山东大学　计算机科学与技术　学院

操作系统　课程实验报告

学号：201705130120	姓名：苑宗鹤	班级：17 1 班
实验题目：实验一		
实验学时：2	实验日期：2020/3/13	
实验目的： 　目的 （1）　安装编译 Nachos 系统, 理解 Nachos 系统的组织结构, 熟悉 c++编程语言 （2）　安装测试 gccMIPS 交叉编译器； （3）　掌握利用 Linux 调试工具 GDB 调试跟踪 Nachos 的执行过程； （4）　通过跟踪 Nachos 的 C++程序及汇编代码，理解 Nachos 中线程上下文切换的过程。 （5）　阅读 Nachos 的相关源代码，理解 Nachos 内核的工作原理及其测试过程。 任务 install the LINUX operating system (optional) install the Nachos system and gcc mips crOss compiler compile and test the Nachos system installed exercise with gdb context switch in Nachos		
实验环境：ubuntu18 x64 windows10 clion python3		

实验步骤：

1. 安装 linux

由于需要在 x64 上运行 32 位程序 所以需要装必要的库

sudo apt-get install cmake

sudo apt-get install lib32stdc++6

sudo apt-get install libc6-dev-i386

sudo apt-get install g++-multilib

sudo apt-get install build-essential module-assistant

sudo apt-get install g++

sudo apt-get install build-essential g++-multilib gcc-multilib －y
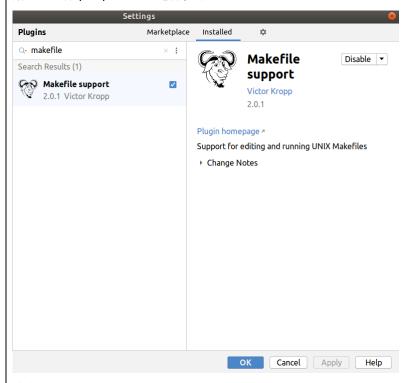
sudo apt-get install gcc

sudo dpkg –add-architecture i386

安装 clion 编译器

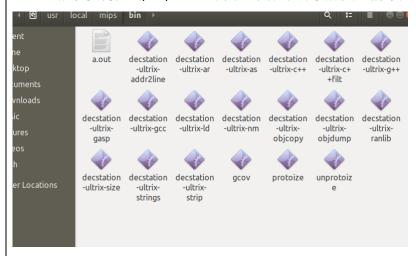

在 clion 官网上下载 clion

解压 运行./bin/clion.sh 进行安装

安装 clion 的 makefile 支持

Pip install compiledb

安装编译数据库 用来让 clion 分析 makefile 的关系

安装交叉编译器

将 gccMIPS 解压之后 执行 sudo cp -r mips /usr/local 拷贝到安装路径中
注意:之所以要拷贝到/usr/local 是为了让其他程序执行交叉编译程序的时候有权限执行

测试 gccMIPS

Cd test 进入测试目录

执行 make clean

Make

测试 Nachos

Nachos 的内核基本测试

进入 threads 测试基本内核

执行 make clean

Make

./nachos 运行

```
yuan@ubuntu:~/Desktop/code/threads$ ./nachos
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
yuan@ubuntu:~/Desktop/code/threads$
```

Gdb 调试 switch 观察进程切换

放不下了 请看下一页>>>>>>.

```
(gdb) q
yuan@ubuntu:~/Desktop/code/threads$ gdb nachos
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from nachos...done.
(gdb) break SWITCH
Breakpoint 1 at 0x4642
(gdb) start
Temporary breakpoint 2 at 0xdcd: file main.cc, line 87.
Starting program: /home/yuan/Desktop/code/threads/nachos

Temporary breakpoint 2, main (argc=1, argv=0xffffd134) at main.cc:87
87              DEBUG('t', "Entering main");
(gdb) c
Continuing.
*** thread 0 looped 0 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 1 looped 0 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 0 looped 1 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 1 looped 1 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 0 looped 2 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 1 looped 2 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 0 looped 3 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 1 looped 3 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 0 looped 4 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
*** thread 1 looped 4 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) c
Continuing.
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
[Inferior 1 (process 11515) exited normally]
(gdb)
```

断点 ThreadRoot 观察执行情况


```
yuan@ubuntu:~/Desktop/code/threads$ gdb nachos
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from nachos...done.
(gdb) break ThreadRoot
Breakpoint 1 at 0x4638
(gdb) start
Temporary breakpoint 2 at 0xdcd: file main.cc, line 87.
Starting program: /home/yuan/Desktop/code/threads/nachos

Temporary breakpoint 2, main (argc=1, argv=0xffffd134) at main.cc:87
87          DEBUG('t', "Entering main");
(gdb) Quit
(gdb) c
Continuing.
*** thread 0 looped 0 times

Breakpoint 1, 0x56559638 in ThreadRoot ()
(gdb) c
Continuing.
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
[Inferior 1 (process 11525) exited normally]
(gdb) c
The program is not being run.
(gdb) a
```

