

# 《操作系统》课程作业

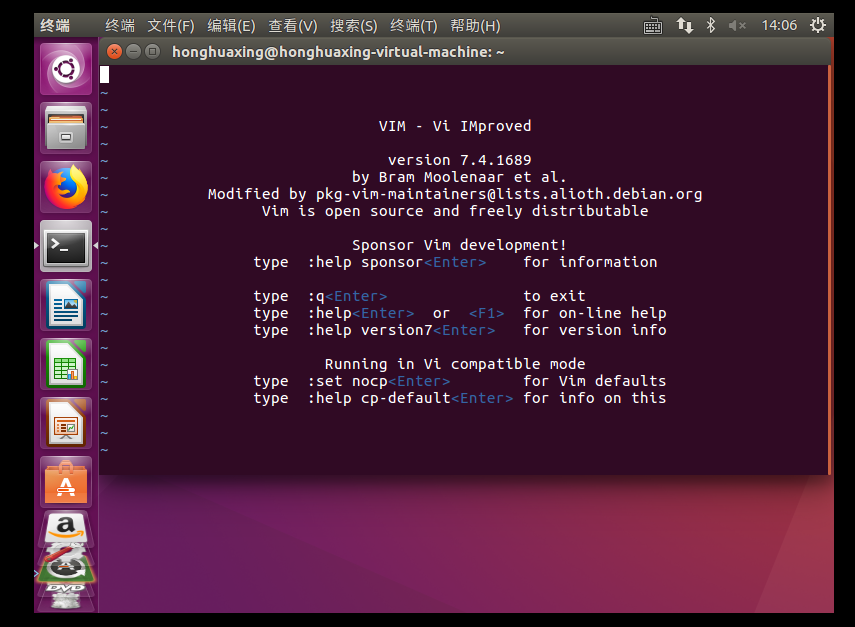
|  |  |  |
| --- | --- | --- |
| 学 号： | 16281004 | |
| 姓 名： | 洪华兴 | |
| 专 业： | 计算机科学与技术 | |
| 学 院： | 计算机与信息技术学院 | |
|  |  |  |
|  |  |  |
| 提交日期： | 2019年03月23日 | |

# 实验二: 进程控制

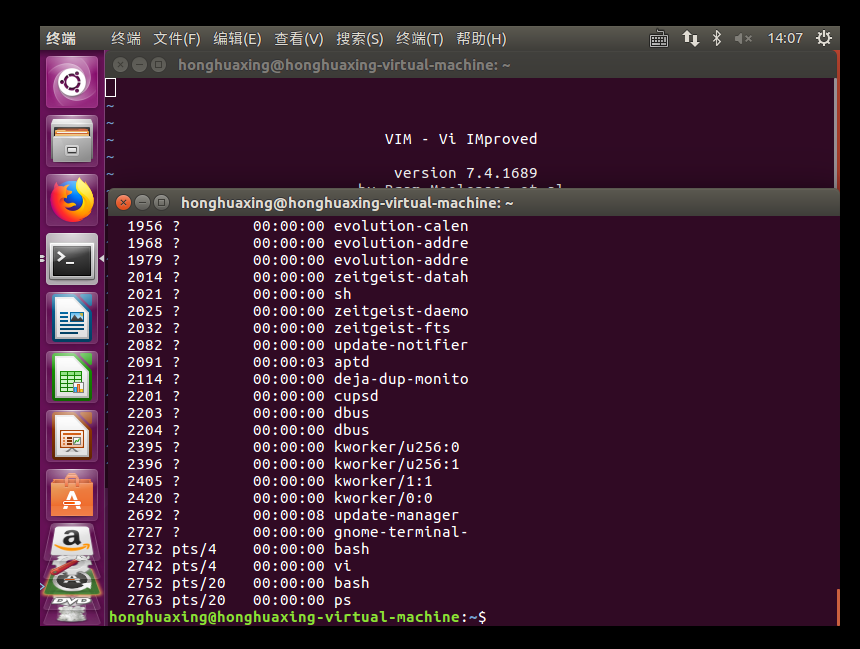
## 作业题目：

 一、打开一个vi进程。通过ps命令以及选择合适的参数，只显示名字为vi的进程。寻找vi进程的父进程，直到init进程为止。记录过程中所有进程的ID和父进程ID。将得到的进程树和由pstree命令的得到的进程树进行比较。

