浙江大學

本科实验报告

课程名称：	操作系统
姓　　名：	徐文皓
学　　院：	计算机科学与技术学院
系：	软件工程系
专　　业：	软件工程
学　　号：	3210102377
指导教师：	夏莹杰

2024 年　01 月　03 日

浙江大学操作系统实验报告

实验名称：______fork 机制

电子邮件地址：__手机：__.

实验地点：______线上______实验日期：2024 年 01 月 03 日

一、实验目的和要求

本实验的要求是：为 task 加入 fork 机制，支持通过 fork 创建用户态 task。

二、实验过程

我们考虑实现 fork()机制。

修改 task_init()，使得本环节只创建一个内核线程。

```c
// arch/riscv/kernel/proc.c
void task_init() {
  ...
  task[1] = (struct task_struct*)kalloc();
  task[1]->state = TASK_RUNNING;
  task[1]->pid = 1;
  task[1]->counter = task_test_counter[1];
  task[1]->priority = task_test_priority[1];
  task[1]->thread.ra = (uint64)__dummy;
  task[1]->thread.sp = (uint64)task[1] + PGSIZE;

  task[1]->vma_cnt = 0;

  task[1]->pgd = (pagetable_t)kalloc();
  memset(task[1]->pgd, 0, PGSIZE);
  memcpy(task[1]->pgd, swapper_pg_dir, PGSIZE);

  load_program(task[1]);
  ...
  printk("\n...proc_init done!\n");
}
```

修改 entry.S，增加一个__ret_from_fork 入口。

```asm
# arch/riscv/kernel/entry.S
_traps:
    ...
    call trap_handler
    .global __ret_from_fork
__ret_from_fork:
    ...
```

我们维护一个全局变量 num_of_task，用于表示目前创建的线程数量。

这个变量同时会在调度算法中造成一些改动。

```c
// arch/riscv/kernel/proc.c
void schedule(void) {
  ...
    for (int i = 1; i < num_of_task; i++) {
      ...
    }
    ...
    for (int i = 1; i < num_of_task; i++) task[i]->counter = rand();
 ...
}
```

实现 sys_clone()，其中的一些关键逻辑是：子线程的 regs 在 task_struct 中的偏移应当与父线程一致，我们由此可以得到子线程的 regs；子线程在被创建后并不会经由 trap_handler()返回，因此要在这里就将它的 sepc 移动到下一条指令的地址处；在建立映射时，遍历其 vma，对 vma 的每一页检索它是否在页表内，如果在说明在父线程中已经建立映射，我们对子线程建立映射；我们不必专门关注子线程的用户态栈，因为它也是 vma 之一。

```c
// arch/riscv/kernel/proc.c
uint64 num_of_task = 2;

uint64_t sys_clone(struct pt_regs* regs) {

  uint64 new_task_index = num_of_task;
  num_of_task++;
  task[new_task_index] = (struct task_struct*)kalloc();
  memcpy((void*)task[new_task_index], (void*)current, PGSIZE);
  task[new_task_index]->pid = new_task_index;

  task[new_task_index]->thread.ra = (uint64)__ret_from_fork;
  struct pt_regs* child_regs = (struct pt_regs*)((uint64)task[new_task_index] + ((uint64)regs -PGROUNDDOWN((uint64)regs)));
  task[new_task_index]->thread.sp = (uint64)child_regs;
  child_regs->x[10] = 0;
  child_regs->sepc = regs->sepc + 4;

  task[new_task_index]->pgd = (pagetable_t)kalloc();

  memset(task[new_task_index]->pgd, 0, PGSIZE);
  memcpy((void*)task[new_task_index]->pgd, (void*)swapper_pg_dir, PGSIZE);

  for (uint64 i = 0; i < task[new_task_index]->vma_cnt; i++) {
    int flag = 1;
    for (uint64 addr = PGROUNDDOWN(task[new_task_index]->vmas[i].vm_start); addr < task[new_task_index]->vmas[i].vm_end; addr = addr + PGSIZE) {
      uint64 VPN[3];
      VPN[0] = (addr >> 12) & 0x1ff;
      VPN[1] = (addr >> 21) & 0x1ff;
      VPN[2] = (addr >> 30) & 0x1ff;
      uint64* pte = current->pgd;
      for (int j = 2; j > 0 && flag; j--) {
        if ((pte[VPN[j]] & 0x1) == 0)flag = 0;
        else pte = (uint64*)(((pte[VPN[j]] >> 10) << 12) + PA2VA_OFFSET);
      }
      if (flag && (pte[VPN[0]] & 0x1) != 0) {
        uint64 page = alloc_page();
        memcpy((void*)page, (void*)addr, PGSIZE);
        create_mapping(task[new_task_index]->pgd, addr, page - PA2VA_OFFSET, PGSIZE, (current->vmas[i].vm_flags & 0b1110) | 0x11);
      }
    }
  }
```

