北京航空航天大學
BEIHANG UNIVERSITY

操作系统 Lab5-1 课上测试简介

测试说明

- 考试时间 14:15 ~ 16：00
- 每次课上测试题目分为基础测试和附加测试(选做)两部分
- 通过课上测试的条件是基础测试通过（基础题成绩>=60 ）
- 通过附加测试将会给予额外加分（附加题成绩>=60 ）
- 在本次课上测试中，本地测试的难度很低且非常有助于发现bug，**强烈建议本地测试通过后提交评测！**

- *注意，若未通过基础测试，则无论是否通过附加测试均记为未通过。*

Lab5-1课上基础题

Step1: 创建lab5-1-exam分支

- cd ~/学号-lab/
- git checkout lab5
- git checkout –b lab5-1-exam

Step2: 完成Lab5-1-exam

- 实验背景：系统调用的一个重要作用就在于，它为用户进程提供了一种访问外部设备的途径。在完成实验的过程中你已经使用了syscall_read_dev等系统调用来访问磁盘设备，在这里希望你使用系统调用为用户进程提供更加丰富的功能。

第1部分：syscall_get_time

- 添加新的系统调用 int syscall_get_time();（在user/lib.h中添加定义并在已有文件中进行实现）。该系统调用的返回值是当前的UNIX标准时间（1970年1月1日至今的时间），单位为秒。
- 注意：获取UNIX标准时间需要直接对gxemul进行读取，用其他方式获得时间无法通过评测，对gxemul设备的具体信息需要参考gxemul文档节选。

  提示：当前UNIX标准时间是一个10位整数，且前三位数为162

  当系统调用输出一直为0时，请仔细阅读gxemul的设备规范！！！

Step2: 完成Lab5-1-exam

第2部分：syscall_read_str

- 添加新的系统调用 int syscall_read_str(char *buf, int secno); （在user/lib.h中添加定义并在已有文件中进行实现）
- 功能：系统调用从控制台中读取一个以换行('\r')结尾的字符串，并将读取的字符串写入0号磁盘的secno号扇区中。

✓ 系统调用的返回值为读取的字符串的长度，该系统调用需要将读取的字符串存入buf中（buf和写入磁盘的字符串都不包括'\r'且需要以'\0'结尾）。

✓ 当系统调用开始之后进入内核态并且不断读取控制台输入，直到读取到结束标志符号'\r'才从内核态返回。

✓ （提示：需要循环读取控制台地址，读取到的非零值即为控制台的输入，可参考gxemul文档节选）

✓ 保证secno在合适的范围内且buf的长度小于512字节，也就是说只需要写入一个扇区。

本地测试说明：可以使用已有的文件系统进程进行测试

✓ 检查在init/init.c下是否已有进程创建语句ENV_CREATE(fs_serv);

✓ 之后只需要对fs/test.c文件的main()进行修改即可测试书写的系统调用

Step3: 提交更改

- cd ~/学号-lab/
- git add --all
- git commit -a -m “balabala”
- git push origin lab5-1-exam:lab5-1-exam

Step4: 提交结果

```
[ PASSED:2 ]
[ TOTAL:2 ]
[ You have passed syscall_get_time testcase ]
[ PASSED:4 ]
[ TOTAL:4 ]
[ You have passed all syscall_read_str testcases! ]
[ You have passed lab5-1-exam! ]
[ You got 100 (of 100) this time. Sat May 15 11:39:39 CST 2021 ]
```

Lab5-1-exam得分说明：

- 课下强测 10分
- 编译正确 10分（请先对所有系统调用进行定义，否则得分为0分）
- syscall_get_time 30分
- syscall_read_str 返回值和buf正确 20分
- syscall_read_str 写入磁盘 30分

Lab5-1课上附加题

Step5: 创建lab5-1-Extra分支

- cd ~/学号-lab/
- git checkout lab5
- 或：git checkout lab5-1-exam
- git checkout -b lab5-1-Extra
- （可以基于lab5或lab5-1-exam创建分支）

Step6: 完成lab5-1-Extra

- 实验背景：从boot/start.S中可以看出，我们的操作系统在启动之前关闭了所有的中断。而在开始启用进程前，操作系统通过调用kclock_init()函数开启了时钟中断。现在我们希望能够开启操作系统的控制台中断，实现对控制台输入的中断响应。同时我们还希望记录下每一次触发控制台中断的时间。
- 第1部分：打开控制台中断
- 首先你需要为系统打开控制台中断。在开启控制台中断时需要建议参考对于时间中断的实现。
- 建议参考的文件有：lib/genex.S　lib/kclock.c　lib/kclock_asm.S　include/asm/cp0regdef.h。
- 你需要创建include/kcons.h文件，在其中声明函数void kcons_init()并对其进行实现，该函数可以开启控制台中断。
- 模仿时钟中断的实现，你需要创建文件lib/kcons.c、lib/kcons_asm.S和lib/handle_cons_ir.c并书写相应函数，并修改genex.S文件，使得触发控制台中断后可以跳转到控制台中断处理函数。除上述文件外请不要创建其他文件。

