实验报告：Pintos 修改 Testcase

课程名称：操作系统	年级：2023 级本科	上机实践成绩：
指导教师：张民	姓名：张梓卫	
上机实践名称：Pintos 修改 Testcase	学号：10235101526	上机实践日期：2024/10/14
上机实践编号：（2）	组号：	上机实践时间：2 学时

目录

一 实验目的

掌握部分命令行参数解析，并且熟练使用 Docker 与 VSCode 进行远程开发。掌握新建一个 test 的方法、理解 pintos 操作系统中的程序入口、函数参数，部分源码以及初步理解文件结构。

本次实验作出修改的代码如下所示：

同时上传到了 Github 之上，仓库地址为：https://github.com/Shichien/ECNU-23-SEI-Homework

请在上传的 PDF 文件中直接点击粉色链接即可。

二 内容与设计思想

使用 Pintos 创建一个 Test case，并且能够使用 pintos 成功运行自己创建的 test，由此知道如何对操作系统的架构进行操作。

三 使用环境

使用 Docker v27.1.1 进行 Pintos 的安装实验，基于 Windows 11 操作系统使用 WSL2。

实验报告使用 LaTeX 进行撰写，使用 VSCode + Vim 编辑器进行文本编辑。

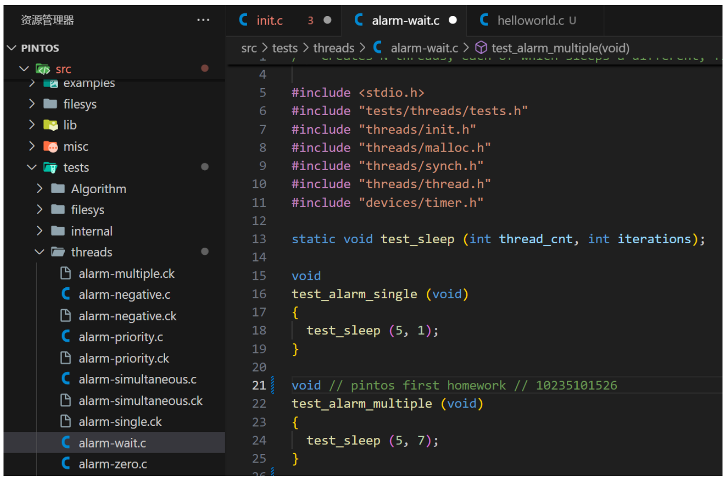
四 实验过程与分析

1 在 VSCode 中安装 Remote Development 插件

在插件商店中搜索 Remote Development，安装 Remote Development 插件。根据 PPT 指引，获取本地中运行的 Docker 容器中的文件配置。

2 查看文件内容，作注释

打开 VSCode，打开 Pintos 项目。并进入 src/test/threads 目录。



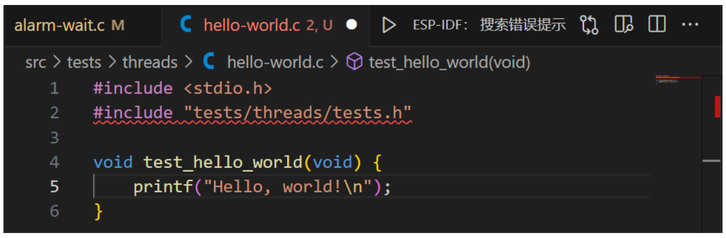
```c
#include <stdio.h>
#include "tests/threads/tests.h"
#include "threads/init.h"
#include "threads/malloc.h"
#include "threads/synch.h"
#include "threads/thread.h"
#include "devices/timer.h"

static void test_sleep (int thread_cnt, int iterations);

void
test_alarm_single (void)
{
  test_sleep (5, 1);
}

void // pintos first homework // 10235101526
test_alarm_multiple (void)
{
  test_sleep (5, 7);
}
```

图 1: 在 VSCode 中打开 Pintos 项目

作好注释，表示这是第二次课程的作业，记录学号以确保本截图来自本人。

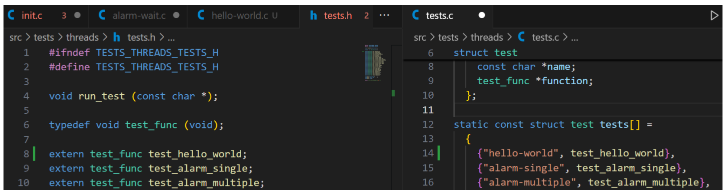
3 添加指定内容

进入 src/test/threads/test.c 与 test.h，新增 hello-world 部分的代码。


alarm-wait.c M
hello-world.c 2, U
ESP-IDF：搜索错误提示
src > tests > threads > hello-world.c > test_hello_world(void)

```c
#include <stdio.h>
#include "tests/threads/tests.h"

void test_hello_world(void) {
    printf("Hello, world!\n");
}
```

图 2: 在 VSCode 中添加 hello-world.c 代码


init.c 3
alarm-wait.c
hello-world.c U
tests.h 2
tests.c
src > tests > threads > tests.h > ...
src > tests > threads > tests.c > ...