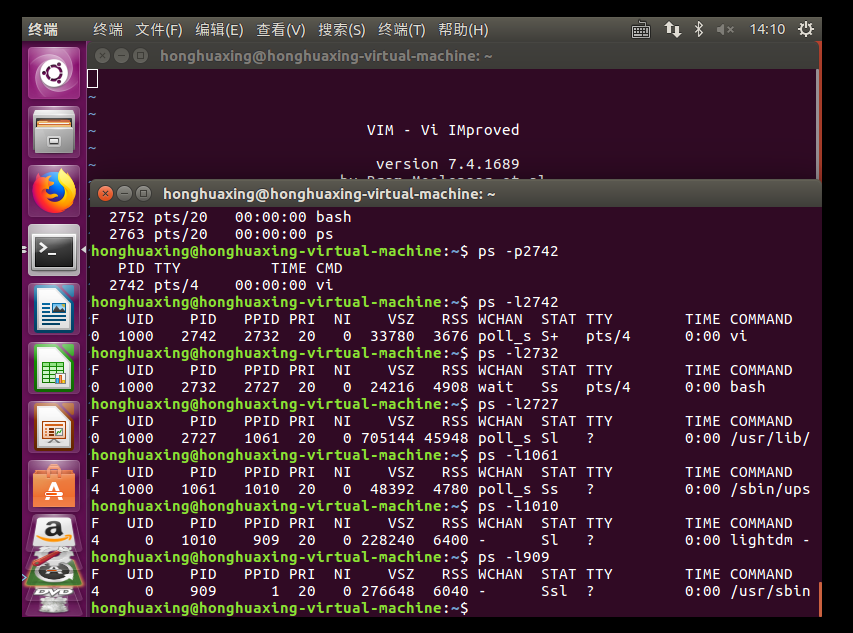
### 解答：先进入终端然后输入vi进入vi界面



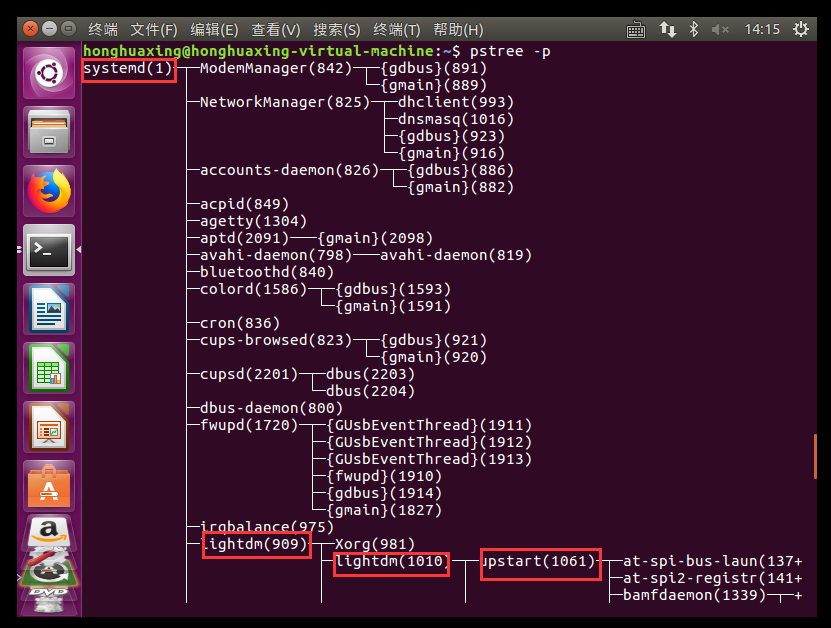
然后再打开另一个终端输入ps-A找到vi界面的进程PID为2742，然后输入ps -l2742找到PPID，后通过ps-l指令逐步查找PPID