Step6: 完成lab5-1-Extra

- 第2部分：处理控制台中断
- 你需要在handle_cons_ir.c中书写控制台中断的处理函数，处理函数有如下要求：
- 在第一次触发控制台中断时处理函数需要打印printf("CP0 STATUS: %x, 1st interrupt: %d\n", status, time);

  status是进入中断处理程序时CP0 STATUS寄存器的值，time是第一次触发控制台中断时的UNIX标准时间，单位为秒。
- 在之后触发控制台中断时需要打印语句printf("interval: %d\n", interval);

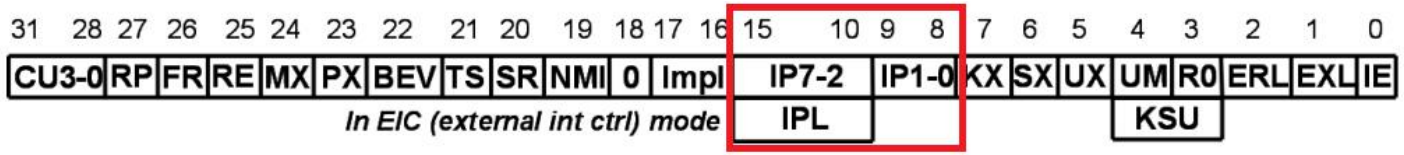
  interval表示当前时间和第一次触发控制台中断的时间差，单位为秒。
- 当interval的值大于等于5时需要结束gxemul的模拟。

  在结束模拟前需要输出语句：printf("length=%d, string=%s\n", length, string);

  string是将所有控制台输入的字符进行拼接的结果，length表示string的长度。我们保证在触发控制台中断时仅输入字母和数字，且string的长度不超过128（提示：在处理函数中使用静态字符串数组实现）。

Step6: 完成lab5-1-Extra

- 提示：下图是CP0状态寄存器的设置，其中IP7-0这8个中断位决定了哪些中断源会得到响应。这8个中断源和硬件的对应关系可以在gxemul文档节选中找到。从右表中可以看出，4号rtc对应的是时钟中断，2号cons对应的是本次需要开启的控制台中断（console）。



testmips (as native MIPS interrupts)	
IRQ:	Used for:
7	MIPS count/compare interrupt
6	mp (inter-processor interrupts)
4	rtc
3	ether
2	cons

- 本地测试说明：首先需要将新创建的文件添加到lib/Makefile中。测试程序需要在init/init.c的main函数中使用kcons_init()开启控制台中断（为方便测试，建议不要开启时钟中断）。开启gxemul模拟之后，每次敲击键盘都会触发控制台中断，进入中断处理函数并进行相应的输出操作。在距离第一次控制台中断五秒的时间之后再敲击键盘时，gxemul会自动退出，并在退出前输出指定的语句。

（注意：退出前输入的最后一个字符也应该打印，如右图）

```
CP0 STATUS:          , 1st interrupt:
interval: 0
interval: 0
interval: 1
interval: 1
interval: 1
interval: 2
interval: 2
interval: 2
interval: 2
interval: 2
interval: 3
interval: 6
length = 12, string = goodbyeworld
```

Step7: 提交更改

- 必须通过基础测试才能获得附加题分数
- cd ~/学号-lab/
- git add --all
- git commit -a -m “balabala...”
- git push origin lab5-1-Extra:lab5-1-Extra

Step8: 提交结果

```
remote: Beginning test  >>>>>>>>>
remote: [ possible error codes: ]
remote: [ DONT_HAVE_FIRST = -1 ]
remote: [ FIRST_TIME_WRONG = -2 ]
remote: [ INTERVAL_LSS_ZERO = -3 ]
remote: [ INTERVAL_WRONG = -4 ]
remote: [ NOT_QUIT_RIGHT = -5 ]
remote: [ VALUE_WRONG = -6 ]
remote: [ STATUS_WRONG = -7 ]
remote: [ Extra Test Succeed ]
remote: [ You got 100 (of 100) this time. Thu May 20 11:17:20 CST 2021 ]
```

Lab5-1-Extra得分说明：

- 编译正确 10分
- 开启控制台中断并正确书写处理函数 90分（评测机将对CP0寄存器的值、首次中断的时间、时间间隔、字符串输出和gxemul终止进行测试，全部正确方可通过，报错信息仅供参考，建议在本地充分测试后进行提交）

北京航空航天大學
BEIHANG UNIVERSITY

下面请同学们开始做题
有问题可以随时提问

祝实验顺利！