1. 请将鼠标指针移入其中或按 Ctrl+G

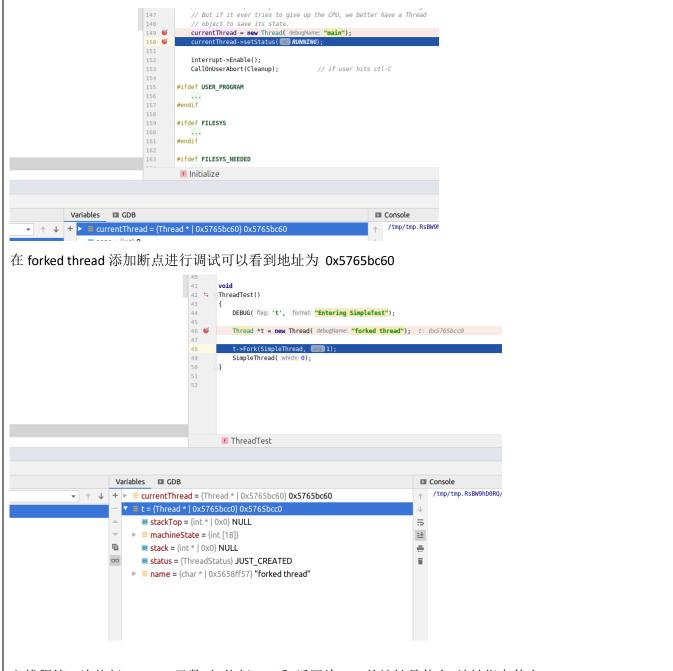观察四个函数的地址


```
Reading symbols from nachos...done.
(gdb) b InterruptEnable
Breakpoint 1 at 0x2912: file thread.cc, line 242.
(gdb) b SimpleThread
Breakpoint 2 at 0x2b43: file threadtest.cc, line 29.
(gdb) b ThreadFinish
Breakpoint 3 at 0x28ea: file thread.cc, line 241.
(gdb) b ThreadRoot
Breakpoint 4 at 0x4638
(gdb)
```

The main thread of Nachos

直接在创建主进程处加入断点 然后启动调试 添加 watch 可以看到 地址为 0x5765bc60

```
147        // But if it ever tries to give up the CPU, we better have a Thread
148        // object to save its state.
149        currentThread = new Thread( debugName: "main");
150        currentThread->setStatus( RUNNING);
151
152        interrupt->Enable();
153        CallOnUserAbort(Cleanup);           // if user hits ctl-C
154
155  #ifdef USER_PROGRAM
156        ...
157  #endif
158
159  #ifdef FILESYS
160        ...
161  #endif
162
163  #ifdef FILESYS_NEEDED
```

f Initialize

Variables	GDB		Console
▼ ↑ ↓ + ▶ ≡ currentThread = {Thread * | 0x5765bc60} 0x5765bc60			↑ /tmp/tmp.RsBW9h

在 forked thread 添加断点进行调试可以看到地址为 0x5765bc60

```
40
41        void
42   ↳   □ThreadTest()
43        {
44            DEBUG( flag: 't',  format: "Entering SimpleTest");
45
46            Thread *t = new Thread( debugName: "forked thread");   t: 0x5765bcc0
47
48            t->Fork(SimpleThread, arg: 1);
49            SimpleThread( which: 0);
50        □}
51
52
```

f ThreadTest

Variables	GDB	Console
▼ ↑ ↓ + ▶ ≡ currentThread = {Thread * | 0x5765bc60} 0x5765bc60		↑ /tmp/tmp.RsBW9hD0RQ/
− ▼ ≡ t = {Thread * | 0x5765bcc0} 0x5765bcc0		↓
▲ 🔢 stackTop = {int * | 0x0} NULL		⇥
▶ ≡ machineState = {int [18]}		↕
▼ 🔢 stack = {int * | 0x0} NULL		🖨
🔢 status = {ThreadStatus} JUST_CREATED		🗑
∞ ▶ ≡ name = {char * | 0x5658ff57} "forked thread"		