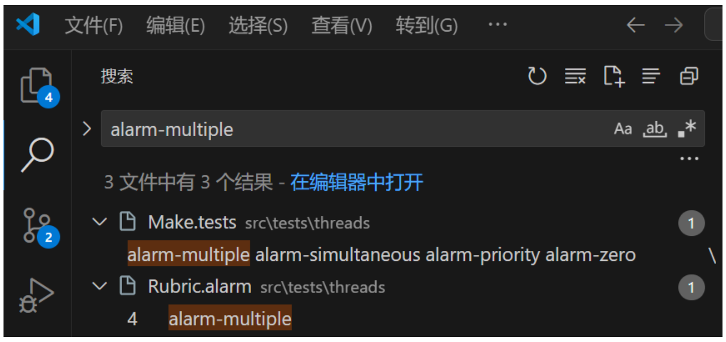
```c
#ifndef TESTS_THREADS_TESTS_H
#define TESTS_THREADS_TESTS_H

void run_test (const char *);

typedef void test_func (void);

extern test_func test_hello_world;
extern test_func test_alarm_single;
extern test_func test_alarm_multiple;
```

```c
struct test
    const char *name;
    test_func *function;
  };

static const struct test tests[] =
  {
    {"hello-world", test_hello_world},
    {"alarm-single", test_alarm_single},
    {"alarm-multiple", test_alarm_multiple},
```

图 3: 添加指定内容

4 全局搜索，查看编译方式

使用 Ctrl + Shift + F 进行全局搜索，因为我们要通过一个测试，而之前我们已经通过了 alarm-multiple 的测试，所以不妨使用全局搜索查看它是怎样运行的。

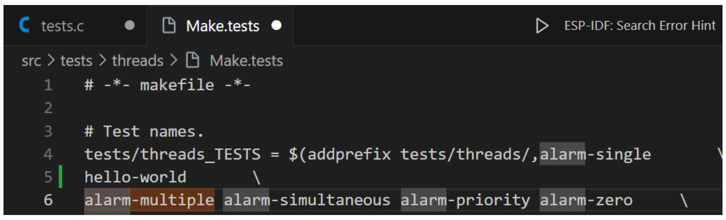


```
alarm-multiple
3 文件中有 3 个结果 - 在编辑器中打开
Make.tests src\tests\threads  1
    alarm-multiple alarm-simultaneous alarm-priority alarm-zero    \
Rubric.alarm src\tests\threads  1
    4 alarm-multiple
```

图 4: 全局搜索，查看编译方式

可以看到，有三个结果，其中一个是我们已经操作过的 test.c 中的内容，接下来，我们应该对 Make.testc 文件进行重点关注。

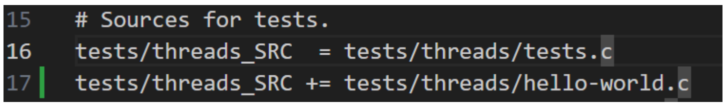
我们增加一个名为”hello-world” 的测试，反斜杠代表不换行。


tests.c
Make.tests
ESP-IDF: Search Error Hint
src > tests > threads > Make.tests

```
# -*- makefile -*-

# Test names.
tests/threads_TESTS = $(addprefix tests/threads/,alarm-single     \
hello-world       \
alarm-multiple alarm-simultaneous alarm-priority alarm-zero     \
```

图 5: Make.testc 文件

在下面的 thread-SRC 中按照相关的格式添加 hello-world 部分的代码。

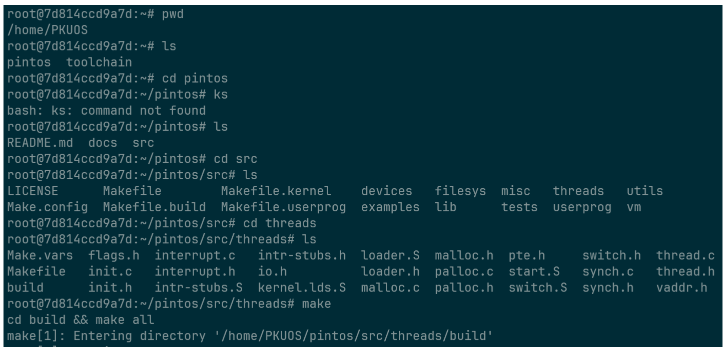


```
# Sources for tests.
tests/threads_SRC  = tests/threads/tests.c
tests/threads_SRC += tests/threads/hello-world.c
```

