□fork□□□□□□□□id□□□□□□get_pid□□id□□□□□□□□□□□□□□□□□last_id□□□□□□□□□□□□□id□□□□□□□□□□□□□□last_id□□1□□□□□□□□□□□□□□□□□□□id□□□id□

□□□□□□□□□□□id□□□□□□□□□□id□□□□□□□□□□□□□□□□□□□□□□fork□□□□□□□□□id□

□□3□□□proc_run □□□□□□□□

> proc_run□□□□□□□□□□□CPU□□□□□□□□□□□□□□□
>
> - □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
> - □□□□□□□□□□`/kern/sync/sync.h`□□□□□□`local_intr_save(x)`□`local_intr_restore(x)`□□□□□□□□□
> - □□□□□□□□□□□□□□
> - □□□□□□□□□□□□□□□□□□`/libs/riscv.h`□□□□`lcr3(unsigned int cr3)`□□□□□□□□CR3□□□□□□□□
> - □□□□□□□□`/kern/process`□□□□□□□□□`switch.S`□□□□□□`switch_to()`□□□□□□□□□□□context□□□
> - □□□□□
>
> □□□□□□□□
>
> - □□□□□□□□□□□□□□□□□□□□□□□□

3.1 schedule() □□□□□□□□□

[illegible]schd.c[illegible]schedule()[illegible]

```c
void schedule(void) {
    bool intr_flag;//[illegible]
    list_entry_t *le, *last;//[illegible]
    struct proc_struct *next = NULL;//[illegible]
    local_intr_save(intr_flag);//[illegible]
    {
        current->need_resched = 0;
        //[illegible]idle[illegible]idle[illegible]
        last = (current == idleproc) ? &proc_list : &(current->list_link);
        le = last;

        do {//[illegible]proc_list[illegible]
            if ((le = list_next(le)) != &proc_list) {
                next = le2proc(le, list_link);
                if (next->state == PROC_RUNNABLE) {
                    break;//[illegible]
                }
            }
        } while (le != last);

        if (next == NULL || next->state != PROC_RUNNABLE) {
            next = idleproc;//[illegible]idle
        }

        next->runs ++;//[illegible]

        if (next != current) {
            proc_run(next);//[illegible]proc_run[illegible]
        }
    }
    local_intr_restore(intr_flag);//[illegible]
}
```

schedule() [illegible] **FIFO (First In, First Out)** [illegible] proc_list [illegible]

1. [illegible] current->need_resched [illegible] 0[illegible]
2. [illegible]le[illegible]idle[illegible]current[illegible] proc_list [illegible] PROC_RUNNABLE [illegible]
3. [illegible] idleproc[illegible]
4. [illegible]
    - [illegible]runs++[illegible]
    - [illegible] proc_run() [illegible]
5. [illegible]

3.2 lcr3[illegible]

1. [illegible] (boot_cr3)

- `boot_cr3` □□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□

2. `lcr3` □□ □□□□□□□□□□□□ RISC-V □ `sptbr` CSR□□□□□□□□□□□□□□□□□□□□□ `proc->cr3` □□ □□□□□□□□□□□□□ □□ □□□□□ **(`sptbr`)** □□□□ Supervisor Page Table Base Register□□□ CPU □□□□□ □□□□□

```
static inline void
lcr3(unsigned int cr3) {
    write_csr(sptbr, SATP32_MODE | (cr3 >> RISCV_PGSHIFT));
}
```

