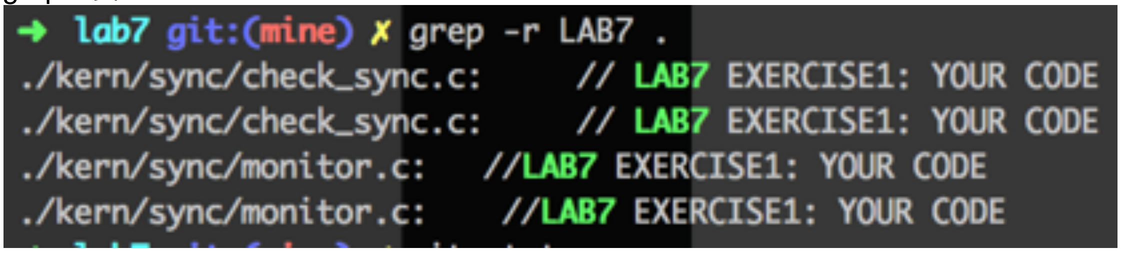
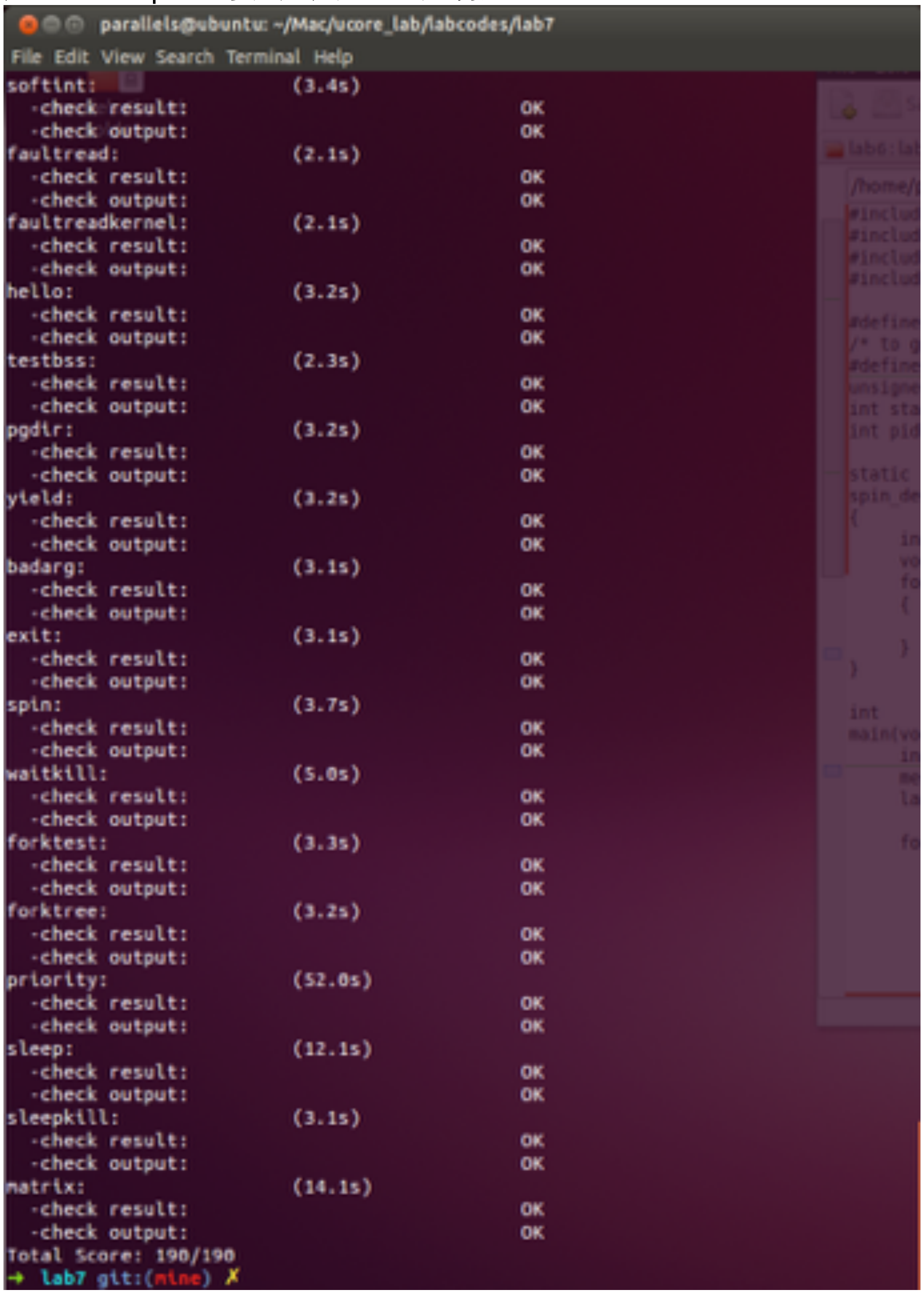
0.
首先将LAB1-6的代码填充（包括整个default_sched要替换，priority.c的MAX_TIME要改大），然后grep查找LAB7：



```
➜ lab7 git:(mine) ✗ grep -r LAB7 .
./kern/sync/check_sync.c:       // LAB7 EXERCISE1: YOUR CODE
./kern/sync/check_sync.c:       // LAB7 EXERCISE1: YOUR CODE
./kern/sync/monitor.c:     //LAB7 EXERCISE1: YOUR CODE
./kern/sync/monitor.c:      //LAB7 EXERCISE1: YOUR CODE
```