图 6: Make.testc 文件

5 尝试运行

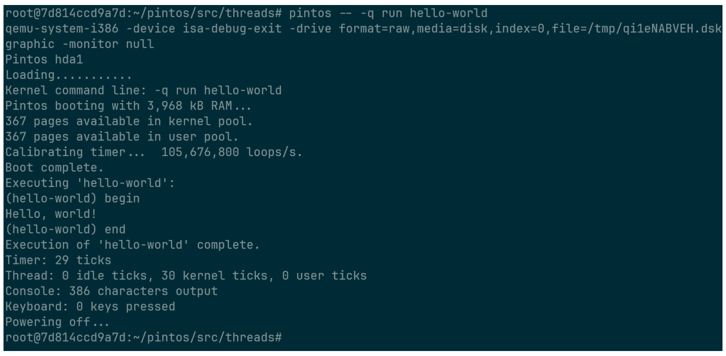
接下来，保存，尝试运行。



```
root@7d814ccd9a7d:~# pwd
/home/PKUOS
root@7d814ccd9a7d:~# ls
pintos  toolchain
root@7d814ccd9a7d:~# cd pintos
root@7d814ccd9a7d:~/pintos# ks
bash: ks: command not found
root@7d814ccd9a7d:~/pintos# ls
README.md  docs  src
root@7d814ccd9a7d:~/pintos# cd src
root@7d814ccd9a7d:~/pintos/src# ls
LICENSE      Makefile        Makefile.kernel    devices   filesys  misc   threads   utils
Make.config  Makefile.build  Makefile.userprog  examples  lib      tests  userprog  vm
root@7d814ccd9a7d:~/pintos/src# cd threads
root@7d814ccd9a7d:~/pintos/src/threads# ls
Make.vars  flags.h  interrupt.c   intr-stubs.h  loader.S  malloc.h  pte.h    switch.h  thread.c
Makefile   init.c   interrupt.h   io.h          loader.h  palloc.c  start.S  synch.c   thread.h
build      init.h   intr-stubs.S  kernel.lds.S  malloc.c  palloc.h  switch.S synch.h   vaddr.h
root@7d814ccd9a7d:~/pintos/src/threads# make
cd build && make all
make[1]: Entering directory '/home/PKUOS/pintos/src/threads/build'
```

图 7: 尝试运行

运行命令：**pintos – -q run hello-world**，结果如下所示：

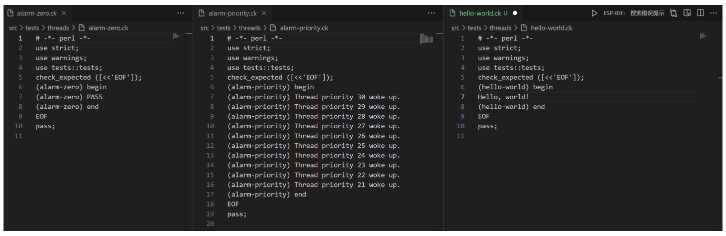


```
root@7d814ccd9a7d:~/pintos/src/threads# pintos -- -q run hello-world
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/qi1eNABVEH.dsk
graphic -monitor null
Pintos hda1
Loading...........
Kernel command line: -q run hello-world
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  105,676,800 loops/s.
Boot complete.
Executing 'hello-world':
(hello-world) begin
Hello, world!
(hello-world) end
Execution of 'hello-world' complete.
Timer: 29 ticks
Thread: 0 idle ticks, 30 kernel ticks, 0 user ticks
Console: 386 characters output
Keyboard: 0 keys pressed
Powering off...
root@7d814ccd9a7d:~/pintos/src/threads#
```

图 8: 运行结果

6 配置 make check

pintos 的测试检查使用 perl 脚本语言，观察其他的.ck 文件，可以发现格式几乎一致。不同的地方在于预期的结果不同。故根据其他两个文件对比，可以照搬格式，然后将输出结果写入特殊的位置即可。
注意到其他文件里最后都留了一行空行，我们也可以留以避免不必要的 BUG。



src > tests > threads > alarm-zero.ck

```
# -*- perl -*-
use strict;
use warnings;
use tests::tests;
check_expected ([<<'EOF']);
(alarm-zero) begin
(alarm-zero) PASS
(alarm-zero) end
EOF
pass;