3.3 □□□□

□□□□□ `switch_to` □□□□□ **callee-saved** □□□□□□□□□□□□□□□□□□□□□□□□ `trapframe` □□ □□□□□□□□□□□□□□□□□□

```
.text
# void switch_to(struct proc_struct* from, struct proc_struct* to)
.globl switch_to
switch_to:
    # save from's registers
    STORE ra, 0*REGBYTES(a0)
    STORE sp, 1*REGBYTES(a0)
    STORE s0, 2*REGBYTES(a0)
    STORE s1, 3*REGBYTES(a0)
    STORE s2, 4*REGBYTES(a0)
    STORE s3, 5*REGBYTES(a0)
    STORE s4, 6*REGBYTES(a0)
    STORE s5, 7*REGBYTES(a0)
    STORE s6, 8*REGBYTES(a0)
    STORE s7, 9*REGBYTES(a0)
    STORE s8, 10*REGBYTES(a0)
    STORE s9, 11*REGBYTES(a0)
    STORE s10, 12*REGBYTES(a0)
    STORE s11, 13*REGBYTES(a0)

    # restore to's registers
    LOAD ra, 0*REGBYTES(a1)
    LOAD sp, 1*REGBYTES(a1)
    LOAD s0, 2*REGBYTES(a1)
    LOAD s1, 3*REGBYTES(a1)
    LOAD s2, 4*REGBYTES(a1)
    LOAD s3, 5*REGBYTES(a1)
    LOAD s4, 6*REGBYTES(a1)
    LOAD s5, 7*REGBYTES(a1)
    LOAD s6, 8*REGBYTES(a1)
    LOAD s7, 9*REGBYTES(a1)
    LOAD s8, 10*REGBYTES(a1)
    LOAD s9, 11*REGBYTES(a1)
    LOAD s10, 12*REGBYTES(a1)
    LOAD s11, 13*REGBYTES(a1)

    ret
```

3.4 proc_run□□□□

```
void proc_run(struct proc_struct *proc) {
    if (proc != current) {
       // □□□□□
       bool intr_flag;
       struct proc_struct *prev = current;// prev□□□□□□□□□□□
       local_intr_save(intr_flag);// □□□□□□
       {
        current = proc; // □□□□□□
        lcr3(proc->cr3);//□□□□□□□□□
        switch_to(&(prev->context), &(proc->context));// □□□□□□□
       }
       local_intr_restore(intr_flag); // □□□□□□
    }
}
```

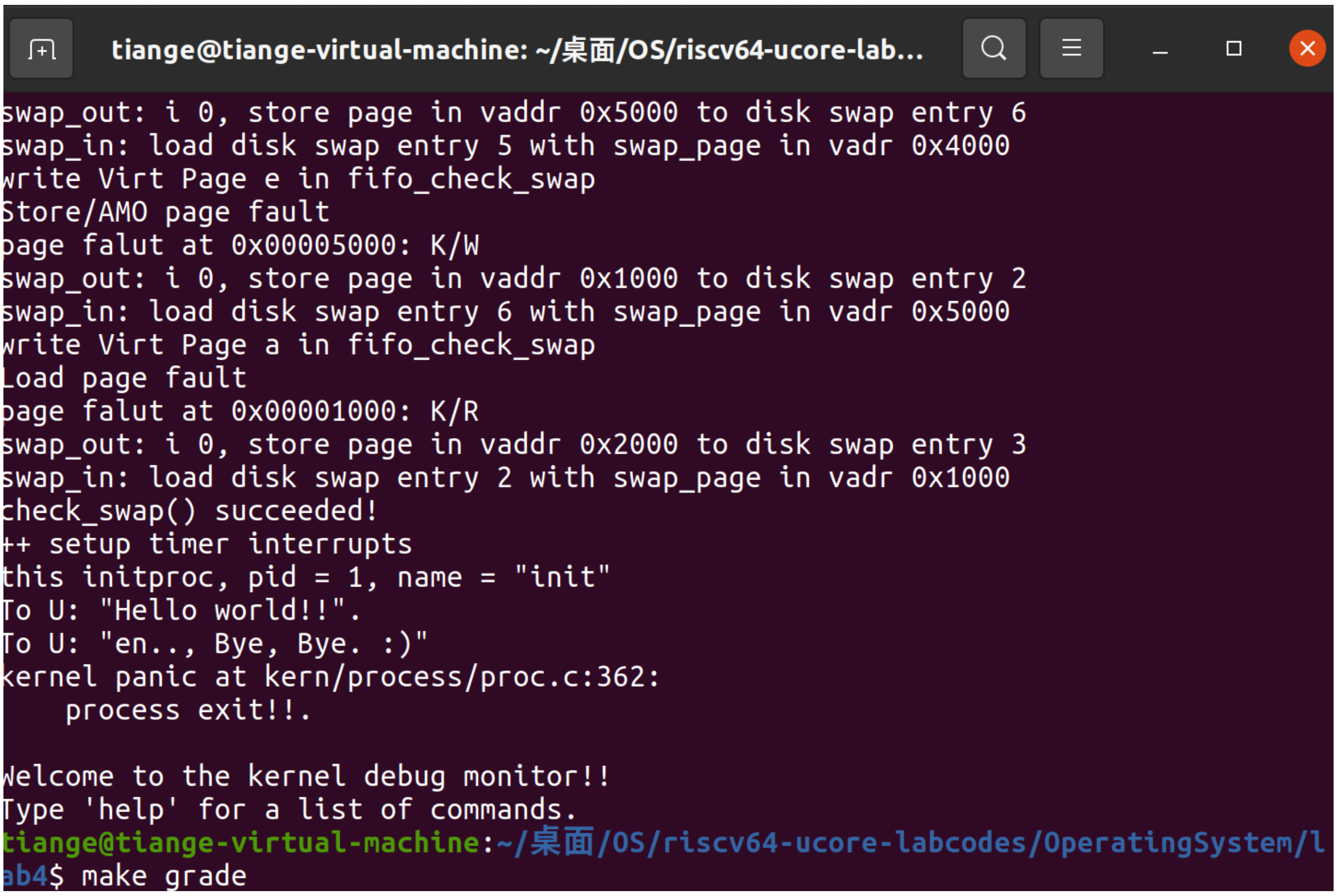
1. □□□□□□□ `local_intr_save` □□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□ `current` □□□□□□□□□ `proc`□□□□□□□□□□□
3. □□□□□□□□□ `lcr3` □□□□□□□□□□□□□□□□□□□□□
4. □□□□□□□□ `switch_to` □□□□□□□□□□□□□□□□□□□□□□
5. □□□□□□□ `local_intr_restore` □□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□