最后得到父进程顺序2742->2732->2727->1061->1010->909->1



通过pstree指令得到进程树



 二、编写程序，首先使用fork系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用exec打开vi编辑器。然后在另外一个终端中，通过ps –Al命令、ps aux命令，查看vi进程及其父进程的运行状态，理解每个参数所表达的意义。选择合适的命令参数，对所有进程按照cpu占用率排序。

### 解答：代码如下：

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

pid\_t pid;

pid=fork();

if(pid>0)

while(1);

else if(!pid)

{

int ret;

ret=execlp("vi","",NULL);

if(ret==-1)

perror("execl");

}

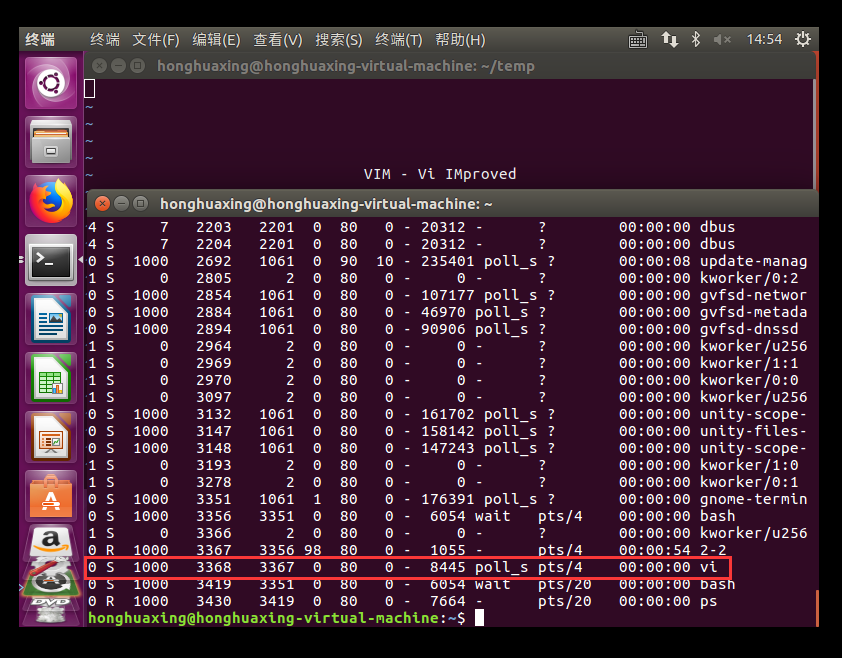
else if(pid==-1)

perror("fork");

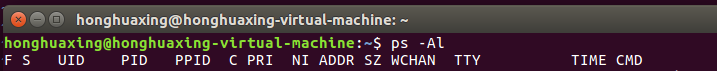
return 0;