主线程第一次执行 SWITCH 函数时 执行 ret 后 返回给 cpu 的地址是什么 地址指向什么
由下图可知 返回的地址为 0x56559634 指向 threadroot

```
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from nachos...done.
(gdb) b SWITCH
Breakpoint 1 at 0x4642
(gdb) ni
The program is not being run.
(gdb) start
Temporary breakpoint 2 at 0xdcd: file main.cc, line 87.
Starting program: /home/yuan/Desktop/code/threads/nachos

Temporary breakpoint 2, main (argc=1, argv=0xffffd134) at main.cc:87
87          DEBUG('t', "Entering main");
(gdb) ni
0x56555dd0     87          DEBUG('t', "Entering main");
(gdb) ni
0x56555dd6     87          DEBUG('t', "Entering main");
(gdb) c
Continuing.
*** thread 0 looped 0 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) ni
0x56559647 in SWITCH ()
(gdb) ni
0x5655964b in SWITCH ()
(gdb) ni
0x5655964e in SWITCH ()
(gdb) ni
0x56559651 in SWITCH ()
(gdb) ni
0x56559654 in SWITCH ()
(gdb) ni
0x56559657 in SWITCH ()
(gdb) ni
0x5655965a in SWITCH ()
(gdb) ni
0x5655965d in SWITCH ()
(gdb) ni
0x5655965f in SWITCH ()
(gdb) ni
0x56559665 in SWITCH ()
(gdb) ni
0x56559668 in SWITCH ()
(gdb) ni
0x5655966b in SWITCH ()
(gdb) ni
0x5655966e in SWITCH ()
(gdb) ni
0x56559672 in SWITCH ()
(gdb) ni
0x56559675 in SWITCH ()
(gdb) ni
0x5655967b in SWITCH ()
(gdb) ni
0x5655967e in SWITCH ()
(gdb) ni
0x56559681 in SWITCH ()
(gdb) ni
0x56559684 in SWITCH ()
(gdb) ni
0x56559687 in SWITCH ()
(gdb) ni
0x5655968a in SWITCH ()
(gdb) ni
0x5655968d in SWITCH ()
(gdb) ni
0x5655968f in SWITCH ()
(gdb) ni
0x56559692 in SWITCH ()
(gdb) ni
0x56559695 in SWITCH ()
(gdb) ni
0x5655969a in SWITCH ()
(gdb) ni
0x56559634 in ThreadRoot ()
(gdb) ni
0x56559635 in ThreadRoot ()
(gdb) ni
0x56559637 in ThreadRoot ()
(gdb)
```

调用 fork 创建线程之后 SWITCH 时 返回给 cpu 的地址是什么

在之前的基础上 c 然后 ni 查找地址 为 0x6556396 指向 debug 语句

```
*** thread 1 looped 0 times

Breakpoint 1, 0x56559642 in SWITCH ()
(gdb) ni
0x56559647 in SWITCH ()
(gdb) ni
0x5655964b in SWITCH ()
(gdb) ni
0x5655964e in SWITCH ()
(gdb) ni
0x56559651 in SWITCH ()
(gdb) ni
0x56559654 in SWITCH ()
(gdb) ni
0x56559657 in SWITCH ()
(gdb) ni
0x5655965a in SWITCH ()
(gdb) ni
0x5655965d in SWITCH ()
(gdb) ni
0x5655965f in SWITCH ()
(gdb) ni
0x56559665 in SWITCH ()
(gdb)
0x56559668 in SWITCH ()
(gdb) ni
0x5655966b in SWITCH ()
(gdb) ni
0x5655966e in SWITCH ()
(gdb) ni
0x56559672 in SWITCH ()
(gdb) ni
0x56559675 in SWITCH ()
(gdb) ni
0x5655967b in SWITCH ()
(gdb) ni
0x5655967e in SWITCH ()
(gdb) ni
0x56559681 in SWITCH ()
(gdb) ni
0x56559684 in SWITCH ()
(gdb) ni
0x56559687 in SWITCH ()
(gdb) ni
0x5655968a in SWITCH ()
(gdb) ni
0x5655968d in SWITCH ()
(gdb) ni
0x5655968f in SWITCH ()
(gdb) ni
0x56559692 in SWITCH ()
(gdb) ni
0x56559695 in SWITCH ()
(gdb) ni
0x5655969a in SWITCH ()
(gdb) ni
0x56556396 in Scheduler::Run (this=0x56562c40, nextThread=0x56562cc0) at scheduler.cc:116
116         SWITCH(oldThread, nextThread);
(gdb) ni
118         DEBUG('t', "Now in thread \"%s\"\n", currentThread->getName());
(gdb) ni
```

实验结果：

问题及收获：

学号姓名实验一.doc；