浙江大学实验报告

课程名称： 操作系统 实验类型： 综合型
实验项目名称： 同步互斥
学生姓名： 杨樾人 学号： 3160104875
电子邮件地址： yangyueren@outlook.com
实验日期： 2018 年 11 月 28 日

一、实验环境

Ubuntu 16.04.3LTS(GNU/Linux 4.4.0-93-generic x86_64)，mac 下虚拟机

二、实验内容和结果及分析

1. 实验内容：

编写一个Linux的内核模块，其功能是遍历操作系统所有进程。该内核模块输出系统中每个进程的：名字、进程pid、进程的状态、父进程的名字等；以及统计系统中进程个数，包括统计系统中TASK_RUNNING、TASK_INTERRUPTIBLE、TASK_UNINTERRUPTIBLE、TASK_ZOMBIE、TASK_STOPPED等（还有其他状态）状态进程的个数。同时还需要编写一个用户态下执行的程序，格式化输出（显示）内核模块输出的内容。

2. 设计文档：

1. 模块划分：

按功能模块划分：

内核模块：加载模块，并输出各进程信息到log中。

用户态程序：读取log中的信息，打印在屏幕上。

2. 程序流程概述：

1. 内核模块加载到内核中并执行init函数；
2. 遍历进程信息，输出到日志
3. 卸载模块
4. 执行用户态程序，输出日子内容

2. 概要设计说明：

2.1 编写目的：

内核模块加载到内核中读取所有进程信息并输出到日志。

用户态程序将日志打印到屏幕。

2.1 参考资料：

1. Linux线程创建和同步的编程参考资料：
2. 操作系统原理教材“Operating System Concepts”的第四章和第六章
3. 边干边学Linux内核指导

2.2 输入输出：

输入：无

输出：各进程信息和进程的统计信息

2.3 模块具体实现：

内核模块：

task_struct *pp，通过for_each_process函数遍历进程。

各进程统计信息变量：

```
int totalProcess = 0;//record the different state of processes


   int running = 0;
   int interruptible = 0;
   int uninterruptible = 0;
   int zombie = 0;
int stopped = 0;
```

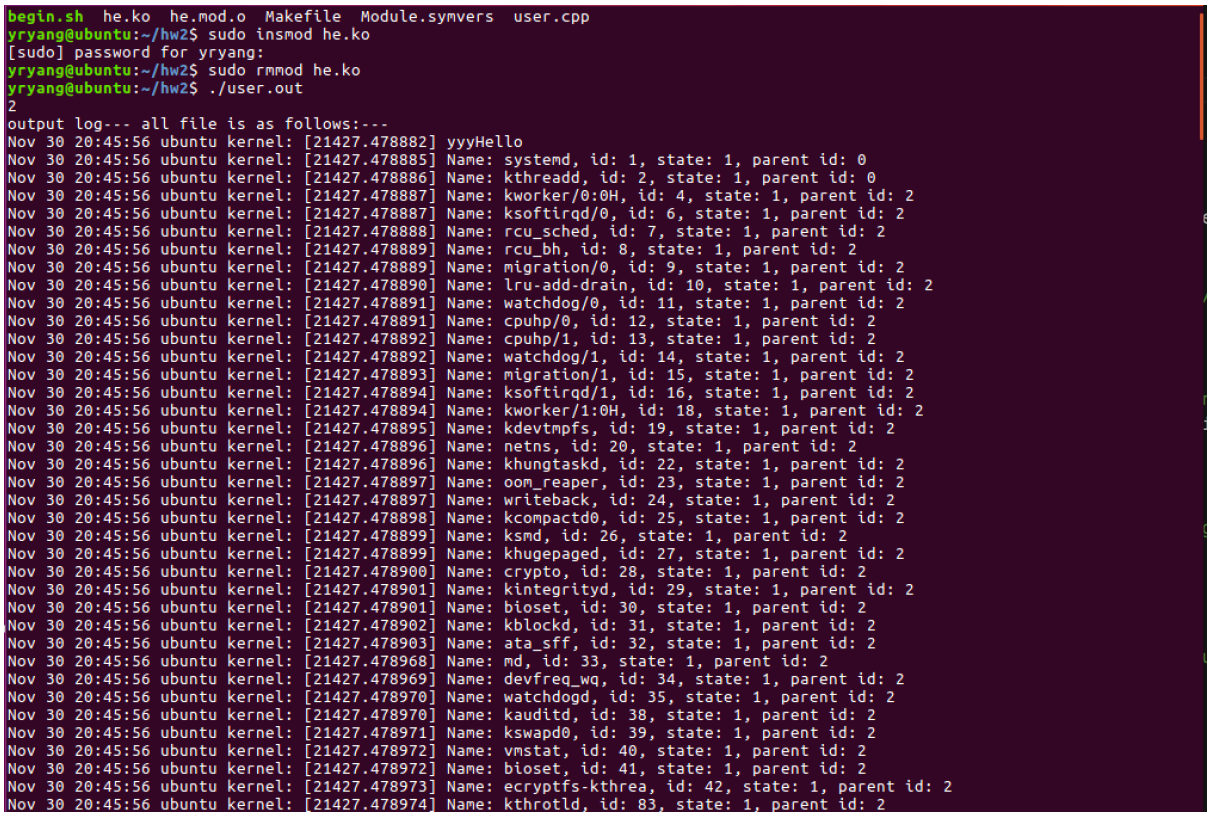
init_module函数执行，遍历每个进程信息，并将进程信息name pid state parent输出到日志，同时统计该进程的状态，用统计信息变量记录。