}

使用ps -Al查看信息，结果如下：



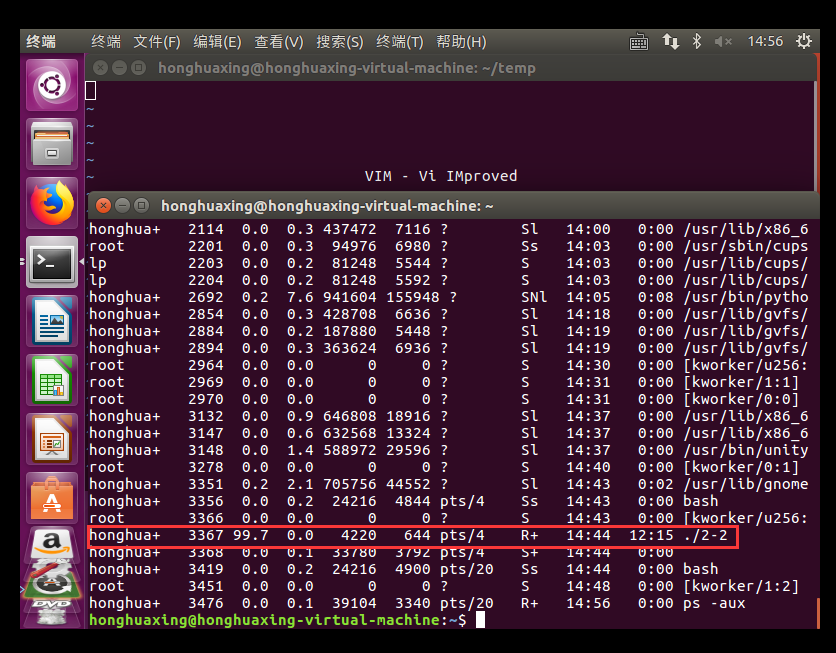
参数如下：



意义：

|  |  |
| --- | --- |
| 参数名 | 意义 |
| F | flag |
| S | stat，状态 |
| UID | 执行者身份 |
| PID | 进程ID |
| PPID | 父进程ID |
| C | CPU使用资源的百分比 |
| PRI | priority，优先级 |
| NI | Nice值 |
| ADDR | 核心功能，指出该进程在内存的哪一部分 |
| SZ | 用掉的内存的大小 |
| WCHAN | 当前进程是否正在运行，若为“-”表示正在进行 |
| TTY | 登陆者的终端位置 |
| TIME | 用掉的CPU的时间 |
| CMD | 所执行的指令 |

使用ps -aux查看信息，结果如下：



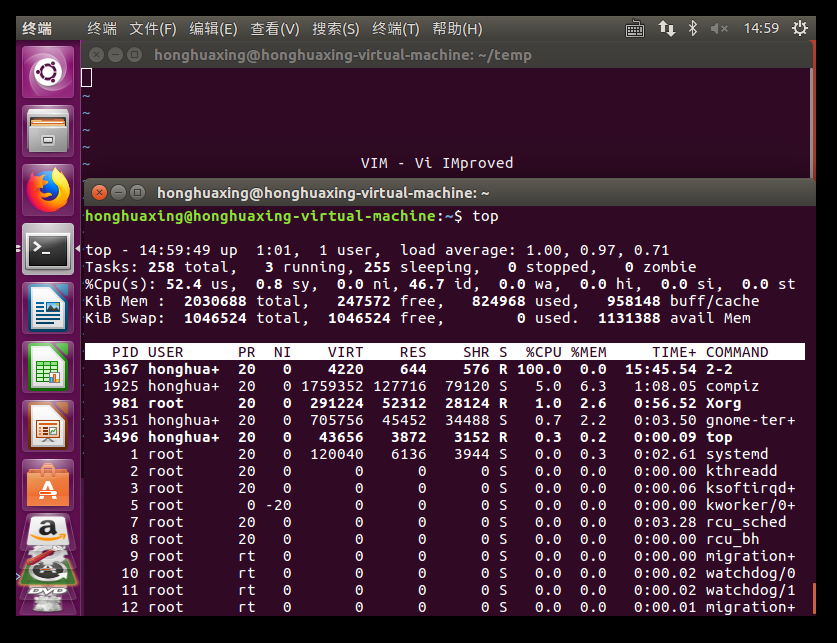
参数如下:



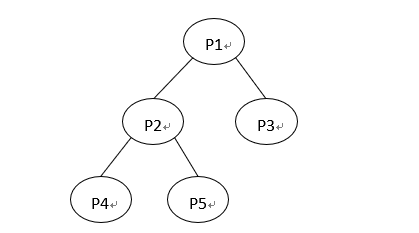
意义：

|  |  |
| --- | --- |
| 参数名 | 意义 |
| USER | 进程的属主 |
| PID | 进程ID |
| %CPU | 进程占用的CPU百分比 |
| %MEM | 占用内存的百分比 |
| VSZ | 发进程使用的虚拟内存量（KB） |
| RSS | 占用的记忆体大小 |
| TTY | 终端的次要装置号码 |
| STAT | 该行程的状态(D=不可中断的睡眠状态,R=运行,S=睡眠,T=跟踪/停止,Z=僵尸进程) |
| START | 该进程的启动时间 |
| TIME | 占用CPU的时间 |
| COMMAND | 命令的名称和参数 |

