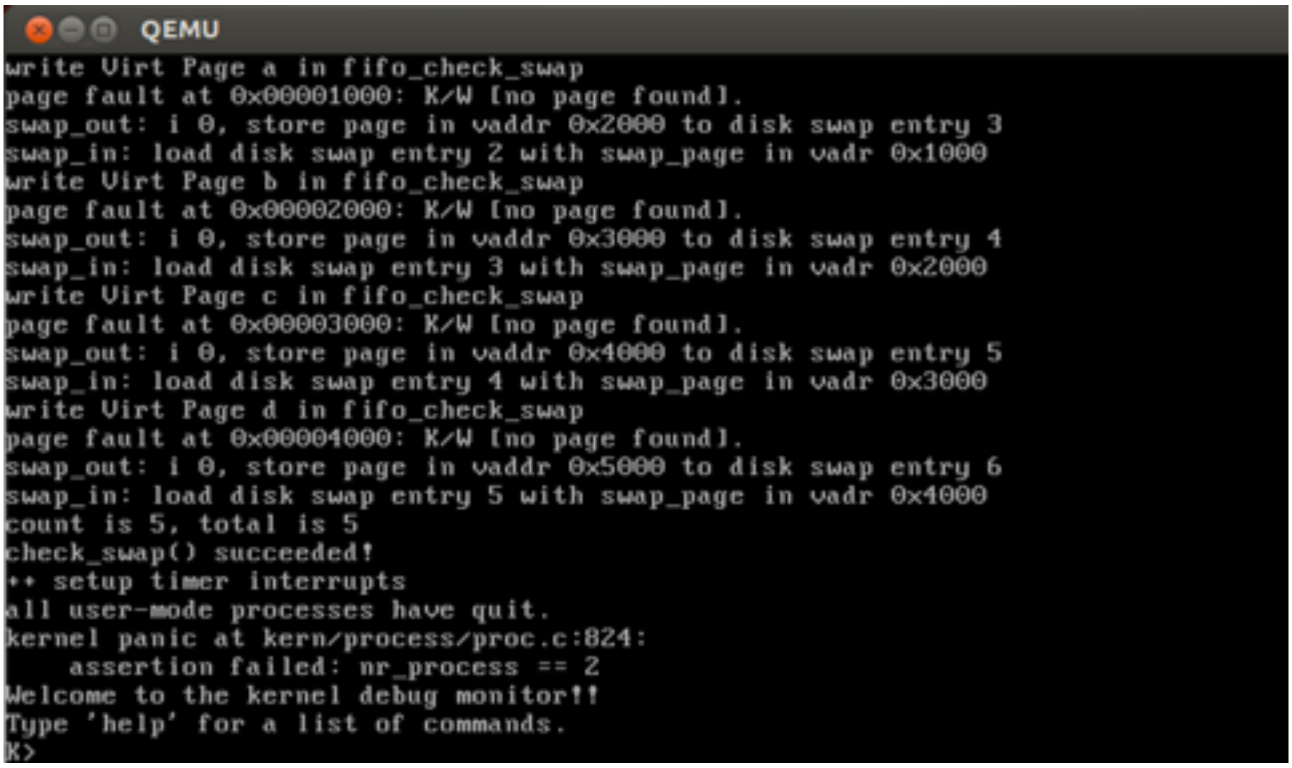
0.



```
write Virt Page a in fifo_check_swap
page fault at 0x00001000: K/W [no page found].
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
write Virt Page b in fifo_check_swap
page fault at 0x00002000: K/W [no page found].
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 3 with swap_page in vadr 0x2000
write Virt Page c in fifo_check_swap
page fault at 0x00003000: K/W [no page found].
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page d in fifo_check_swap
page fault at 0x00004000: K/W [no page found].
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
count is 5, total is 5
check_swap() succeeded!
++ setup timer interrupts
all user-mode processes have quit.
kernel panic at kern/process/proc.c:824:
    assertion failed: nr_process == 2
Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

1、2.