发现没有需要修改的地方
然后make qemu可以发现，已经是满分：



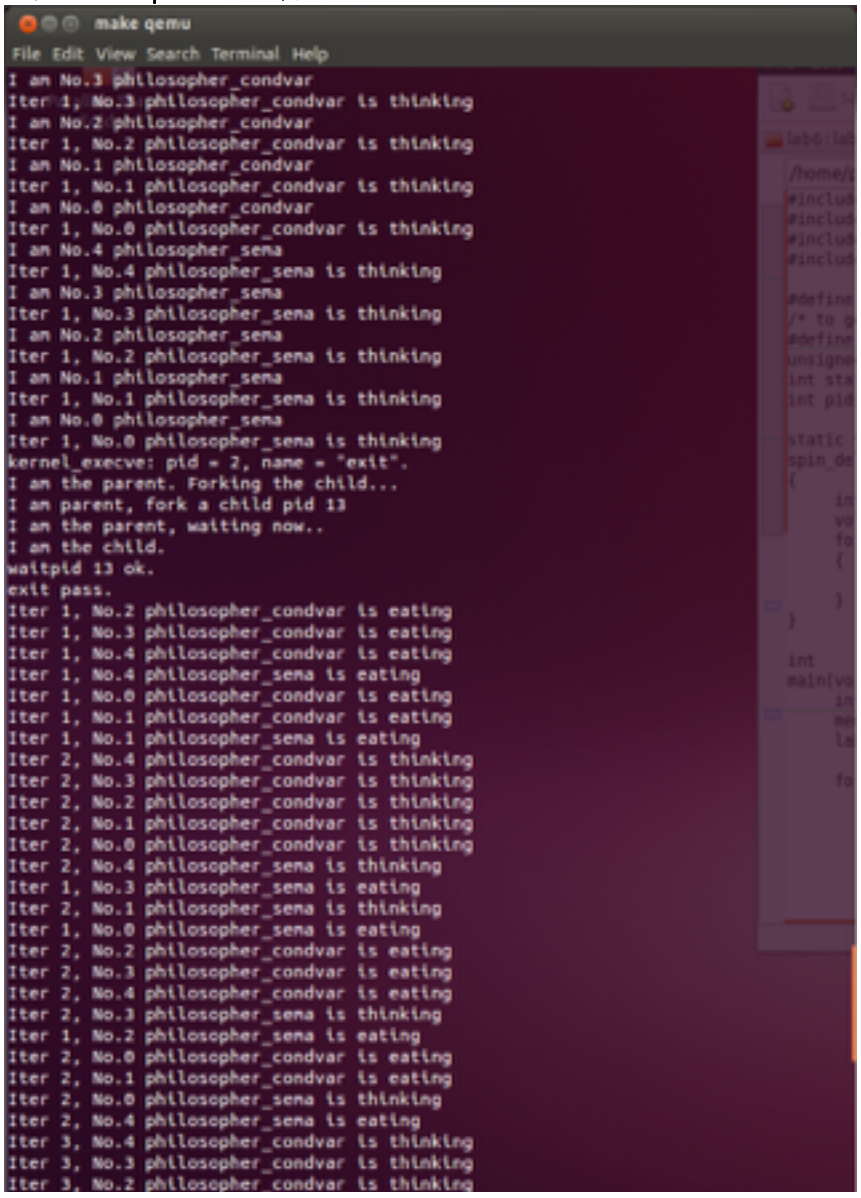
parallels@ubuntu: ~/Mac/ucore_lab/labcodes/lab7
File Edit View Search Terminal Help

```
softint:                    (3.4s)
  -check result:                              OK
  -check output:                              OK
faultread:                  (2.1s)
  -check result:                              OK
  -check output:                              OK
faultreadkernel:            (2.1s)
  -check result:                              OK
  -check output:                              OK
hello:                      (3.2s)
  -check result:                              OK
  -check output:                              OK
testbss:                    (2.3s)
  -check result:                              OK
  -check output:                              OK
pgdir:                      (3.2s)
  -check result:                              OK
  -check output:                              OK
yield:                      (3.2s)
  -check result:                              OK
  -check output:                              OK
badarg:                     (3.1s)
  -check result:                              OK
  -check output:                              OK
exit:                       (3.1s)
  -check result:                              OK
  -check output:                              OK
spin:                       (3.7s)
  -check result:                              OK
  -check output:                              OK
waitkill:                   (5.0s)
  -check result:                              OK
  -check output:                              OK
forktest:                   (3.3s)
  -check result:                              OK
  -check output:                              OK
forktree:                   (3.2s)
  -check result:                              OK
  -check output:                              OK
priority:                   (52.0s)
  -check result:                              OK
  -check output:                              OK
sleep:                      (12.1s)
  -check result:                              OK
  -check output:                              OK
sleepkill:                  (3.1s)
  -check result:                              OK
  -check output:                              OK
matrix:                     (14.1s)
  -check result:                              OK
  -check output:                              OK
Total Score: 190/190
➜ lab7 git:(mine) ✗
```