使用top指令对CPU占有量进行降序排序



 三、使用fork系统调用，创建如下进程树，并使每个进程输出自己的ID和父进程的ID。观察进程的执行顺序。



### 解答：代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

int main(void) {

int p1,p2,p3,p4,p5;

while ((p1=fork())==-1);

if (!p1) {

printf("p1 pid %d, ppid %d.\n",getpid(),getppid());

while ((p2=fork())==-1);

if (!p2) {

printf("p2 pid %d, ppid %d.\n",getpid(),getppid());

while((p4=fork())==-1);

if(!p4){

printf("p4 pid %d, ppid %d.\n",getpid(),getppid());

exit(0);

}

else

wait(0);

while((p5=fork())==-1);

if(!p5){

printf("p5 pid %d, ppid %d.\n",getpid(),getppid());

exit(0);

}

else

wait(0);

exit(0);

}

else

wait(0);

while ((p3=fork())==-1);

if (!p3) {

printf("p3 pid %d, ppid %d.\n",getpid(),getppid());

exit(0);

}

else

wait(0);

exit(0);

}

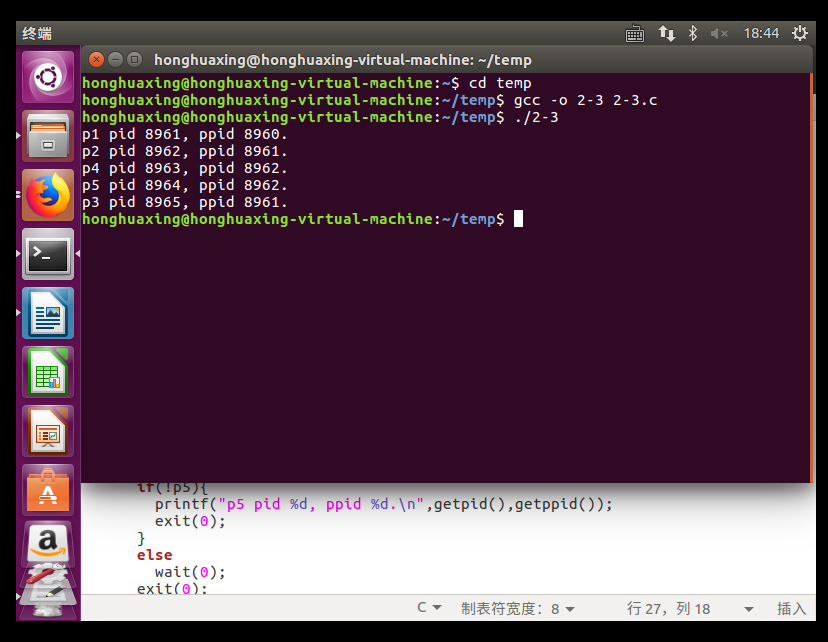
else

wait(0);

return 0;

}

输出结果如下：



 四、修改上述进程树中的进程，使得所有进程都循环输出自己的ID和父进程的ID。然后终止p2进程(kill或者自动退出)，观察p1、p3、p4、p5进程的运行状态和其他相关参数有何改变。

### 解答：代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

int main(void) {

pid\_t p1,p2,p3,p4,p5;

int i=0;

while ((p1=fork())==-1);

if (!p1)

{

while ((p2=fork())==-1);

if (!p2)

{

while((p4=fork())==-1);

if(!p4)

{

while(1)

{

printf("p4 pid %d, parent pid %d.\n",getpid(),getppid());

sleep(2);

}

}

else

{

while((p5=fork())==-1);

if(!p5)

{

while(1)

{

printf("p5 pid %d, parent pid %d.\n",getpid(),getppid());

sleep(2);