cleanup_module函数，卸载模块，输出good bye。

用户态程序：

在日志中查找最后一次模块加载信息，并将所有信息输出到屏幕。

3. 程序运行结果截图：


begin.sh he.ko he.mod.o Makefile Module.symvers user.cpp
yryang@ubuntu:~/hw2$ sudo insmod he.ko
[sudo] password for yryang:
yryang@ubuntu:~/hw2$ sudo rmmod he.ko
yryang@ubuntu:~/hw2$ ./user.out
2
output log--- all file is as follows:---
Nov 30 20:45:56 ubuntu kernel: [21427.478882] yyyHello
Nov 30 20:45:56 ubuntu kernel: [21427.478885] Name: systemd, id: 1, state: 1, parent id: 0
Nov 30 20:45:56 ubuntu kernel: [21427.478886] Name: kthreadd, id: 2, state: 1, parent id: 0
Nov 30 20:45:56 ubuntu kernel: [21427.478887] Name: kworker/0:0H, id: 4, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478887] Name: ksoftirqd/0, id: 6, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478888] Name: rcu_sched, id: 7, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478889] Name: rcu_bh, id: 8, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478889] Name: migration/0, id: 9, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478890] Name: lru-add-drain, id: 10, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478891] Name: watchdog/0, id: 11, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478891] Name: cpuhp/0, id: 12, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478892] Name: cpuhp/1, id: 13, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478892] Name: watchdog/1, id: 14, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478893] Name: migration/1, id: 15, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478894] Name: ksoftirqd/1, id: 16, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478894] Name: kworker/1:0H, id: 18, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478895] Name: kdevtmpfs, id: 19, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478896] Name: netns, id: 20, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478896] Name: khungtaskd, id: 22, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478897] Name: oom_reaper, id: 23, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478897] Name: writeback, id: 24, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478898] Name: kcompactd0, id: 25, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478899] Name: ksmd, id: 26, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478899] Name: khugepaged, id: 27, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478900] Name: crypto, id: 28, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478901] Name: kintegrityd, id: 29, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478901] Name: bioset, id: 30, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478902] Name: kblockd, id: 31, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478903] Name: ata_sff, id: 32, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478968] Name: md, id: 33, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478969] Name: devfreq_wq, id: 34, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478970] Name: watchdogd, id: 35, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478970] Name: kauditd, id: 38, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478971] Name: kswapd0, id: 39, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478972] Name: vmstat, id: 40, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478972] Name: bioset, id: 41, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478973] Name: ecryptfs-kthrea, id: 42, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.478974] Name: kthrotld, id: 83, state: 1, parent id: 2

```
Nov 30 20:45:56 ubuntu kernel: [21427.479189] Name: jfsIO, id: 55663, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479190] Name: jfsCommit, id: 55664, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479192] Name: jfsCommit, id: 55665, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479192] Name: jfsSync, id: 55666, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479193] Name: bioset, id: 55704, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479194] Name: uuidd, id: 56634, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479194] Name: snapd, id: 60271, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479195] Name: polkitd, id: 60408, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479196] Name: avahi-daemon, id: 60869, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479196] Name: avahi-daemon, id: 60870, state: 1, parent id: 60869
