

实验报告：Pintos 安装

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

1

二 内容与设计思想

1

三 使用环境

1

四 实验过程与分析

1

1 在 VSCode 中安装 Remote Development 插件

1

2 查看文件内容，作注释

2

3 添加指定内容

2

4 全局搜索，查看编译方式

2

5 尝试运行

3

6 配置 make check

4

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。

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 L^AT_EX 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。

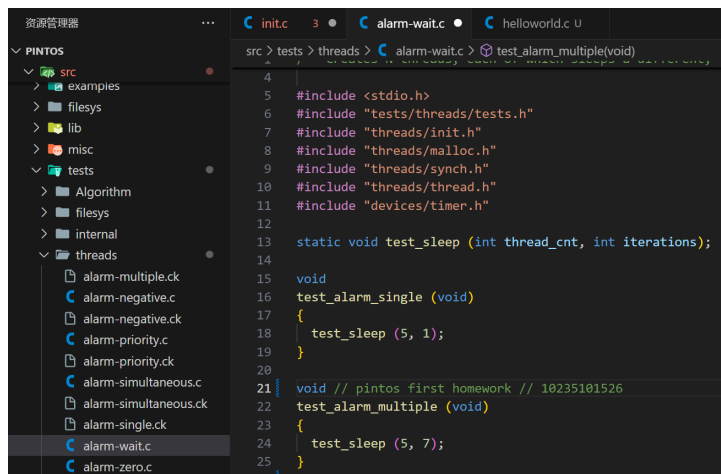


图 1: 在 VSCode 中打开 Pintos 项目

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。



图 2: 在 VSCode 中添加 hello-world.c 代码

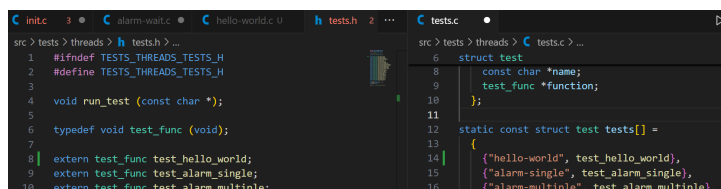


图 3: 添加指定内容

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。



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。

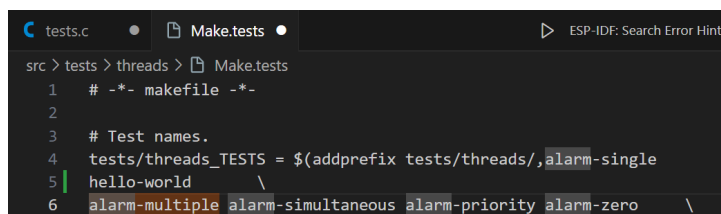


图 5: Make.testc 文件

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

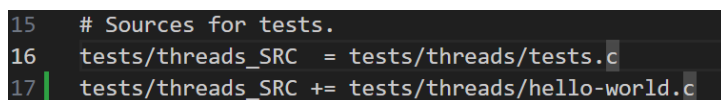


图 6: Make.testc 文件

5 尝试运行

接下来, 保存, 尝试运行。

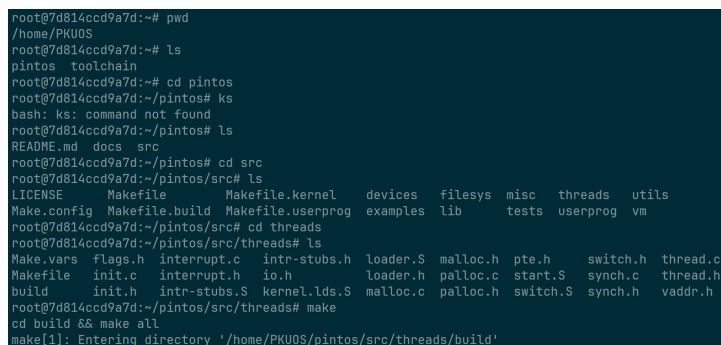


图 7: 尝试运行

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

```

make[1]: Entering directory '/home/PKU05/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

```

root@d814ccd9a7d:~/pintos/src/threads# pintos -- -q run hello-world
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/qileNABVEH.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... 185,676,800 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@d814ccd9a7d:~/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。