}

}

}

while(1)

{

printf("p2 pid %d, parent pid %d.\n",getpid(),getppid());

/\*i++;

if(i==2)

{

int \*p=NULL;

\*p=0;

}\*/ //段地址错误

/\*i++;

if(i==2)

exit(0);\*/ //exit（）函数

sleep(2);

}

}

else

{

while ((p3=fork())==-1);

if (!p3)

{

while(1)

{

printf("p3 pid %d, parent pid %d.\n",getpid(),getppid());

sleep(2);

}

}

}

while(1)

{

printf("p1 pid %d, parent pid %d.\n",getpid(),getppid());

sleep(2);

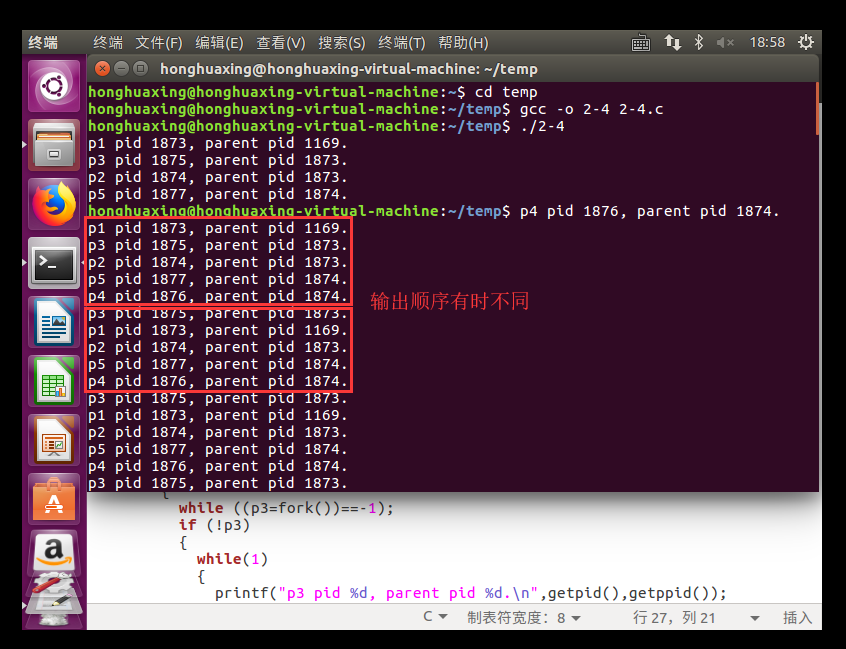
}

}

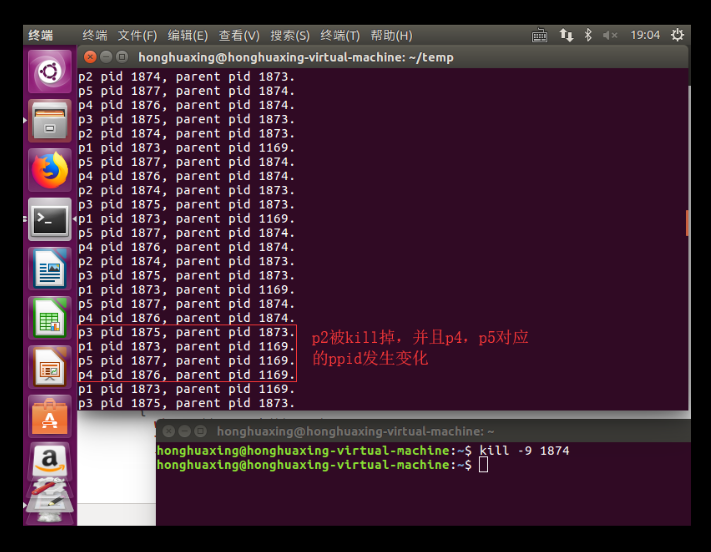
return 0;