Nov 30 20:45:56 ubuntu kernel: [21427.479197] Name: gpg-agent, id: 79728, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479198] Name: unity-scope-hom, id: 80495, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479198] Name: unity-files-dae, id: 80509, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479199] Name: unity-scope-loa, id: 80510, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479200] Name: kworker/1:1, id: 82436, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479200] Name: kworker/0:2, id: 87305, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479201] Name: notify-osd, id: 88043, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479202] Name: kworker/1:2, id: 92344, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479203] Name: kworker/0:1, id: 92345, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479203] Name: systemd-network, id: 92460, state: 1, parent id: 1
Nov 30 20:45:56 ubuntu kernel: [21427.479204] Name: update-manager, id: 93388, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479205] Name: kworker/u256:0, id: 96738, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479205] Name: kworker/u256:1, id: 96802, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479206] Name: kworker/u256:2, id: 96876, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479207] Name: kworker/1:0, id: 97371, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479207] Name: kworker/1:3, id: 97379, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479208] Name: kworker/u257:0, id: 97431, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479209] Name: hci0, id: 97432, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479209] Name: hci0, id: 97433, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479210] Name: kworker/u257:1, id: 97434, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479211] Name: kworker/u257:2, id: 97435, state: 1, parent id: 2
Nov 30 20:45:56 ubuntu kernel: [21427.479211] Name: gnome-terminal-, id: 97453, state: 1, parent id: 4313
Nov 30 20:45:56 ubuntu kernel: [21427.479212] Name: bash, id: 97460, state: 1, parent id: 97453
Nov 30 20:45:56 ubuntu kernel: [21427.479212] Name: sudo, id: 97475, state: 1, parent id: 97460
Nov 30 20:45:56 ubuntu kernel: [21427.479213] Name: insmod, id: 97476, state: 0, parent id: 97475
Nov 30 20:45:56 ubuntu kernel: [21427.479213] Name: systemd-udevd, id: 97477, state: 0, parent id: 54249
Nov 30 20:45:56 ubuntu kernel: [21427.479237] The number of total process is 266
Nov 30 20:45:56 ubuntu kernel: [21427.479238] The number of running process is 2
Nov 30 20:45:56 ubuntu kernel: [21427.479238] The number of interruptible process is 264
Nov 30 20:45:56 ubuntu kernel: [21427.479238] The number of uninterruptible process is 0
Nov 30 20:45:56 ubuntu kernel: [21427.479239] The number of zombie process is 0
Nov 30 20:46:03 ubuntu kernel: [21427.479239] The number of stopped process is 0
Nov 30 20:46:03 ubuntu kernel: [21434.172307] Good-bye!
```

结果分析：

该实验成功编译了内核模块，并实现内核模块的加载与卸载。

该内核模块也实现了遍历进程的功能，并将各个进程信息输出到了日志之中，并统计了各个状态进程的总数。

4. 源程序：（见附录）

三、讨论、心得

列举实验中遇到的问题和对应的解决方案：

1. 内核模块的 Makefile 文件编写。

内核模块的 Makefile 文件包含目标文件，目标文件依赖的文件，更新（或生成）目标文件的命令，并且每个命令之前必须是 TAB 键，否则会编译错误。我一开始 TAB 默认设置的是空格，所以会一直报编译错误，后来将 TAB 重新设置为 TAB 键，编译成功。

2. 用户态程序从 log 中读取最后一次模块输出的信息

由于每次加载模块都会在/var/log/kern.log 中输出信息，而我只想输出最后一次的信息，所有我在内核模块加载时会先输出 yyyHello 这个标志字符串，用来做标记。在用户态程序读取

了 log 文件后，会先查找 yyyHello 出现的最后一次的位置，然后程序从该位置开始写出 log 文件的内容。

我也查阅了是否可以通过 printk 直接输出到终端。

cat /proc/sys/kernel/printk

4 4 1 7

第一个“4”表示内核打印函数 printk 的打印级别，只有级别比他高的信息才能在控制台上打印出来，既 0—3 级别的信息。所以无法直接输出到终端。我采用了用户态程序读取 log 文件到方法，将信息打印到终端。

