OS实验2第一部分实验报告

实验过程

在该部分中需要添加两个系统调用`sys_print_val`和`sys_str2num`来实现把字符串读入并转化成整型输出的测试函数。操作流程如下：

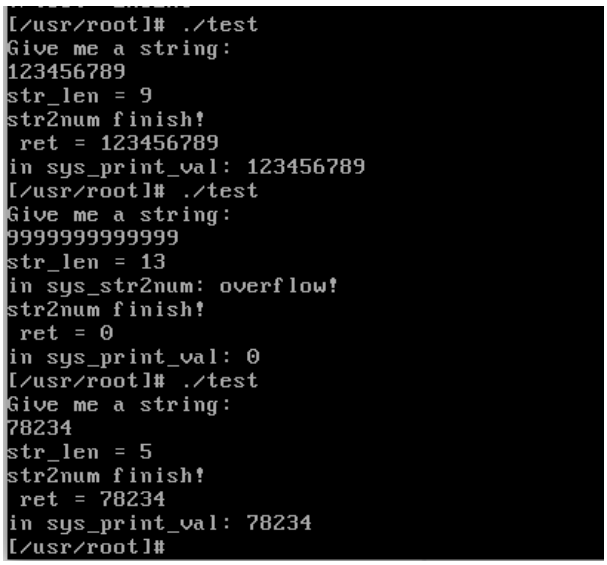
1. 在 `unistd.h` 中添加 `# define __NR_XXX`
2. 在 `kernel/system_call.s` 修改 `nr_system_calls = 74`
3. 在 `include/linux/sys.h` 添加函数原型以及在 `sys_call_table[ ]` 中添加对应指针
4. 写出对应的系统调用函数 `sys_xxx`
5. 修改 `makefile文件`
6. 编写 `test.c` 测试文件

以上 步骤均在 `ubuntu 18.04` 下完成，***此后*** 使用 **qemu** 编译运行 `linux 0.11`：

7. 使用命令编译并执行测试文件

```
gcc test.c -o test
./test
```

实验结果



> 可以看到上面的结果中正常的转化成功实现，并且，位数溢出时会给出 `overflow` 的提示。

问题回答

- **如何在 Linux0.11 中添加一个系统调用？** 答：在实验过程的 1~4 步骤已经给出流程。

- **系统是如何通过系统调用号索引到具体函数的?** 答：首先由例化 `syscallx()` 唤起一个 `int 0x80` 的中断，此中断通过 `__NR_##name` 得到了宏定义中的系统调用号，此后会执行汇编函数 `system_call` 其中通过 `call sys_call_table(, %eax, 4)` 调用系统函数表中该调用号所指的函数，由其实现具体功能。在这一流程中也完成了用户态到内核态的切换。

- **在Linux 0.11中, 系统调用多支持几个参数? 有什么方法可以超过这个限制吗?** 答：最多支持 **3** 个参数，可以自己仿写__syscallx,来增加参数。