1.
对比可以发现，lab7相对于lab6，改动的文件不少。
首先是vmm.[ch]，mm_struct中的mm_lock被替换成mm_sem，即将原来的锁替换成了使用信号量来加锁；

接着是proc.[ch]，在系统创建的第二个内核进程中，创建完用户进程后，调用了check_sync来检验哲学家问题。然后增加了do_sleep函数，用来使当前进程睡眠一定时间，函数中将当前进程设为SLEEPING状态，同时增加timer来达到定时的作用，最后调用schedule来放弃CPU；
然后sched.c中max time slice被减小到了20；
接着syscall.c中增加了sys_sleep这个系统调用，通过调用之前的do_sleep来实现进程sleep；
测试程序中增加了sleep.c，sleepkill.c用来测试新增的sleep函数以及之前的kill函数是否正常工作;
最后，改动最大的就是sync部分：
增加了wait.[ch]，实现了等待队列（wait queue）以及wait结构的各种基本操作，增加wakeup_wait用于唤醒等待队列中一个特定的wait，wakeup_first用于唤醒等待队列中的第一个wait，wakeup_queue用于唤醒整个等待队列中所有的wait，wait_current_set用于使当前进程进入等待队列；
增加了sem.[ch]，实现了信号量，up和down分别对应于信号量的signal和wait，分别调用内部__up和__down；
增加了monitor.[ch]，利用前面实现的信号量，实现了管程和条件变量（需要后面自己实现），增加cond_signal和cond_wait函数；
check_sync.c实现了哲学家问题，分别以信号量和条件变量的方式实现，每个哲学家都是在不断的循环进行思考、拿起筷子（叉子）、放下筷子（叉子），从而根据最后实际的进餐情况可以知道是信号量（或条件变量）是否确实起作用了。