首先填充load_icode，发现我们只是需要填充其中的很小一部分:
    * should set tf_cs,tf_ds,tf_es,tf_ss,tf_esp,tf_eip,tf_eflags
    * NOTICE: If we set trapframe correctly, then the user level process can return to USER MODE from kernel. So
    *   tf_cs should be USER_CS segment (see memlayout.h)
    *   tf_ds=tf_es=tf_ss should be USER_DS segment
    *   tf_esp should be the top addr of user stack (USTACKTOP)
    *   tf_eip should be the entry point of this binary program (elf->e_entry)
    *   tf_eflags should be set to enable computer to produce Interrupt
类似于lab1 challenge中设置tf一样，把这里的tf中的各项值设成注释中所写即可。

然后填充copy_range，这个函数比较短，可以发现，该函数是将start到end间的每个页，依次复制一份，并加入到新的进程的页表中，再看我们需要填充的部分，首先是通过page2kva获取kernel virtual address，从而调用memcpy将原来每个页上的内容copy到新的页中，最后再将新的页插入到新的进程的页表。

至此，看似已经填充好了，但是运行的时候，会发现还是和最开始一样，再观察会发现，在lab5中，还需要修改lab1-lab4的部分代码。

首先调用 grep -r LAB5 . 查看LAB5中需要修改的地方:
```
./kern/mm/pmm.c:       /* LAB5:EXERCISE2 YOUR CODE
./kern/mm/vmm.c:    * LAB5 CHALLENGE ( the implmentation Copy on Write)
./kern/mm/vmm.c:                              //(4) [NOTICE]: you myabe need to update your lab3's implementation for LAB5's normal execution.
./kern/process/proc.c:     //LAB5 YOUR CODE : (update LAB4 steps)
