**北 京 林 业 大 学**

**2017学年—2018学年第 二 学期 Linux应用实验报告书**

专 业： 计算机科学与技术(创新实验班) 班 级： 计创16

姓 名： 陈楠 学 号： 161002107

实验地点： 计算中心N09 任课教师： 李群

实验题目： Linux进程管理

实验环境： Linux操作系统

实验目的、实现内容、实验结果及结论分析等：

一．实验目的：

1. 掌握Linux前后与后台进程控制操作；

2. 掌握利用进程监控工具来维护系统的正常运行；

3. 掌握Linux的进程创建，理解进程创建后两个并发进程的执行。

二．实验内容：

1.管理进程：

1. 打开终端，输入命令“yes”，然后空格输入“test process”后回车。命令的执行结果是重复打印字符串“test process”。按键“Ctrl+z”中断，暂停前台进程的运行，转为挂起进程。

输入命令“jobs”，记录看到的结果。

1. 把挂起的进程转为前台进程执行：“fg 1”

再次按键“Ctrl+z”中断，重新挂起进程

输入命令“jobs -l”，记录看到的结果，进程号是多少？

1. 把挂起的进程以后台方式运行：“bg 1”

此时能否再次按键“Ctrl+z”中断？为什么？

用鼠标点击关闭中断窗口。

1. 打开终端，输入命令“xclock &”，记录下时钟时间。

输入命令“xcalc &”

输入命令“jobs ”，记录看到的结果；

输入命令“ps”，记录看到的结果；

输入命令“fg %1” 将时钟进程转到前台运行；

按键“Ctrl+z” 将时钟进程挂起，记录时钟的时间，经过二～三分钟，观察时钟有走动吗？

1. 输入命令“kill %2”杀死计算器进程，看计算器是否消失？
2. 查看进程树，并在进程树中显示进程号（pstree –p）。
3. 杀死bash进程，发生了什么？（kill -9 (进程号)）
4. 图形界面下的进程控制和系统维护：
   1. 点击“主菜单/系统工具/系统监视器”；
   2. 查看标签页“进程列表”，观察各进程的状态；

注意观察进程名、用户、内存、%CPU、ID等选项；点击各项目旁“▼”或“▲”按钮，降序或升序排列。

* 1. 点击“进程列表”下的“查看”中的“活动的进程”；
  2. 切换到终端键入命令“yes test process”
  3. 切换到“进程列表”窗口，观察“活动的进程”有什么变化？
  4. 点击标签页“系统监视器”，观察“%CPU使用历史”、“内存/交换使用历史”、“设备”项目内容；
  5. 此时CPU的利用率很高，是由哪个进程引起的？
  6. 再次查看标签页“进程列表”中的“活动进程”，鼠标选中那个非常“活跃”的进程，右键快捷方式杀死该进程。

2. 进程创建

（1）编写一段程序，实现父进程创建一个子进程，返回后父子进程分别循环输出进程号，例如“I am parent ,my ID is……”及“I am child ,my ID is ……”循环3次，每次输出后使用sleep(1)延时1秒，然后再进入下一次循环，观察程序运行的结果。

（2）修改上述程序，使用exit()和wait()实现父进程和子进程同步，其同步方式为父进程等待子进程的同步，即：子进程循环输出3次，然后父进程再循环输出3次。观察程序运行的结果。

实验指导：

本实验相关函数：

1. fork( )函数

fork()函数创建一个新进程。

其中返回值int取值：

等于0：创建子进程，从子进程返回的ID值；

大于0：从父进程返回子进程的ID值。

等于-1：进程创建失败。

1. sleep（时间）, 作用是延时，程序暂停若干时间。

Linux下（使用的gcc的库），sleep()函数是以秒为单位。

1. wait（ ）函数，常用来控制父进程与子进程的同步。父进程中调用wait()函数，则父进程被阻塞，进入等待队列，等待子进程结束。当子进程结束，会产生一个终止状态字，系统会向父进程发出信号。当接到信号后，父进程提取子进程的终止状态字，从wait()函数返回继续执行原程序。

系统调用格式：

int wait(status)

int \*status;

正确返回：大于0：子进程的进程ID值；

等于0：其它。

错误返回：等于-1：调用失败。

1. exit（ ），终止进程的执行。

调用方式：

　　 　exit(status)

　　 int status;

其中，status是返回给父进程的一个整数，以备查考。

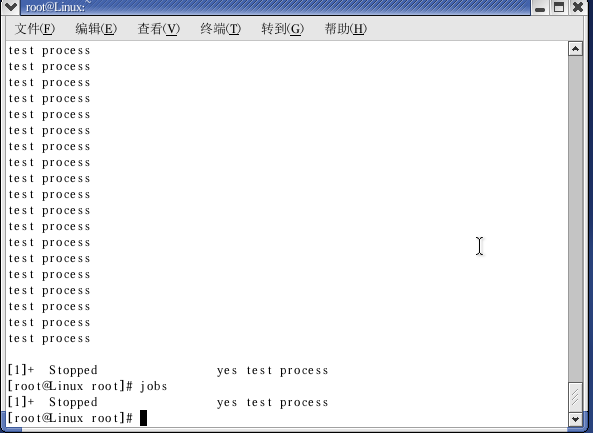
1. 通常父进程在创建子进程时，应在进程的末尾安排一条exit( )，使子进程自我终止。exit(0)表示进程正常终止，exit(1)表示进程运行有错，异常终止。