2.
直接make qemu可以看到：

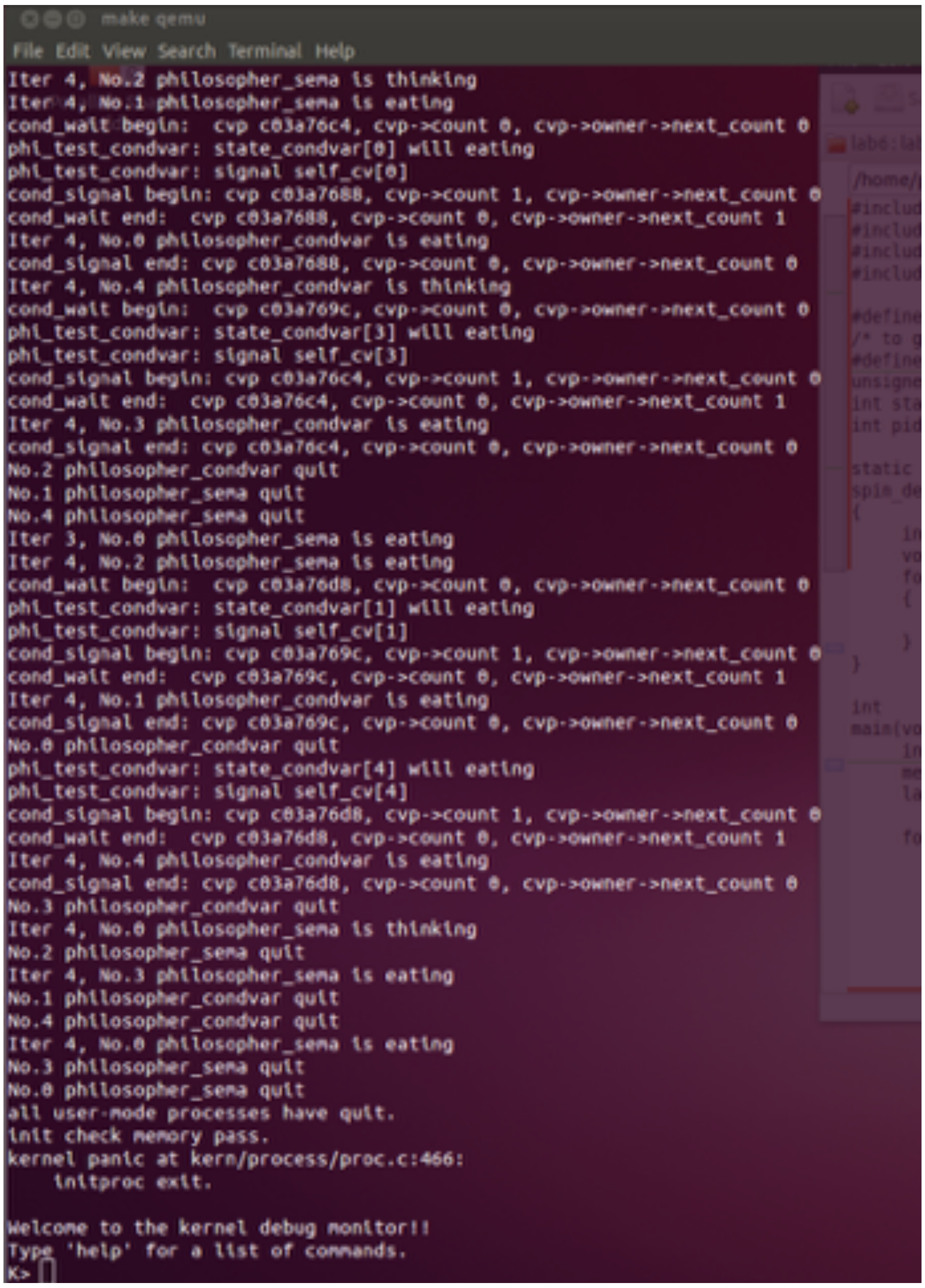


```
I am No.3 philosopher_condvar
Iter 1, No.3 philosopher_condvar is thinking
I am No.2 philosopher_condvar
Iter 1, No.2 philosopher_condvar is thinking
I am No.1 philosopher_condvar
Iter 1, No.1 philosopher_condvar is thinking
I am No.0 philosopher_condvar
Iter 1, No.0 philosopher_condvar is thinking
I am No.4 philosopher_sema
Iter 1, No.4 philosopher_sema is thinking
I am No.3 philosopher_sema
Iter 1, No.3 philosopher_sema is thinking
I am No.2 philosopher_sema
Iter 1, No.2 philosopher_sema is thinking
I am No.1 philosopher_sema
Iter 1, No.1 philosopher_sema is thinking
I am No.0 philosopher_sema
Iter 1, No.0 philosopher_sema is thinking
kernel_execve: pid = 2, name = "exit".
I am the parent. Forking the child...
I am parent, fork a child pid 13
I am the parent, waiting now..
I am the child.
waitpid 13 ok.
exit pass.
Iter 1, No.2 philosopher_condvar is eating
Iter 1, No.3 philosopher_condvar is eating
Iter 1, No.4 philosopher_condvar is eating
Iter 1, No.4 philosopher_sema is eating
Iter 1, No.0 philosopher_condvar is eating
Iter 1, No.1 philosopher_condvar is eating
Iter 1, No.1 philosopher_sema is eating
Iter 2, No.4 philosopher_condvar is thinking
Iter 2, No.3 philosopher_condvar is thinking
Iter 2, No.2 philosopher_condvar is thinking
Iter 2, No.1 philosopher_condvar is thinking
Iter 2, No.0 philosopher_condvar is thinking
Iter 2, No.4 philosopher_sema is thinking
Iter 1, No.3 philosopher_sema is eating
Iter 2, No.1 philosopher_sema is thinking
Iter 1, No.0 philosopher_sema is eating
Iter 2, No.2 philosopher_condvar is eating
Iter 2, No.3 philosopher_condvar is eating
Iter 2, No.4 philosopher_condvar is eating
Iter 2, No.3 philosopher_sema is thinking
Iter 1, No.2 philosopher_sema is eating
Iter 2, No.0 philosopher_condvar is eating
Iter 2, No.1 philosopher_condvar is eating
Iter 2, No.0 philosopher_sema is thinking
Iter 2, No.4 philosopher_sema is eating
Iter 3, No.4 philosopher_condvar is thinking
Iter 3, No.3 philosopher_condvar is thinking
Iter 3, No.2 philosopher_condvar is thinking
```