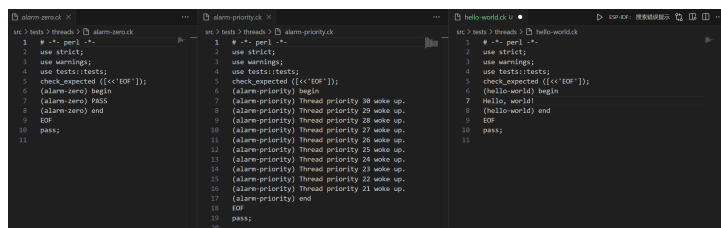


图 9: 配置 make check

保存文件，并查看 make check 指令的结果，如下图所示：

发现结果错误，猜测应该是由\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：

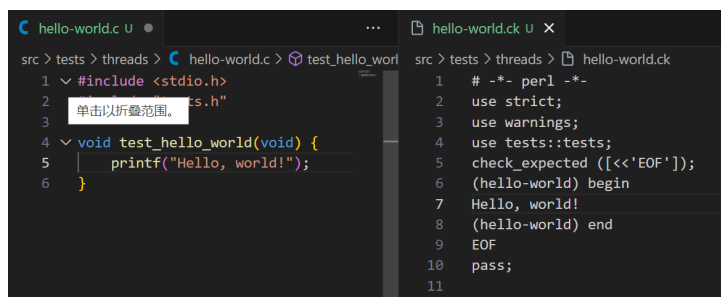


图 11: 修改换行符

修改结束后再次执行 make check 指令，通过测试点。

```

root@7d814ccd9e7d:~/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：

```

1 #include <stdio.h>
2 #include "tests.h"
3
4 void test_hello_world(void) {
5     printf("Hello, world!\n");
6 }

```

hello-world.c

```

1 # -*- perl -*-
2 use strict;
3 use warnings;
4 use tests::tests;
5 check_expected ([<<'EOF']]);
6 (hello-world) begin
7 Hello, world!
8 (hello-world) end
9 EOF
10 pass;

```

hello-world.ck

```

1 # -*- makefile -*-
2
3 # Test names.
4 tests/threads_TESTS = $(addprefix tests/threads/,alarm-single \
5 hello-world \
6 alarm-multiple alarm-simultaneous alarm-priority alarm-zero \
7 alarm-negative priority-change priority-donate-one \
8 priority-donate-multiple priority-donate-multiple2 \
9 priority-donate-nest priority-donate-sema priority-donate-lower \
10 priority-fifo priority-preempt priority-sema priority-condvar \
11 priority-donate-chain \
12 mlfqs-load-1 mlfqs-load-60 mlfqs-load-avg mlfqs-recent-1 mlfqs-fair-2 \
13 mlfqs-fair-20 mlfqs-nice-2 mlfqs-nice-10 mlfqs-block)
14

```

```

15 # Sources for tests.
16 tests/threads_SRC = tests/threads/tests.c
17 tests/threads_SRC += tests/threads/hello-world.c
18 tests/threads_SRC += tests/threads/alarm-wait.c
19 tests/threads_SRC += tests/threads/alarm-simultaneous.c
20 tests/threads_SRC += tests/threads/alarm-priority.c
21 tests/threads_SRC += tests/threads/alarm-zero.c
22 tests/threads_SRC += tests/threads/alarm-negative.c
23 tests/threads_SRC += tests/threads/priority-change.c
24 tests/threads_SRC += tests/threads/priority-donate-one.c
25 tests/threads_SRC += tests/threads/priority-donate-multiple.c
26 tests/threads_SRC += tests/threads/priority-donate-multiple2.c
27 tests/threads_SRC += tests/threads/priority-donate-nest.c
28 tests/threads_SRC += tests/threads/priority-donate-sema.c
29 tests/threads_SRC += tests/threads/priority-donate-lower.c
30 tests/threads_SRC += tests/threads/priority-fifo.c
31 tests/threads_SRC += tests/threads/priority-preempt.c
32 tests/threads_SRC += tests/threads/priority-sema.c
33 tests/threads_SRC += tests/threads/priority-condvar.c
34 tests/threads_SRC += tests/threads/priority-donate-chain.c
35 tests/threads_SRC += tests/threads/mlfqs-load-1.c
36 tests/threads_SRC += tests/threads/mlfqs-load-60.c
37 tests/threads_SRC += tests/threads/mlfqs-load-avg.c
38 tests/threads_SRC += tests/threads/mlfqs-recent-1.c
39 tests/threads_SRC += tests/threads/mlfqs-fair.c
40 tests/threads_SRC += tests/threads/mlfqs-block.c
41
42 MLFQS_OUTPUTS = \
43 tests/threads/mlfqs-load-1.output \
44 tests/threads/mlfqs-load-60.output \
45 tests/threads/mlfqs-load-avg.output \
46 tests/threads/mlfqs-recent-1.output \
47 tests/threads/mlfqs-fair-2.output \
48 tests/threads/mlfqs-fair-20.output \
49 tests/threads/mlfqs-nice-2.output \
50 tests/threads/mlfqs-nice-10.output \
51 tests/threads/mlfqs-block.output
52
53 $(MLFQS_OUTPUTS): KERNELFLAGS += -mlfqs
54 $(MLFQS_OUTPUTS): TIMEOUT = 480

```

Make.tests