./kern/process/proc.c:     * below fields(add in LAB5) in proc_struct need to be initialized
./kern/process/proc.c: //LAB5 YOUR CODE : (update LAB4 steps)
./kern/process/proc.c:    /* LAB5:EXERCISE1 YOUR CODE
./kern/trap/trap.c:     /* LAB5 YOUR CODE */
./kern/trap/trap.c:       /* LAB5 YOUR CODE */
```

./obj/kernel.asm: /* LAB5 YOUR CODE */
./obj/kernel.asm: * below fields(add in LAB5) in proc_struct need to be initialized

pmm.c中的就是我们之前改的；
vmm.c是LAB5 challenge要改的；

proc.c中

```
    //LAB5 YOUR CODE : (update LAB4 steps)
   /*
    * below fields(add in LAB5) in proc_struct need to be initialized
    *       uint32_t wait_state;                        // waiting state
    *       struct proc_struct *cptr, *yptr, *optr;     // relations between processes
           */
```

发现是要初始化proc_struct中的部分变量，由于之前是使用的memset来初始化的，故此处无须修改；

```
   //LAB5 YOUR CODE : (update LAB4 steps)
  /* Some Functions
   *    set_links:  set the relation links of process.  ALSO SEE: remove_links:  lean the relation links of process
   *    -------------------
          *    update step 1: set child proc's parent to current process, make sure current process's wait_state is 0