至此，我们已经实现了 fork 机制，随实验提供的 4 个 main()输出结果将在后面 4 页中依序列出。

不包含 fork()的 main()每个线程在创建时会产生 3 次缺页异常。这四个线程一共产生了 12 次缺页异常。

包含 fork()的第一个 main()在第一个线程产生 3 次缺页异常，之后的子线程会产生 1 次缺页异常(Load Page Fault)。这两个线程一共产生了 4 次缺页异常。

包含 fork()的第二个 main()在第一个线程产生 3 次缺页异常，之后的子线程不产生缺页异常。这两个线程一共产生了 3 次缺页异常。

包含 fork()的第三个 main()在第一个线程产生 3 次缺页异常，之后的子线程不产生缺页异常。这四个线程一共产生了 3 次缺页异常。

```
...proc_init done!
2023[S-Mode] Hello RISC-V

SWITCH TO [pid=1 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010124
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011880, sepc: 0x0000000000010140
[PID = 1] is running, variable: 0
[PID = 1] is running, variable: 1
[PID = 1] is running, variable: 2
[PID = 1] is running, variable: 3
[PID = 1] is running, variable: 4
[PID = 1] is running, variable: 5
[PID = 1] is running, variable: 6
[PID = 1] is running, variable: 7
[PID = 1] is running, variable: 8
[PID = 1] is running, variable: 9
[PID = 1] is running, variable: 10

SWITCH TO [pid=4 counter=5 priority=66]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010124
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011880, sepc: 0x0000000000010140
[PID = 4] is running, variable: 0
[PID = 4] is running, variable: 1
[PID = 4] is running, variable: 2
[PID = 4] is running, variable: 3
[PID = 4] is running, variable: 4
[PID = 4] is running, variable: 5
[PID = 4] is running, variable: 6
[PID = 4] is running, variable: 7
[PID = 4] is running, variable: 8
[PID = 4] is running, variable: 9
[PID = 4] is running, variable: 10
[PID = 4] is running, variable: 11
[PID = 4] is running, variable: 12
[PID = 4] is running, variable: 13

SWITCH TO [pid=3 counter=8 priority=52]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010124
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011880, sepc: 0x0000000000010140
[PID = 3] is running, variable: 0
[PID = 3] is running, variable: 1
[PID = 3] is running, variable: 2
[PID = 3] is running, variable: 3
[PID = 3] is running, variable: 4
[PID = 3] is running, variable: 5
[PID = 3] is running, variable: 6
[PID = 3] is running, variable: 7
[PID = 3] is running, variable: 8
[PID = 3] is running, variable: 9
[PID = 3] is running, variable: 10
[PID = 3] is running, variable: 11
[PID = 3] is running, variable: 12
[PID = 3] is running, variable: 13
[PID = 3] is running, variable: 14
[PID = 3] is running, variable: 15
[PID = 3] is running, variable: 16
[PID = 3] is running, variable: 17
[PID = 3] is running, variable: 18
[PID = 3] is running, variable: 19
[PID = 3] is running, variable: 20
[PID = 3] is running, variable: 21

SWITCH TO [pid=2 counter=9 priority=88]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010124
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011880, sepc: 0x0000000000010140
[PID = 2] is running, variable: 0
[PID = 2] is running, variable: 1
[PID = 2] is running, variable: 2
[PID = 2] is running, variable: 3
[PID = 2] is running, variable: 4
[PID = 2] is running, variable: 5
[PID = 2] is running, variable: 6
[PID = 2] is running, variable: 7
[PID = 2] is running, variable: 8
[PID = 2] is running, variable: 9
[PID = 2] is running, variable: 10
[PID = 2] is running, variable: 11
[PID = 2] is running, variable: 12