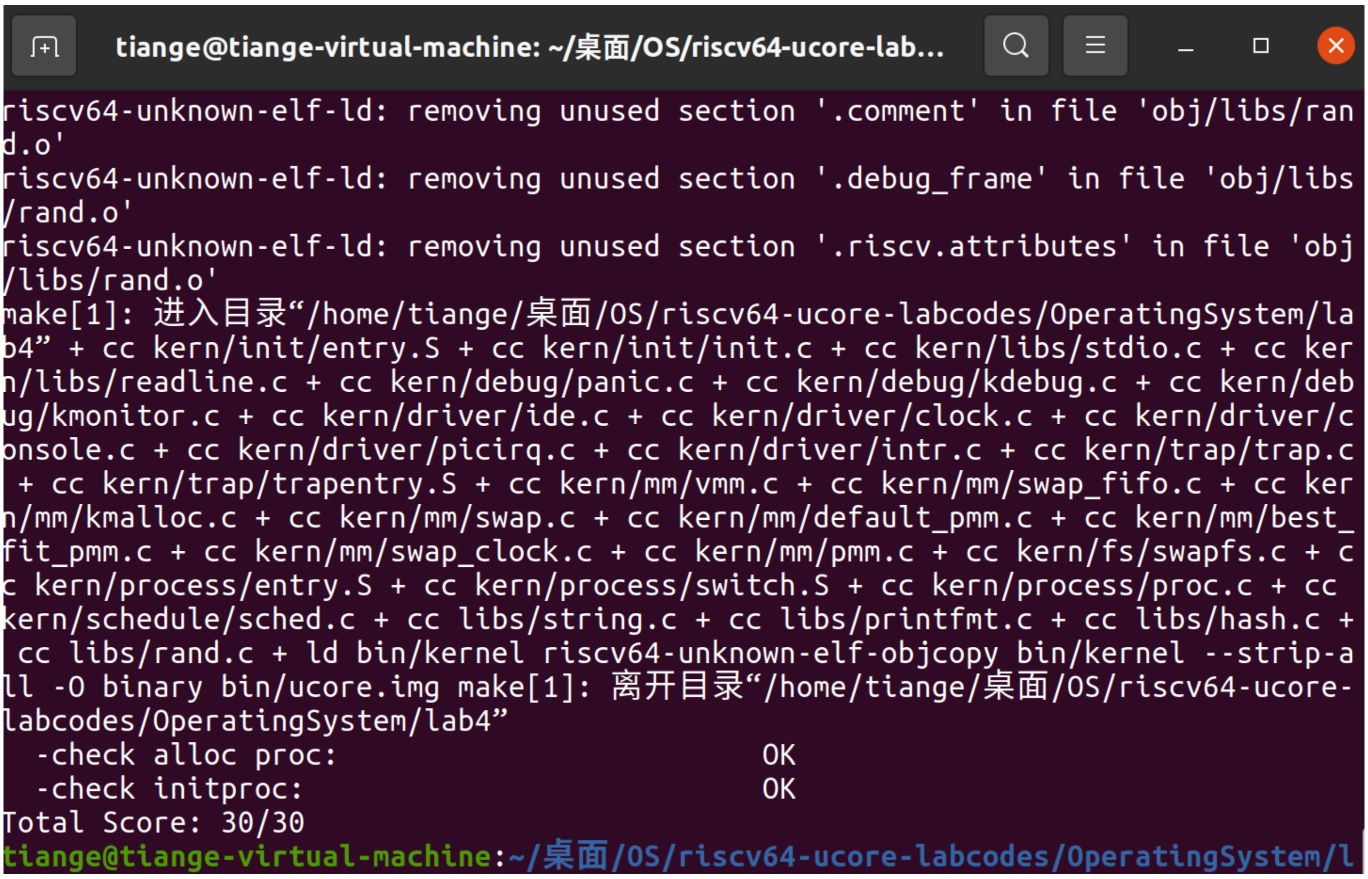
1. idle_thread□□ idle□□□□init□RUNNABLE□□□□□□□□□init□□□
2. init_thread□□□□□□□□□□□□□“Hello World”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□