发现此时由于没有限制，所有的哲学家都是随时想吃就吃，并没有等待，几乎都是同一时间在吃。

然后首先修改monitor.c，基本和注释一致，然后signal换成up，wait换成down，注意别弄混了，参数为指针。
然后修改check_sync.c，state_condvar[i]表示第i个哲学家当前的状态，条件变量cv[i]表示第i个哲学家等待进食，故若wait到cv[i]，则pick up成功，可以进食了，函数phi_test_condvar用来尝试让第i个哲学家进食。具体按照注释或文档中代码实现即可。

最后完成后运行make qemu，可得：



```
Iter 4, No.2 philosopher_sema is thinking
Iter 4, No.1 philosopher_sema is eating
cond_wait begin:  cvp c03a76c4, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[0] will eating
phi_test_condvar: signal self_cv[0]
cond_signal begin: cvp c03a7688, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp c03a7688, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.0 philosopher_condvar is eating
cond_signal end: cvp c03a7688, cvp->count 0, cvp->owner->next_count 0
Iter 4, No.4 philosopher_condvar is thinking
cond_wait begin:  cvp c03a769c, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[3] will eating
phi_test_condvar: signal self_cv[3]
cond_signal begin: cvp c03a76c4, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp c03a76c4, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.3 philosopher_condvar is eating
cond_signal end: cvp c03a76c4, cvp->count 0, cvp->owner->next_count 0
No.2 philosopher_condvar quit
No.1 philosopher_sema quit
No.4 philosopher_sema quit
Iter 3, No.0 philosopher_sema is eating
Iter 4, No.2 philosopher_sema is eating
cond_wait begin:  cvp c03a76d8, cvp->count 0, cvp->owner->next_count 0
phi_test_condvar: state_condvar[1] will eating
phi_test_condvar: signal self_cv[1]
cond_signal begin: cvp c03a769c, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp c03a769c, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.1 philosopher_condvar is eating
cond_signal end: cvp c03a769c, cvp->count 0, cvp->owner->next_count 0
No.0 philosopher_condvar quit
phi_test_condvar: state_condvar[4] will eating
phi_test_condvar: signal self_cv[4]
cond_signal begin: cvp c03a76d8, cvp->count 1, cvp->owner->next_count 0
cond_wait end:  cvp c03a76d8, cvp->count 0, cvp->owner->next_count 1
Iter 4, No.4 philosopher_condvar is eating
cond_signal end: cvp c03a76d8, cvp->count 0, cvp->owner->next_count 0
No.3 philosopher_condvar quit
Iter 4, No.0 philosopher_sema is thinking
No.2 philosopher_sema quit
Iter 4, No.3 philosopher_sema is eating
No.1 philosopher_condvar quit
No.4 philosopher_condvar quit
Iter 4, No.0 philosopher_sema is eating
No.3 philosopher_sema quit
No.0 philosopher_sema quit
all user-mode processes have quit.
init check memory pass.
kernel panic at kern/process/proc.c:466:
    initproc exit.

Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
K>
```

此时可发现很明显有的哲学家实在cond_signal的时候，即其它哲学家进食完之后才能开始进食，仔细观察也可发现没有出现两个相邻的哲学家同时进食了。