[PID = 2] is running, variable: 13
[PID = 2] is running, variable: 14
[PID = 2] is running, variable: 15
[PID = 2] is running, variable: 16
[PID = 2] is running, variable: 17
[PID = 2] is running, variable: 18
[PID = 2] is running, variable: 19
[PID = 2] is running, variable: 20
[PID = 2] is running, variable: 21
[PID = 2] is running, variable: 22
[PID = 2] is running, variable: 23
[PID = 2] is running, variable: 24

SWITCH TO [pid=1 counter=1 priority=37]
[PID = 1] is running, variable: 11
[PID = 1] is running, variable: 12
[PID = 1] is running, variable: 13

SWITCH TO [pid=2 counter=4 priority=88]
[PID = 2] is running, variable: 25
[PID = 2] is running, variable: 26
[PID = 2] is running, variable: 27
[PID = 2] is running, variable: 28
[PID = 2] is running, variable: 29
[PID = 2] is running, variable: 30
[PID = 2] is running, variable: 31
[PID = 2] is running, variable: 32
[PID = 2] is running, variable: 33
[PID = 2] is running, variable: 34

SWITCH TO [pid=4 counter=4 priority=66]
[PID = 4] is running, variable: 14
[PID = 4] is running, variable: 15
[PID = 4] is running, variable: 16
[PID = 4] is running, variable: 17
[PID = 4] is running, variable: 18
[PID = 4] is running, variable: 19
[PID = 4] is running, variable: 20
[PID = 4] is running, variable: 21
[PID = 4] is running, variable: 22
[PID = 4] is running, variable: 23
[PID = 4] is running, variable: 24

SWITCH TO [pid=3 counter=10 priority=52]
[PID = 3] is running, variable: 22
[PID = 3] is running, variable: 23
```

```
...proc_init done!
2023[S-Mode] Hello RISC-V

SWITCH TO [pid=1 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010158
FORK [pid=2 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011978, sepc: 0x00000000000101f0
[U-PARENT] pid: 1 is running!, global_variable: 0
[U-PARENT] pid: 1 is running!, global_variable: 1
[U-PARENT] pid: 1 is running!, global_variable: 2
[U-PARENT] pid: 1 is running!, global_variable: 3
[U-PARENT] pid: 1 is running!, global_variable: 4
[U-PARENT] pid: 1 is running!, global_variable: 5
[U-PARENT] pid: 1 is running!, global_variable: 6
[U-PARENT] pid: 1 is running!, global_variable: 7
[U-PARENT] pid: 1 is running!, global_variable: 8
[U-PARENT] pid: 1 is running!, global_variable: 9
[U-PARENT] pid: 1 is running!, global_variable: 10

SWITCH TO [pid=2 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011978, sepc: 0x000000000001018c
[U-CHILD] pid: 2 is running!, global_variable: 0
[U-CHILD] pid: 2 is running!, global_variable: 1
[U-CHILD] pid: 2 is running!, global_variable: 2
[U-CHILD] pid: 2 is running!, global_variable: 3
[U-CHILD] pid: 2 is running!, global_variable: 4
[U-CHILD] pid: 2 is running!, global_variable: 5
[U-CHILD] pid: 2 is running!, global_variable: 6
[U-CHILD] pid: 2 is running!, global_variable: 7
[U-CHILD] pid: 2 is running!, global_variable: 8
[U-CHILD] pid: 2 is running!, global_variable: 9
[U-CHILD] pid: 2 is running!, global_variable: 10

SWITCH TO [pid=1 counter=1 priority=37]
[U-PARENT] pid: 1 is running!, global_variable: 11
[U-PARENT] pid: 1 is running!, global_variable: 12
[U-PARENT] pid: 1 is running!, global_variable: 13

SWITCH TO [pid=2 counter=4 priority=37]
[U-CHILD] pid: 2 is running!, global_variable: 11