}

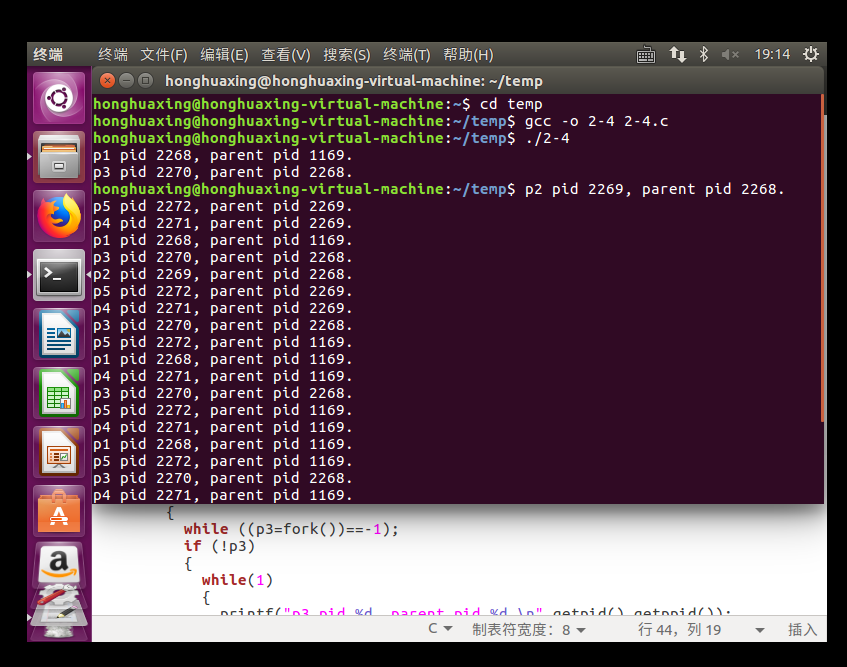
输出结果如下：



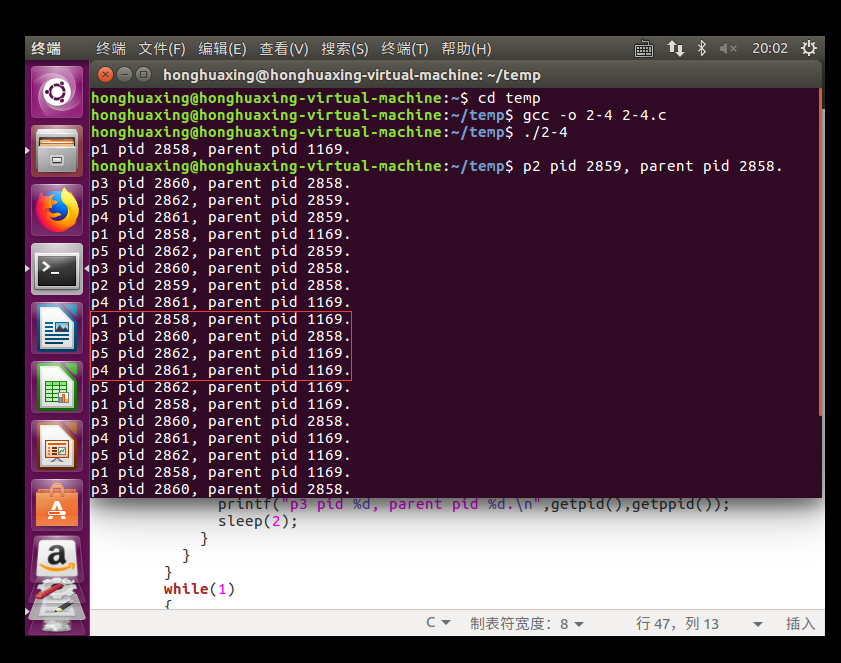
使用kill -9手动中断进程



使用exit（）函数中断进程



段错误，即访问了非法内存或未申请的内存产生的错误



通过这些方式杀死进程2后，可以发现进程2已经不存在，但它的子程序进程4和进程5仍然存在，并且完成输出信息的任务，但其PPID已经变成了进程1的PPID。