          *    update step 5: insert proc_struct into hash_list && proc_list, set the relation links of process
   */
```

设置进程的父进程，并assert确保current->wait_state为0
然后set_links设置新进程间的关系，查看下实现会发现参数应该就是新进程。

trap.c

```
    /* LAB5 YOUR CODE */
    //you should update your lab1 code (just add ONE or TWO lines of code), let user app to use syscall to get the service of ucore
    //so you should setup the syscall interrupt gate in here
```

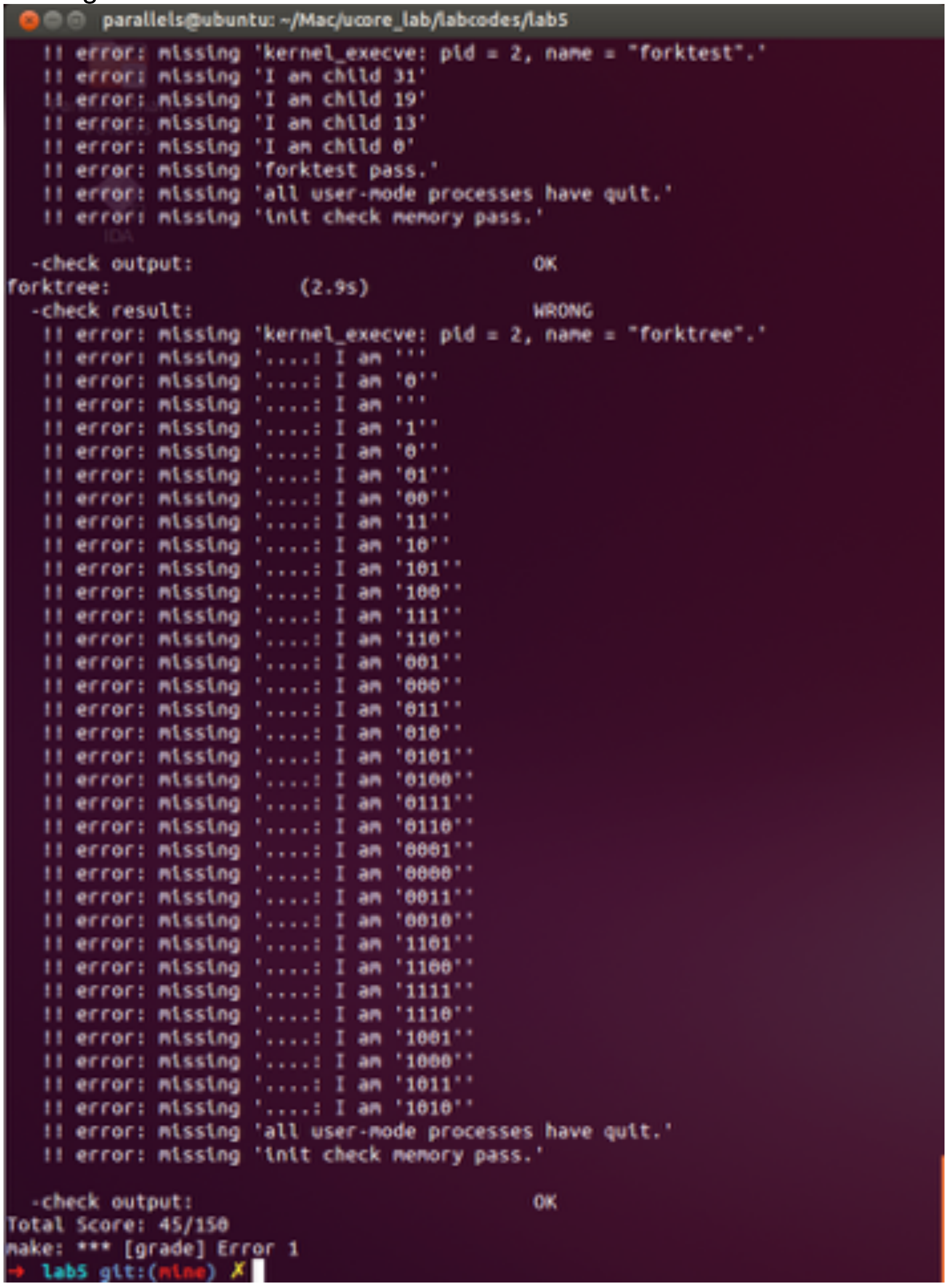
在idt_init的时候，要为syscall设置用户调用权限，在LAB1 challenge的时候此处已经设定。

```
       /* LAB5 YOUR CODE */
       /* you should upate you lab1 code (just add ONE or TWO lines of code):
        *    Every TICK_NUM cycle, you should set current process's current->need_resched = 1
        */
```

为了使得用户进程能够由操作系统调度，按照要求设置current->need_resched即可。

此时运行的话，结果如下：

make grade:

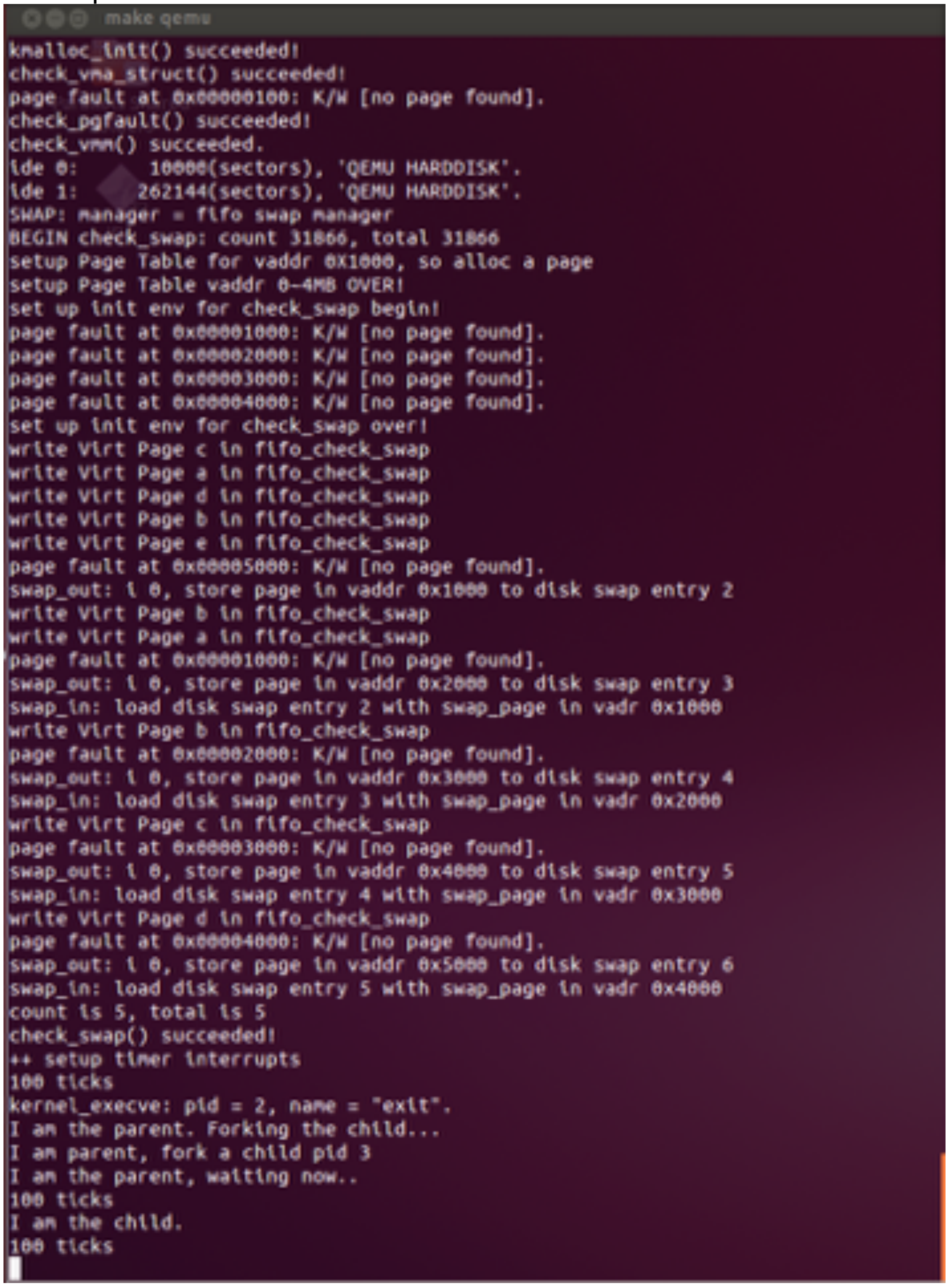


```
parallels@ubuntu: ~/Mac/ucore_lab/labcodes/lab5
  !! error: missing 'kernel_execve: pid = 2, name = "forktest".'
  !! error: missing 'I am child 31'
  !! error: missing 'I am child 19'
  !! error: missing 'I am child 13'
  !! error: missing 'I am child 0'
  !! error: missing 'forktest pass.'
  !! error: missing 'all user-mode processes have quit.'