[U-CHILD] pid: 2 is running!, global_variable: 12
[U-CHILD] pid: 2 is running!, global_variable: 13
[U-CHILD] pid: 2 is running!, global_variable: 14
[U-CHILD] pid: 2 is running!, global_variable: 15
[U-CHILD] pid: 2 is running!, global_variable: 16
[U-CHILD] pid: 2 is running!, global_variable: 17
[U-CHILD] pid: 2 is running!, global_variable: 18
[U-CHILD] pid: 2 is running!, global_variable: 19
[U-CHILD] pid: 2 is running!, global_variable: 20
[U-CHILD] pid: 2 is running!, global_variable: 21
[U-CHILD] pid: 2 is running!, global_variable: 22
[U-CHILD] pid: 2 is running!, global_variable: 23
[U-CHILD] pid: 2 is running!, global_variable: 24
[U-CHILD] pid: 2 is running!, global_variable: 25
[U-CHILD] pid: 2 is running!, global_variable: 26
[U-CHILD] pid: 2 is running!, global_variable: 27
[U-CHILD] pid: 2 is running!, global_variable: 28
[U-CHILD] pid: 2 is running!, global_variable: 29
[U-CHILD] pid: 2 is running!, global_variable: 30
[U-CHILD] pid: 2 is running!, global_variable: 31
[U-CHILD] pid: 2 is running!, global_variable: 32

SWITCH TO [pid=1 counter=10 priority=37]
[U-PARENT] pid: 1 is running!, global_variable: 14
[U-PARENT] pid: 1 is running!, global_variable: 15
[U-PARENT] pid: 1 is running!, global_variable: 16
[U-PARENT] pid: 1 is running!, global_variable: 17
[U-PARENT] pid: 1 is running!, global_variable: 18
[U-PARENT] pid: 1 is running!, global_variable: 19
[U-PARENT] pid: 1 is running!, global_variable: 20
[U-PARENT] pid: 1 is running!, global_variable: 21
[U-PARENT] pid: 1 is running!, global_variable: 22
[U-PARENT] pid: 1 is running!, global_variable: 23
QEMU: Terminated
```

```
...proc_init done!
2023[S-Mode] Hello RISC-V

SWITCH TO [pid=1 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010158
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011a00, sepc: 0x000000000001017c
[U] pid: 1 is running!, global_variable: 0
[U] pid: 1 is running!, global_variable: 1
[U] pid: 1 is running!, global_variable: 2
FORK [pid=2 counter=4 priority=37]
[U-PARENT] pid: 1 is running!, global_variable: 3
[U-PARENT] pid: 1 is running!, global_variable: 4
[U-PARENT] pid: 1 is running!, global_variable: 5
[U-PARENT] pid: 1 is running!, global_variable: 6
[U-PARENT] pid: 1 is running!, global_variable: 7
[U-PARENT] pid: 1 is running!, global_variable: 8
[U-PARENT] pid: 1 is running!, global_variable: 9
[U-PARENT] pid: 1 is running!, global_variable: 10
[U-PARENT] pid: 1 is running!, global_variable: 11
[U-PARENT] pid: 1 is running!, global_variable: 12
[U-PARENT] pid: 1 is running!, global_variable: 13

SWITCH TO [pid=2 counter=4 priority=37]
[U-CHILD] pid: 2 is running!, global_variable: 3
[U-CHILD] pid: 2 is running!, global_variable: 4
[U-CHILD] pid: 2 is running!, global_variable: 5
[U-CHILD] pid: 2 is running!, global_variable: 6
[U-CHILD] pid: 2 is running!, global_variable: 7
[U-CHILD] pid: 2 is running!, global_variable: 8
[U-CHILD] pid: 2 is running!, global_variable: 9
[U-CHILD] pid: 2 is running!, global_variable: 10
[U-CHILD] pid: 2 is running!, global_variable: 11
[U-CHILD] pid: 2 is running!, global_variable: 12
[U-CHILD] pid: 2 is running!, global_variable: 13

SWITCH TO [pid=1 counter=1 priority=37]
[U-PARENT] pid: 1 is running!, global_variable: 14