```

src > tests > threads > alarm-priority.ck

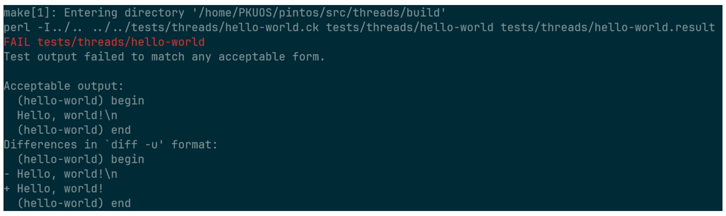
```
# -*- perl -*-
use strict;
use warnings;
use tests::tests;
check_expected ([<<'EOF']);
(alarm-priority) begin
(alarm-priority) Thread priority 30 woke up.
(alarm-priority) Thread priority 29 woke up.
(alarm-priority) Thread priority 28 woke up.
(alarm-priority) Thread priority 27 woke up.
(alarm-priority) Thread priority 26 woke up.
(alarm-priority) Thread priority 25 woke up.
(alarm-priority) Thread priority 24 woke up.
(alarm-priority) Thread priority 23 woke up.
(alarm-priority) Thread priority 22 woke up.
(alarm-priority) Thread priority 21 woke up.
(alarm-priority) end
EOF
pass;

```

src > tests > threads > hello-world.ck

```
# -*- perl -*-
use strict;
use warnings;
use tests::tests;
check_expected ([<<'EOF']);
(hello-world) begin
Hello, world!
(hello-world) end
EOF
pass;

```

图 9: 配置 make check

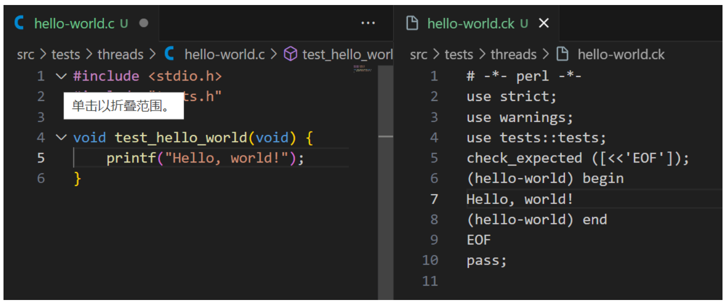
保存文件，并查看 make check 指令的结果，如下图所示：



```
make[1]: Entering directory '/home/PKUOS/pintos/src/threads/build'
perl -I../.. ../../tests/threads/hello-world.ck tests/threads/hello-world tests/threads/hello-world.result
FAIL tests/threads/hello-world
Test output failed to match any acceptable form.

Acceptable output:
  (hello-world) begin
  Hello, world!\n
  (hello-world) end
Differences in `diff -u' format:
  (hello-world) begin
- Hello, world!\n
+ Hello, world!
  (hello-world) end
```

图 10: Make Check 结果

发现结果错误，猜测应该是由于\n 导致的，换行符的存在可能会对缓冲区造成一定的影响。
故去掉换行符，修改如下：



src > tests > threads > hello-world.c > test_hello_worl

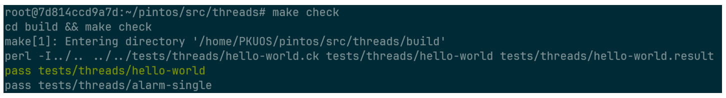
```
#include <stdio.h>
                ts.h"

void test_hello_world(void) {
    printf("Hello, world!");
}
```

src > tests > threads > hello-world.ck

```
# -*- perl -*-
use strict;
use warnings;
use tests::tests;
check_expected ([<<'EOF']);
(hello-world) begin
Hello, world!
(hello-world) end
EOF
pass;

```

图 11: 修改换行符

修改结束后再次执行 `make check` 指令，通过测试点。



```
root@7d814ccd9a7d:~/pintos/src/threads# make check
cd build && make check
make[1]: Entering directory '/home/PKUOS/pintos/src/threads/build'
perl -I../.. ../../tests/threads/hello-world.ck tests/threads/hello-world tests/threads/hello-world.result
pass tests/threads/hello-world
pass tests/threads/alarm-single
```