  !! error: missing 'init check memory pass.'

 -check output:                           OK
forktree:               (2.9s)
 -check result:                           WRONG
  !! error: missing 'kernel_execve: pid = 2, name = "forktree".'
  !! error: missing '....: I am '''
  !! error: missing '....: I am '0''
  !! error: missing '....: I am '''
  !! error: missing '....: I am '1''
  !! error: missing '....: I am '0''
  !! error: missing '....: I am '01''
  !! error: missing '....: I am '00''
  !! error: missing '....: I am '11''
  !! error: missing '....: I am '10''
  !! error: missing '....: I am '101''
  !! error: missing '....: I am '100''
  !! error: missing '....: I am '111''
  !! error: missing '....: I am '110''
  !! error: missing '....: I am '001''
  !! error: missing '....: I am '000''
  !! error: missing '....: I am '011''
  !! error: missing '....: I am '010''
  !! error: missing '....: I am '0101''
  !! error: missing '....: I am '0100''
  !! error: missing '....: I am '0111''
  !! error: missing '....: I am '0110''
  !! error: missing '....: I am '0001''
  !! error: missing '....: I am '0000''
  !! error: missing '....: I am '0011''
  !! error: missing '....: I am '0010''
  !! error: missing '....: I am '1101''
  !! error: missing '....: I am '1100''
  !! error: missing '....: I am '1111''
  !! error: missing '....: I am '1110''
  !! error: missing '....: I am '1001''
  !! error: missing '....: I am '1000''
  !! error: missing '....: I am '1011''
  !! error: missing '....: I am '1010''
  !! error: missing 'all user-mode processes have quit.'
  !! error: missing 'init check memory pass.'

 -check output:                           OK
Total Score: 45/150
make: *** [grade] Error 1
➜ lab5 git:(mine) ✗
```

make qemu:

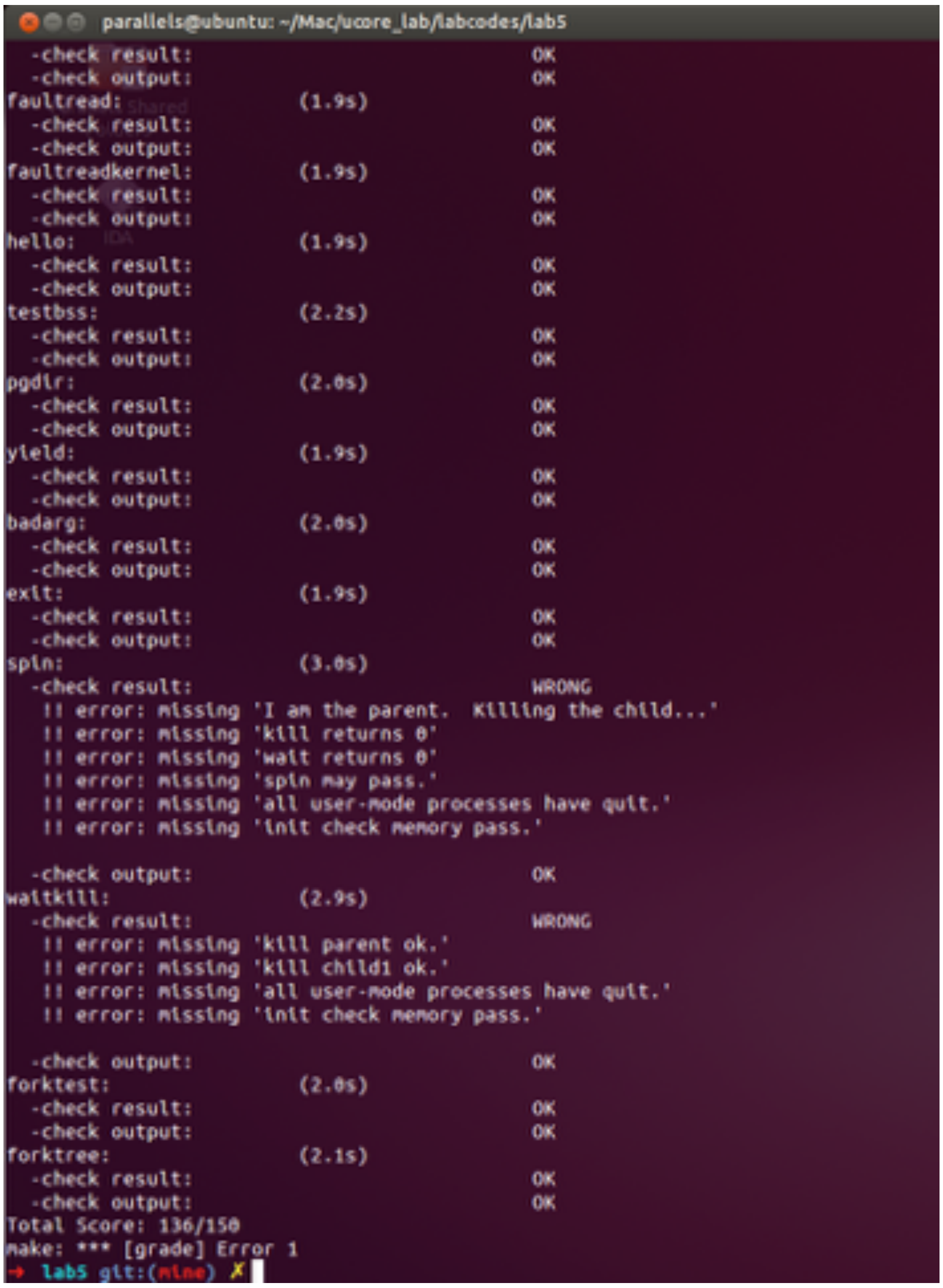


```
kmalloc_init() succeeded!
check_vma_struct() succeeded!