[U-PARENT] pid: 1 is running!, global_variable: 15
[U-PARENT] pid: 1 is running!, global_variable: 16

SWITCH TO [pid=2 counter=4 priority=37]
[U-CHILD] pid: 2 is running!, global_variable: 14
[U-CHILD] pid: 2 is running!, global_variable: 15
[U-CHILD] pid: 2 is running!, global_variable: 16
[U-CHILD] pid: 2 is running!, global_variable: 17
[U-CHILD] pid: 2 is running!, global_variable: 18
[U-CHILD] pid: 2 is running!, global_variable: 19
[U-CHILD] pid: 2 is running!, global_variable: 20
[U-CHILD] pid: 2 is running!, global_variable: 21
[U-CHILD] pid: 2 is running!, global_variable: 22
[U-CHILD] pid: 2 is running!, global_variable: 23
[U-CHILD] pid: 2 is running!, global_variable: 24
[U-CHILD] pid: 2 is running!, global_variable: 25
[U-CHILD] pid: 2 is running!, global_variable: 26
[U-CHILD] pid: 2 is running!, global_variable: 27
[U-CHILD] pid: 2 is running!, global_variable: 28
[U-CHILD] pid: 2 is running!, global_variable: 29
[U-CHILD] pid: 2 is running!, global_variable: 30
[U-CHILD] pid: 2 is running!, global_variable: 31
[U-CHILD] pid: 2 is running!, global_variable: 32
[U-CHILD] pid: 2 is running!, global_variable: 33
[U-CHILD] pid: 2 is running!, global_variable: 34
[U-CHILD] pid: 2 is running!, global_variable: 35

SWITCH TO [pid=1 counter=10 priority=37]
[U-PARENT] pid: 1 is running!, global_variable: 17
[U-PARENT] pid: 1 is running!, global_variable: 18
[U-PARENT] pid: 1 is running!, global_variable: 19
[U-PARENT] pid: 1 is running!, global_variable: 20
[U-PARENT] pid: 1 is running!, global_variable: 21
```

```
...proc_init done!
2023[S-Mode] Hello RISC-V

SWITCH TO [pid=1 counter=4 priority=37]
[S] Supervisor Page Fault, cause: 0x000000000000000c, stval: 0x00000000000100e8, sepc: 0x00000000000100e8
[S] Supervisor Page Fault, cause: 0x000000000000000f, stval: 0x0000003ffffffff8, sepc: 0x0000000000010158
[S] Supervisor Page Fault, cause: 0x000000000000000d, stval: 0x0000000000011930, sepc: 0x0000000000010174
[U] pid: 1 is running!, global_variable: 0
FORK [pid=2 counter=4 priority=37]
[U] pid: 1 is running!, global_variable: 1
FORK [pid=3 counter=4 priority=37]
[U] pid: 1 is running!, global_variable: 2
[U] pid: 1 is running!, global_variable: 3
[U] pid: 1 is running!, global_variable: 4
[U] pid: 1 is running!, global_variable: 5
[U] pid: 1 is running!, global_variable: 6
[U] pid: 1 is running!, global_variable: 7
[U] pid: 1 is running!, global_variable: 8
[U] pid: 1 is running!, global_variable: 9
[U] pid: 1 is running!, global_variable: 10
[U] pid: 1 is running!, global_variable: 11
[U] pid: 1 is running!, global_variable: 12

SWITCH TO [pid=2 counter=4 priority=37]
[U] pid: 2 is running!, global_variable: 1
FORK [pid=4 counter=4 priority=37]
[U] pid: 2 is running!, global_variable: 2
[U] pid: 2 is running!, global_variable: 3
[U] pid: 2 is running!, global_variable: 4
[U] pid: 2 is running!, global_variable: 5
[U] pid: 2 is running!, global_variable: 6
[U] pid: 2 is running!, global_variable: 7
[U] pid: 2 is running!, global_variable: 8
[U] pid: 2 is running!, global_variable: 9
[U] pid: 2 is running!, global_variable: 10
[U] pid: 2 is running!, global_variable: 11
[U] pid: 2 is running!, global_variable: 12