tiange@tiange-virtual-machine: ~/桌面/OS/riscv64-ucore-lab...

```
swap_out: i 0, store page in vaddr 0x5000 to disk swap entry 6
swap_in: load disk swap entry 5 with swap_page in vadr 0x4000
write Virt Page e in fifo_check_swap
Store/AMO page fault
page falut at 0x00005000: K/W
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
swap_in: load disk swap entry 6 with swap_page in vadr 0x5000
write Virt Page a in fifo_check_swap
Load page fault
page falut at 0x00001000: K/R
swap_out: i 0, store page in vaddr 0x2000 to disk swap entry 3
swap_in: load disk swap entry 2 with swap_page in vadr 0x1000
check_swap() succeeded!
++ setup timer interrupts
this initproc, pid = 1, name = "init"
To U: "Hello world!!".
To U: "en.., Bye, Bye. :)"
kernel panic at kern/process/proc.c:362:
    process exit!!.

Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
tiange@tiange-virtual-machine:~/桌面/OS/riscv64-ucore-labcodes/OperatingSystem/l
ab4$ make grade
```


tiange@tiange-virtual-machine: ~/桌面/OS/riscv64-ucore-lab...

```
riscv64-unknown-elf-ld: removing unused section '.comment' in file 'obj/libs/ran
d.o'
riscv64-unknown-elf-ld: removing unused section '.debug_frame' in file 'obj/libs
/rand.o'
riscv64-unknown-elf-ld: removing unused section '.riscv.attributes' in file 'obj
/libs/rand.o'
make[1]: 进入目录“/home/tiange/桌面/OS/riscv64-ucore-labcodes/OperatingSystem/la
b4” + cc kern/init/entry.S + cc kern/init/init.c + cc kern/libs/stdio.c + cc ker
n/libs/readline.c + cc kern/debug/panic.c + cc kern/debug/kdebug.c + cc kern/deb
ug/kmonitor.c + cc kern/driver/ide.c + cc kern/driver/clock.c + cc kern/driver/c
onsole.c + cc kern/driver/picirq.c + cc kern/driver/intr.c + cc kern/trap/trap.c
 + cc kern/trap/trapentry.S + cc kern/mm/vmm.c + cc kern/mm/swap_fifo.c + cc ker
n/mm/kmalloc.c + cc kern/mm/swap.c + cc kern/mm/default_pmm.c + cc kern/mm/best_
fit_pmm.c + cc kern/mm/swap_clock.c + cc kern/mm/pmm.c + cc kern/fs/swapfs.c + c
c kern/process/entry.S + cc kern/process/switch.S + cc kern/process/proc.c + cc
kern/schedule/sched.c + cc libs/string.c + cc libs/printfmt.c + cc libs/hash.c +
 cc libs/rand.c + ld bin/kernel riscv64-unknown-elf-objcopy bin/kernel --strip-a
ll -O binary bin/ucore.img make[1]: 离开目录“/home/tiange/桌面/OS/riscv64-ucore-
labcodes/OperatingSystem/lab4”
  -check alloc proc:                          OK
  -check initproc:                            OK
Total Score: 30/30
tiange@tiange-virtual-machine:~/桌面/OS/riscv64-ucore-labcodes/OperatingSystem/l
```