page fault at 0x00000100: K/W [no page found].
check_pgfault() succeeded!
check_vmm() succeeded.
ide 0:      10000(sectors), 'QEMU HARDDISK'.
ide 1:     262144(sectors), 'QEMU HARDDISK'.
SWAP: manager = fifo swap manager
BEGIN check_swap: count 31866, total 31866
setup Page Table for vaddr 0X1000, so alloc a page
setup Page Table vaddr 0~4MB OVER!
set up init env for check_swap begin!
page fault at 0x00001000: K/W [no page found].
page fault at 0x00002000: K/W [no page found].
page fault at 0x00003000: K/W [no page found].
page fault at 0x00004000: K/W [no page found].
set up init env for check_swap over!
write Virt Page c in fifo_check_swap
write Virt Page a in fifo_check_swap
write Virt Page d in fifo_check_swap
write Virt Page b in fifo_check_swap
write Virt Page e in fifo_check_swap
page fault at 0x00005000: K/W [no page found].
swap_out: i 0, store page in vaddr 0x1000 to disk swap entry 2
write Virt Page b in fifo_check_swap
write Virt Page a in fifo_check_swap
page fault at 0x00001000: K/W [no page found].
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
write Virt Page b in fifo_check_swap
page fault at 0x00002000: K/W [no page found].
swap_out: i 0, store page in vaddr 0x3000 to disk swap entry 4
swap_in: load disk swap entry 3 with swap_page in vadr 0x2000
write Virt Page c in fifo_check_swap
page fault at 0x00003000: K/W [no page found].
swap_out: i 0, store page in vaddr 0x4000 to disk swap entry 5
swap_in: load disk swap entry 4 with swap_page in vadr 0x3000
write Virt Page d in fifo_check_swap
page fault at 0x00004000: K/W [no page found].
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
count is 5, total is 5
check_swap() succeeded!
++ setup timer interrupts
100 ticks
kernel_execve: pid = 2, name = "exit".
I am the parent. Forking the child...
I am parent, fork a child pid 3
I am the parent, waiting now..
100 ticks
I am the child.