图 12: Make Check 结果

7 实验总结

通过本次实验，我成功地在 Pintos 系统中创建了一个新的测试用例 `hello-world`，并了解了 Pintos 操作系统架构的部分实现细节。实验过程中，我掌握了以下关键知识点：

1. **测试用例的创建与调试**：通过编写 `hello-world` 测试用例，我学习了 Pintos 测试框架的工作原理，理解了如何对操作系统的底层代码进行修改，并通过 `make check` 命令验证测试结果。
2. **调试与问题分析**：实验中遇到了输出不符合预期的问题，最终通过分析测试输出格式，发现是由于缺少换行符导致的缓冲区问题。
3. **实验工具的使用**：我还掌握了如何使用 VSCode 的 Remote Development 插件进行远程开发，以及如何通过全局搜索快速定位关键文件和代码模块，提升了代码调试和问题解决的效率。

五 附录

本次实验作出修改的代码如下所示：
同时上传到了 Github 之上，仓库地址为：https://github.com/Shichien/ECNU-23-SEI-Homework
请在上传的 PDF 文件中直接点击粉色链接即可。

```c
#include <stdio.h>
#include "tests.h"

void test_hello_world(void) {
    printf("Hello, world!\n");
}
```

hello-world.c

```perl
# -*- perl -*-
use strict;
use warnings;
use tests::tests;
check_expected ([<<'EOF']);
(hello-world) begin
Hello, world!
(hello-world) end
EOF
pass;
```

hello-world.ck

```makefile
# -*- makefile -*-

# Test names.
tests/threads_TESTS = $(addprefix tests/threads/,alarm-single   \
hello-world        \
alarm-multiple alarm-simultaneous alarm-priority alarm-zero    \
alarm-negative priority-change priority-donate-one       \
priority-donate-multiple priority-donate-multiple2       \
priority-donate-nest priority-donate-sema priority-donate-lower    \
priority-fifo priority-preempt priority-sema priority-condvar    \
priority-donate-chain                                             \
mlfqs-load-1 mlfqs-load-60 mlfqs-load-avg mlfqs-recent-1 mlfqs-fair-2 \
mlfqs-fair-20 mlfqs-nice-2 mlfqs-nice-10 mlfqs-block)

# Sources for tests.
tests/threads_SRC  = tests/threads/tests.c
tests/threads_SRC += tests/threads/hello-world.c
tests/threads_SRC += tests/threads/alarm-wait.c
tests/threads_SRC += tests/threads/alarm-simultaneous.c
tests/threads_SRC += tests/threads/alarm-priority.c
tests/threads_SRC += tests/threads/alarm-zero.c
tests/threads_SRC += tests/threads/alarm-negative.c
tests/threads_SRC += tests/threads/priority-change.c
tests/threads_SRC += tests/threads/priority-donate-one.c
tests/threads_SRC += tests/threads/priority-donate-multiple.c
tests/threads_SRC += tests/threads/priority-donate-multiple2.c
tests/threads_SRC += tests/threads/priority-donate-nest.c
tests/threads_SRC += tests/threads/priority-donate-sema.c
tests/threads_SRC += tests/threads/priority-donate-lower.c
tests/threads_SRC += tests/threads/priority-fifo.c
tests/threads_SRC += tests/threads/priority-preempt.c
tests/threads_SRC += tests/threads/priority-sema.c
tests/threads_SRC += tests/threads/priority-condvar.c
tests/threads_SRC += tests/threads/priority-donate-chain.c
tests/threads_SRC += tests/threads/mlfqs-load-1.c
tests/threads_SRC += tests/threads/mlfqs-load-60.c
tests/threads_SRC += tests/threads/mlfqs-load-avg.c
tests/threads_SRC += tests/threads/mlfqs-recent-1.c
tests/threads_SRC += tests/threads/mlfqs-fair.c
tests/threads_SRC += tests/threads/mlfqs-block.c

MLFQS_OUTPUTS =             \
tests/threads/mlfqs-load-1.output    \
tests/threads/mlfqs-load-60.output    \
tests/threads/mlfqs-load-avg.output   \
tests/threads/mlfqs-recent-1.output   \
tests/threads/mlfqs-fair-2.output    \
tests/threads/mlfqs-fair-20.output    \
tests/threads/mlfqs-nice-2.output    \
tests/threads/mlfqs-nice-10.output    \
tests/threads/mlfqs-block.output

$(MLFQS_OUTPUTS): KERNELFLAGS += -mlfqs
$(MLFQS_OUTPUTS): TIMEOUT = 480
```

Make.tests