□□□□ Challenge□

> □□□□`local_intr_save(intr_flag);....local_intr_restore(intr_flag);`□□□□□□□□□
> □□

□□□□□□:

```
static inline bool __intr_save(void) {
    if (read_csr(sstatus) & SSTATUS_SIE) {
        intr_disable();
        return 1;
    }
    return 0;
}

static inline void __intr_restore(bool flag) {
    if (flag) {
        intr_enable();
    }
}

#define local_intr_save(x)      do { x = __intr_save(); } while (0)
#define local_intr_restore(x)   __intr_restore(x);
```

`local_intr_save` □ `local_intr_restore` □□□□□□□□□□ `sstatus` CSR □□□□□ `SIE` □□□□□□□□□□□□□□□□□□□□□□

□□□□ (`local_intr_save`)□

- □□ `__intr_save()`□□□□□□□□□
    - □□□□□□ (`SSTATUS_SIE = 1`)□□□ `intr_disable()` □□□□□□□□ `1`□
    - □□□□□□□□□□□□ `0`□
- □□□□□□□□□□□□□□□□□□□□□□

□□□□ (`local_intr_restore`)□

- □□ `__intr_restore()`□□□□□□□□□□
    - □□□□□□□□□ (`flag = 1`)□□□ `intr_enable()` □□□□□□□
    - □□□□□□□□□ (`flag = 0`)□□□□□□□□

intr_enable□intr_disable□□□□□:

```
/* intr_enable - enable irq interrupt */
void intr_enable(void) { set_csr(sstatus, SSTATUS_SIE); }

/* intr_disable - disable irq interrupt */
void intr_disable(void) { clear_csr(sstatus, SSTATUS_SIE); }
```

□□□□□

□□□□□

□□□□□□□□□□□□

1. □□□□□□□□

□□

2. □□□□

1. **`proc->context`**: □□□□□□□□□□callee-saved □□□□□
2. **`proc->tf`**: □□□□□□□□□□trapframe□□□□□□□□□□□□□□□□□
3. **`kernel_thread_entry`**: □□□□□□□□□□□
4. **`switch_to`**: □□□□□□□□□□□□□□□□□□□□□□□□□□□□
5. **`forkret`** □ **`__trapret`**: □□□□□□□□□□□□□□□□□□

3. □□□□

3.1 □□□□□□□□

□□ `kernel_thread` □□□□□□□□□

1. □□□□□□□□□□ **tf**□
   - **`s0`** □□□□□□□□□□□□ `fn`□
   - **`s1`** □□□□□□□□□□□□ `arg`□
   - **`epc`** □□□ `kernel_thread_entry`□□□□□□□□□□
   - □□□□□□□□ `sstatus` □□□ S □□□□□□□□□
2. □□ `do_fork`□□□□□□□□□□□□□□□□ `tf`□

3.2 □□□□□□□□

□□ `copy_thread` □□□□□

1. □□□□□□ `tf` □□□□□□□□□□□□□□□□□□ `proc->tf`□
2. □□□□□□□□□□ `proc->context`□□□□
   - **`ra`** □□ `forkret` □□□
   - **`sp`** □□ `proc->tf`□

3.3 □□□□□□

1. □□ `switch_to`□□□□□□□□□
   - □□□□□□□□□□□□**callee-saved** □□□□□ □□ `STORE` □□□ `ra`□`sp` □ `s0-s11` □□□□□□□□□□□□ `proc->context` □□`a0` □□□□
   - □□□□□□□□□□□□ □□ `LOAD` □□□□□□□□□ `proc->context` □□□ `ra`□`sp` □ `s0-s11` □□□□`a1` □□□□
   - □□□□□□□□□ `ra`□□ `forkret` □□□

3.4 □□□□□□

`forkret` □□□□□□□□□□□□□□□□

1. □□□□□□ `trapframe` □□□□□□ `sp`□

2. [illegible] `__trapret`
   - [illegible] `epc` [illegible] `kernel_thread_entry`[illegible]
   - [illegible]

3.5 [illegible]

[illegible] `kernel_thread_entry` [illegible]

1. [illegible] `s1`[illegible] `a0`[illegible]
2. [illegible] `s0`[illegible]
3. [illegible] `do_exit` [illegible]

[illegible]