100 ticks
```

然后程序卡住不动了

很明显，结果还是不对，再观察会发现，在set_links中，将proc插入到了proc_list中，且将nr_process加了1，故我们应在do_fork中删除这两句防止重复操作。

再编译运行make grade：

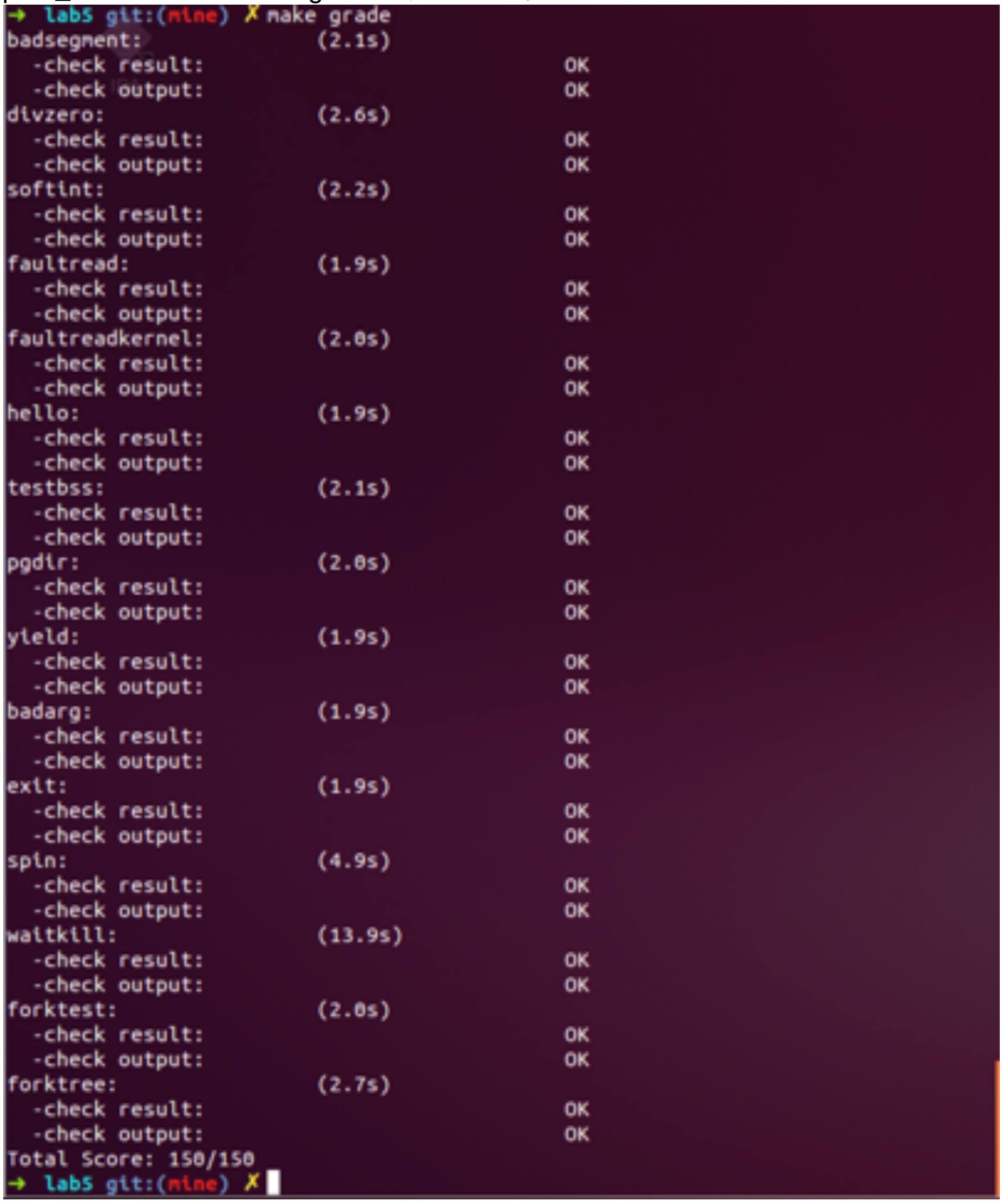

parallels@ubuntu: ~/Mac/ucore_lab/labcodes/lab5

```
  -check result:                            OK
  -check output:                            OK
faultread:               (1.9s)
  -check result:                            OK
  -check output:                            OK
faultreadkernel:         (1.9s)
  -check result:                            OK
  -check output:                            OK
hello:                   (1.9s)
  -check result:                            OK
  -check output:                            OK
testbss:                 (2.2s)
  -check result:                            OK
  -check output:                            OK
pgdir:                   (2.0s)
  -check result:                            OK
  -check output:                            OK
yield:                   (1.9s)
  -check result:                            OK
  -check output:                            OK
badarg:                  (2.0s)
  -check result:                            OK
  -check output:                            OK
exit:                    (1.9s)
  -check result:                            OK
  -check output:                            OK
spin:                    (3.0s)
  -check result:                            WRONG
   !! error: missing 'I am the parent.  Killing the child...'