三．实验结果：

1.管理进程：

1. 打开终端，输入命令“yes”，然后空格输入“test process”后回车。命令的执行结果是重复打印字符串“test process”。按键“Ctrl+z”中断，暂停前台进程的运行，转为挂起进程。

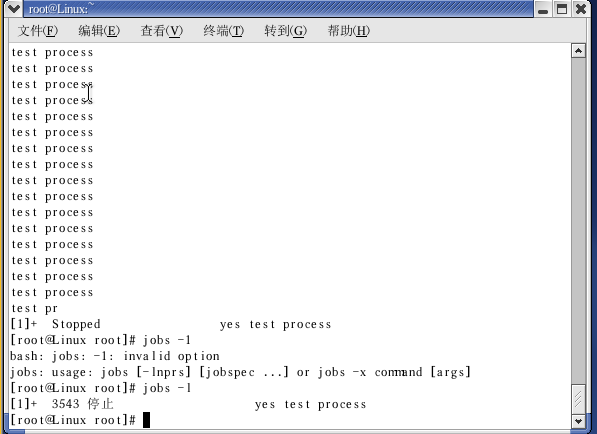
输入命令“jobs”，记录看到的结果。



1. 把挂起的进程转为前台进程执行：“fg 1”

再次按键“Ctrl+z”中断，重新挂起进程

输入命令“jobs -l”，记录看到的结果，进程号是多少？



进程号是3543

1. 把挂起的进程以后台方式运行：“bg 1”

此时能否再次按键“Ctrl+z”中断？为什么？

不能用“Ctrl+z”中断。

因为“Ctrl+z”只能将一个正在前台执行的命令挂起

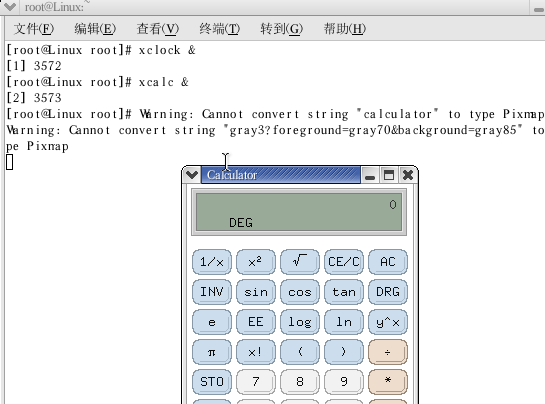
用鼠标点击关闭中断窗口。



1. 打开终端，输入命令“xclock &”，记录下时钟时间。



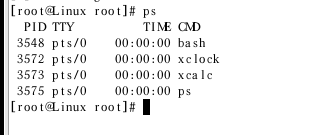
输入命令“xcalc &”



输入命令“jobs ”，记录看到的结果；



输入命令“ps”，记录看到的结果；



输入命令“fg %1” 将时钟进程转到前台运行；



按键“Ctrl+z” 将时钟进程挂起，记录时钟的时间，经过二～三分钟，观察时钟有走动吗？



没有走动

1. 输入命令“kill %2”杀死计算器进程，看计算器是否消失？



消失了

1. 查看进程树，并在进程树中显示进程号（pstree –p）。



1. 杀死bash进程，发生了什么？（kill -9 (进程号)）



终端消失了

1. 图形界面下的进程控制和系统维护：
   1. 点击“主菜单/系统工具/系统监视器”；
   2. 查看标签页“进程列表”，观察各进程的状态；

注意观察进程名、用户、内存、%CPU、ID等选项；点击各项目旁“▼”或“▲”按钮，降序或升序排列。

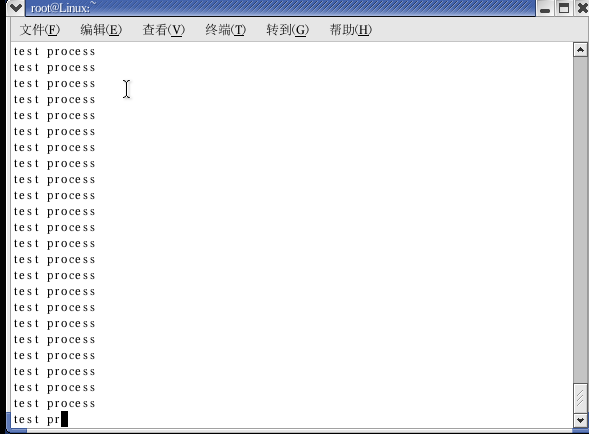




* 1. 点击“进程列表”下的“查看”中的“活动的进程”；



* 1. 切换到终端键入命令“yes test process”