体会：

1. 对内核模块的加载和卸载过程了解更深
2. 对 Makefile 文件的编写更加深入了解
3. 对 printk 的打印级别做了了解

四、附录

hw2.c

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/sched.h>
#include <linux/init.h>
#include <linux/fs.h>
#include <asm/segment.h>
#include <asm/uaccess.h>
#include <linux/buffer_head.h>

static int __init hello_init(void)//加载内核模块
{

    int totalProcess = 0;//record the different state of processes

    int running = 0;
    int interruptible = 0;
    int uninterruptible = 0;
    int zombie = 0;
```

```
    int stopped = 0;

    int state;//进程的状态信息
    int exitState;

    printk("yyyHello\n");

    struct task_struct *pp;
    for_each_process(pp)//遍历
    {
        totalProcess++;
        printk(KERN_INFO "Name: %s, id: %d, state: %d, parent id: %d", pp->comm,
pp->pid, pp->state, pp->parent->pid);
        state = pp->state;//当前状态
        exitState = pp->exit_state;//退出状态
        if (exitState == 0)//如果未退出
        {
            switch (state)//判断当前状态
            {
            case TASK_RUNNING:
                running++;
                break;
            case TASK_INTERRUPTIBLE:
                interruptible++;
                break;
            case TASK_UNINTERRUPTIBLE:
                uninterruptible++;
                break;
            case TASK_STOPPED:
                stopped++;
                break;
            default:
                break;
            }
        }
        else
        {
            switch (exitState)//如果退出
            {
            case EXIT_ZOMBIE:
                zombie++;
                break;
            default:
                break;
            }
        }
    }
    printk(KERN_INFO "The number of total process is %d", totalProcess);//打印信息
    printk(KERN_INFO "The number of running process is %d", running);
```

```
    printk(KERN_INFO "The number of interruptible process is %d", interruptible);
    printk(KERN_INFO "The number of uninterruptible process is %d",
uninterruptible);
    printk(KERN_INFO "The number of zombie process is %d", zombie);
    printk(KERN_INFO "The number of stopped process is %d", stopped);
    return 0;
}
static void __exit hello_exit(void)//卸载内核模块
{
    printk("Good-bye!\n");
}
module_init(hello_init);
module_exit(hello_exit);
MODULE_LICENSE("GPL");
```

user.cpp

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

void testByLine(string file, string lable, int num)
{
    char buffer[1000];
    string temp;
    fstream outFile;
    outFile.open(file, ios::in);//打开文件
    int flag = 0;
    string str1 = lable;
    cout << "output log"
         << "--- all file is as follows:---" << endl;

    while (!outFile.eof())
    {
        outFile.getline(buffer, 1000, '\n'); //getline(char *,int,char) 表示该行字符
达到 256 个或遇到换行就结束
        temp = buffer;
        size_t found = temp.find(str1);
        if (found != string::npos)
            flag++;//直到最后一次的 label 标志，才开始输出信息
        if (flag == num)
            cout << buffer << endl;
```

```
    }
    outFile.close();
}
//返回 label 字符串在 log 中出现的次数
int findPosition(string file, string lable)
{
    char buffer[1000];
    string temp;
    fstream outFile;
    outFile.open(file, ios::in);
    int flag = 0;
    string str1 = lable;//查找 label 所出现的次数

    while (!outFile.eof())
    {
        outFile.getline(buffer, 1000, '\n'); //getline(char *,int,char) 表示该行字符
达到 256 个或遇到换行就结束
        temp = buffer;
        // cout << buffer <<endl;
        size_t found = temp.find(str1);
        if (found != string::npos)
            flag++;//label 出现的次数
    }
    outFile.close();
    return flag;
}
int main()
{
    string file = "/var/log/kern.log";
    string label = "yyyHello";//输出到 log 的标志
    // string label = "CarCross";
    int num = findPosition(file, label);//找到最后一个标志
    cout << num << endl;
    testByLine(file, label, num);//输出 log 里最后一次加载模块的信息
    return 0;
}
```