   !! error: missing 'kill returns 0'
   !! error: missing 'wait returns 0'
   !! error: missing 'spin may pass.'
   !! error: missing 'all user-mode processes have quit.'
   !! error: missing 'init check memory pass.'

  -check output:                            OK
waitkill:                (2.9s)
  -check result:                            WRONG
   !! error: missing 'kill parent ok.'
   !! error: missing 'kill child1 ok.'
   !! error: missing 'all user-mode processes have quit.'
   !! error: missing 'init check memory pass.'

  -check output:                            OK
forktest:                (2.0s)
  -check result:                            OK
  -check output:                            OK
forktree:                (2.1s)
  -check result:                            OK
  -check output:                            OK
Total Score: 136/150
make: *** [grade] Error 1
➜  lab5 git:(mine) ✗
```

查看grade.sh发现，check的时候没有考虑print_ticks的输出，故注释掉trap中lab1时添加的print_ticks，然后再make grade可得正确结果：



```
➜ lab5 git:(mine) ✗ make grade
badsegment:              (2.1s)
  -check result:                           OK
  -check output:                           OK
divzero:                 (2.6s)
  -check result:                           OK
  -check output:                           OK
softint:                 (2.2s)
  -check result:                           OK
  -check output:                           OK
faultread:               (1.9s)
  -check result:                           OK
  -check output:                           OK
faultreadkernel:         (2.0s)
  -check result:                           OK
  -check output:                           OK
hello:                   (1.9s)
  -check result:                           OK
  -check output:                           OK
testbss:                 (2.1s)
  -check result:                           OK
  -check output:                           OK
pgdir:                   (2.0s)
  -check result:                           OK
  -check output:                           OK
yield:                   (1.9s)
  -check result:                           OK
  -check output:                           OK
badarg:                  (1.9s)
  -check result:                           OK
  -check output:                           OK
exit:                    (1.9s)
  -check result:                           OK
  -check output:                           OK
spin:                    (4.9s)
  -check result:                           OK
  -check output:                           OK
waitkill:                (13.9s)
  -check result:                           OK
  -check output:                           OK
forktest:                (2.0s)
  -check result:                           OK
  -check output:                           OK
forktree:                (2.7s)
  -check result:                           OK
  -check output:                           OK
Total Score: 150/150
➜ lab5 git:(mine) ✗
```

3.
fork最终调用到之前实验实现的do_fork函数，从而完成子进程的创建，资源的分配；

exec最终调用到do_execve函数，do_execve函数对于非内核进程首先要切换到内核空间，然后exit_mmap、put_pgdir和mm_destroy来回收当前进程在用户空间的资源，完成后再通过load_icode函数将新程序加载到用户空间，分配资源，设置各指针等，完成进程的改变。

wait最终调用到do_wait函数，do_wait函数通过一个循环，不断的查找子进程中状态为PROC_ZOMBIE的，若找到则结束循环并清理子进程剩余的无法由其自身清理的资源，每一轮循环中若未找到则会先尝试调用schedule让出CPU。

exit最终调用到do_exit函数，do_exit对于非内核进程也是先切换到内核空间，然后和do_execve一样回收当前进程各种资源，接着设置进程状态为PROC_ZOMBIE，最后通过唤醒父进程来清理该进程其余未能回收的资源，并且对于其子进程需要转给initproc。