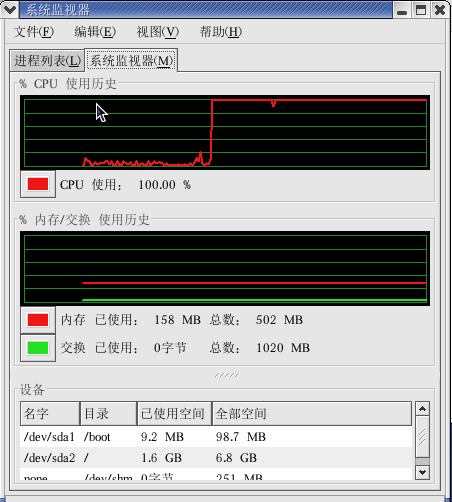


* 1. 切换到“进程列表”窗口，观察“活动的进程”有什么变化？

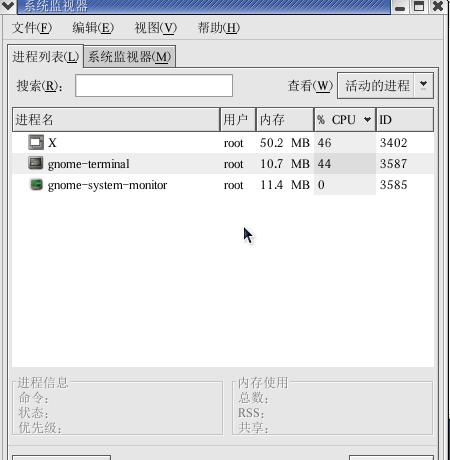


多了几个进程

* 1. 点击标签页“系统监视器”，观察“%CPU使用历史”、“内存/交换使用历史”、“设备”项目内容；



* 1. 此时CPU的利用率很高，是由哪个进程引起的？



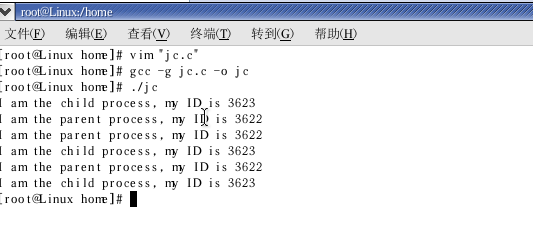
* 1. 再次查看标签页“进程列表”中的“活动进程”，鼠标选中那个非常“活跃”的进程，右键快捷方式杀死该进程。

杀死后 test process 窗口消失

2. 进程创建

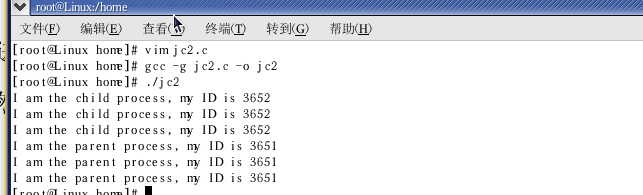
（1）编写一段程序，实现父进程创建一个子进程，返回后父子进程分别循环输出进程号，例如“I am parent ,my ID is……”及“I am child ,my ID is ……”循环3次，每次输出后使用sleep(1)延时1秒，然后再进入下一次循环，观察程序运行的结果。





（2）修改上述程序，使用exit()和wait()实现父进程和子进程同步，其同步方式为父进程等待子进程的同步，即：子进程循环输出3次，然后父进程再循环输出3次。观察程序运行的结果。





实验指导：

本实验相关函数：

1. fork( )函数

fork()函数创建一个新进程。

其中返回值int取值：

等于0：创建子进程，从子进程返回的ID值；

大于0：从父进程返回子进程的ID值。

等于-1：进程创建失败。

1. sleep（时间）, 作用是延时，程序暂停若干时间。

Linux下（使用的gcc的库），sleep()函数是以秒为单位。

1. wait（ ）函数，常用来控制父进程与子进程的同步。父进程中调用wait()函数，则父进程被阻塞，进入等待队列，等待子进程结束。当子进程结束，会产生一个终止状态字，系统会向父进程发出信号。当接到信号后，父进程提取子进程的终止状态字，从wait()函数返回继续执行原程序。

系统调用格式：

int wait(status)

int \*status;

正确返回：大于0：子进程的进程ID值；

等于0：其它。

错误返回：等于-1：调用失败。

1. exit（ ），终止进程的执行。

调用方式：

　　 　exit(status)

　　 int status;

其中，status是返回给父进程的一个整数，以备查考。

1. 通常父进程在创建子进程时，应在进程的末尾安排一条exit( )，使子进程自我终止。exit(0)表示进程正常终止，exit(1)表示进程运行有错，异常终止。

四．结论分析：

通过本次Linux进程管理实验，我对Linux下的进程管理有了更加深入地理解。掌握了Linux前后与后台进程控制操作；初步掌握并能够利用进程监控工具来维护系统的正常运行；尝试了Linux的进程创建，初步理解了进程创建后两个并发进程的执行过程。在Linux中，相关操作的具体实现只会更加复杂，因为要考虑的东西很多，所以需要我们更加努力的学习。