SWITCH TO [pid=3 counter=4 priority=37]
[U] pid: 3 is running!, global_variable: 2
[U] pid: 3 is running!, global_variable: 3
[U] pid: 3 is running!, global_variable: 4
[U] pid: 3 is running!, global_variable: 5
[U] pid: 3 is running!, global_variable: 6
[U] pid: 3 is running!, global_variable: 7
[U] pid: 3 is running!, global_variable: 8
[U] pid: 3 is running!, global_variable: 9
[U] pid: 3 is running!, global_variable: 10
[U] pid: 3 is running!, global_variable: 11
[U] pid: 3 is running!, global_variable: 12

SWITCH TO [pid=4 counter=4 priority=37]
[U] pid: 4 is running!, global_variable: 2
[U] pid: 4 is running!, global_variable: 3
[U] pid: 4 is running!, global_variable: 4
[U] pid: 4 is running!, global_variable: 5
[U] pid: 4 is running!, global_variable: 6
[U] pid: 4 is running!, global_variable: 7
[U] pid: 4 is running!, global_variable: 8
[U] pid: 4 is running!, global_variable: 9
[U] pid: 4 is running!, global_variable: 10
[U] pid: 4 is running!, global_variable: 11
[U] pid: 4 is running!, global_variable: 12

SWITCH TO [pid=1 counter=1 priority=37]
[U] pid: 1 is running!, global_variable: 13
[U] pid: 1 is running!, global_variable: 14
[U] pid: 1 is running!, global_variable: 15

SWITCH TO [pid=2 counter=4 priority=37]
[U] pid: 2 is running!, global_variable: 13
[U] pid: 2 is running!, global_variable: 14
[U] pid: 2 is running!, global_variable: 15
[U] pid: 2 is running!, global_variable: 16
[U] pid: 2 is running!, global_variable: 17
[U] pid: 2 is running!, global_variable: 18
[U] pid: 2 is running!, global_variable: 19
[U] pid: 2 is running!, global_variable: 20
[U] pid: 2 is running!, global_variable: 21
[U] pid: 2 is running!, global_variable: 22
[U] pid: 2 is running!, global_variable: 23

SWITCH TO [pid=4 counter=4 priority=37]
[U] pid: 4 is running!, global_variable: 13
[U] pid: 4 is running!, global_variable: 14
[U] pid: 4 is running!, global_variable: 15
[U] pid: 4 is running!, global_variable: 16
[U] pid: 4 is running!, global_variable: 17
[U] pid: 4 is running!, global_variable: 18
[U] pid: 4 is running!, global_variable: 19
[U] pid: 4 is running!, global_variable: 20
[U] pid: 4 is running!, global_variable: 21
[U] pid: 4 is running!, global_variable: 22
[U] pid: 4 is running!, global_variable: 23

SWITCH TO [pid=3 counter=10 priority=37]
[U] pid: 3 is running!, global_variable: 13
[U] pid: 3 is running!, global_variable: 14
[U] pid: 3 is running!, global_variable: 15
[U] pid: 3 is running!, global_variable: 16
[U] pid: 3 is running!, global_variable: 17
[U] pid: 3 is running!, global_variable: 18
[U] pid: 3 is running!, global_variable: 19
[U] pid: 3 is running!, global_variable: 20
[U] pid: 3 is running!, global_variable: 21
[U] pid: 3 is running!, global_variable: 22
[U] pid: 3 is running!, global_variable: 23
[U] pid: 3 is running!, global_variable: 24
[U] pid: 3 is running!, global_variable: 25
[U] pid: 3 is running!, global_variable: 26
[U] pid: 3 is running!, global_variable: 27
[U] pid: 3 is running!, global_variable: 28
[U] pid: 3 is running!, global_variable: 29
[U] pid: 3 is running!, global_variable: 30
[U] pid: 3 is running!, global_variable: 31
[U] pid: 3 is running!, global_variable: 32
QEMU: Terminated
```

三、讨论和心得

在这里，想高度赞扬 lab6 指导书，感觉这一篇写得非常清楚，也很有对话感，注释指导得也很清晰。

本次实验基本没有遇到什么问题，除了在根据父线程建立映射那里稍有卡顿外，其余部分跟着指导走非常顺利。特别想提到的是，在运行时我发现每产生一个新子线程时会报一次 Instruction Page Fault——经检查，是在建立映射时权限设置错误。

四、思考题

1. 参考 task_init 创建一个新的 task, 将 parent task 的整个页复制到新创建的 task_struct 页上, 这一步复制了哪些东西?

结合 task_struct 的结构，这一步应当复制了以下所有成员。同时，还有 task 高地址的进程栈的内容也在这一步被复制。

```c
// arch/riscv/kernel/proc.h
struct vm_area_struct {
    uint64_t vm_start;          /* VMA 对应的用户态虚拟地址的开始   */
    uint64_t vm_end;            /* VMA 对应的用户态虚拟地址的结束   */
    uint64_t vm_flags;          /* VMA 对应的 flags */

    uint64_t vm_content_offset_in_file;
    uint64_t vm_content_size_in_file;
};
struct thread_struct {
    uint64 ra;
    uint64 sp;
    uint64 s[12];
    uint64 sepc, sstatus, sscratch;
};
struct task_struct {
    uint64 state;    // 线程状态
    uint64 counter;  // 运行剩余时间
    uint64 priority; // 运行优先级 1最低 10最高
    uint64 pid;      // 线程id

    struct thread_struct thread;
    pagetable_t pgd;

    uint64_t vma_cnt;
    struct vm_area_struct vmas[0];
};
```

2. 将 thread.ra 设置为 __ret_from_fork, 并正确设置 thread.sp。仔细想想，这个应该设置成什么值?可以根据 child task 的返回路径来倒推。

我们根据 child task 的返回路径，即__switch_to->__ret_from_fork(in _traps)->user program，加上__switch_to 中并没有使用 sp 作为栈指针进行操作，而是在__ret_from_fork 中发生了第一次将 sp 作为栈指针进行栈操作。因此，需要将

thread.sp 设置为在__ret_from_fork 中预期的栈顶指针。

容易得到，此时 sp 应当指向 regs 的顶端，即 thread.sp 应当指向子线程的 regs 的顶端，而根据子线程和父线程的 regs 在其 task_struct 中的偏移应当相同，我们可以通过如下方式计算得到 thread.sp 的值：

```
struct pt_regs* child_regs = (struct pt_regs*)((uint64)task[new_task_index] + ((uint64)regs -PGROUNDDOWN((uint64)regs)));
task[new_task_index]->thread.sp = (uint64)child_regs;
```

3. 利用参数 regs 来计算出 child task 的对应的 pt_regs 的地址，并将其中的 a0, sp, sepc 设置成正确的值。为什么还要设置 sp?

在稍后的__ret_from_fork 中，我们会从 regs 中恢复所有寄存器的值。上题中设置 sp，是为了在__ret_from_fork 中能够在正确的栈上进行操作，本题中设置 sp 是为了在__ret_from_fork 恢复寄存器值后 sp 仍能保持正确。

如下图所示，在_traps 中，trap_handler 调用前后也发生了一次 sp 的存取。如果只将上题的 sp 做设置而不设置本题中的 sp，从栈上恢复的仍然是从父线程中复制过来的值。

```
# arch/riscv/kernel/entry.S
_traps:
    ...
_traps_start:
    # 1. save 32 registers and sepc to stack
    addi sp, sp, -8*37
    ...
    sd x2, 16(sp)
    ...
    call trap_handler

    .global __ret_from_fork
__ret_from_fork:
    ...
    ld x2, 16(sp)
    addi sp, sp, 8*37
```