**操作系统课程设计实验报告**

——实验二：作业调度

负责人姓名：史天泽

学号：14061121

日期：2016.4.10

**小组成员**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 姓名 | 学号 | 实验分工 |
| 1 | 崔博涵 | 14061127 | 修改bug |
| 2 | 史天泽 | 14061121 | 实现调试任务，stat输出，编写实验报告 |
| 3 | 孟祥鑫 | 14061135 | 修改bug |
| 4 | 韩世依 | 14061129 | 修改bug，实现多级反馈轮转 |

目录

[1.实验目的 4](#_Toc446001831)

[2.需求说明 4](#_Toc446001832)

[2.1基本要求 4](#_Toc446001833)

[2.2 提高要求 4](#_Toc446001834)

[2.3 完成情况 5](#_Toc446001835)

[3.设计说明 6](#_Toc446001836)

[3.1 程序流程图 6](#_Toc446001837)

[3.2基本要求实现说明 6](#_Toc446001838)

[3.3 提高要求实现说明 8](#_Toc446001839)

[4.收获和感想 2](#_Toc446001840)2

[5.例会记录 2](#_Toc446001840)3

# 1.实验目的

# 理解进程调度和作业调度的区别和联系。

# 学习 linux 系统中进程间通信的概念和方方法。

# 理解几几种常用用的调度算法,并能进行行性能分析。

# 能够理解和使用用 linux 给进程调度提供的系统函数。掌握进程控制和进程间通信的方法

# 2.需求说明

## 2.1基本要求

## 1. 实现作业的入队操作

## 2. 实现作业状态命令查看

## 3. 实现出队命令操作

## 2.2 提高要求

## 实现多级反馈的轮转调度算法

## 将stat 命令的就绪队列信息显示在命令终端

## 调试程序状态信息

## 2.3 完成情况

【简述实验完成过程】。完成了以下功能：

## 1. 实现作业的入入队操作

## 2. 实现作业状态命令查看

## 3. 实现出队命令操作

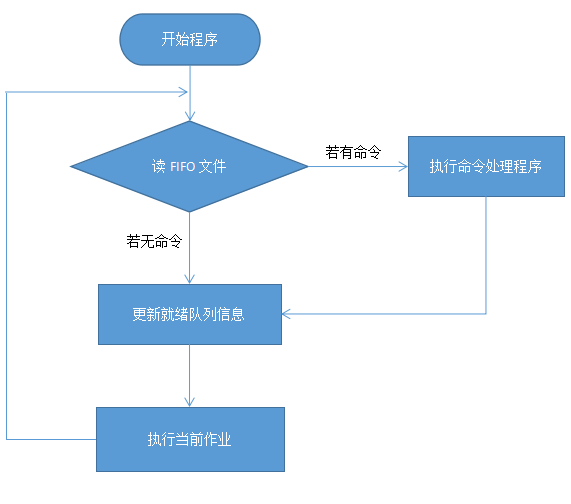
## 实现多级反馈的轮转调度算法

## 将 stat 命令的就绪队列信息显示在命令终端

## 实现调试程序状态信息

# 3.设计说明

## 3.1 程序流程图



## 3.2基本要求实现说明

## 1. 实现作业的入入队操作

格式：enq [-p num] e\_file [args]

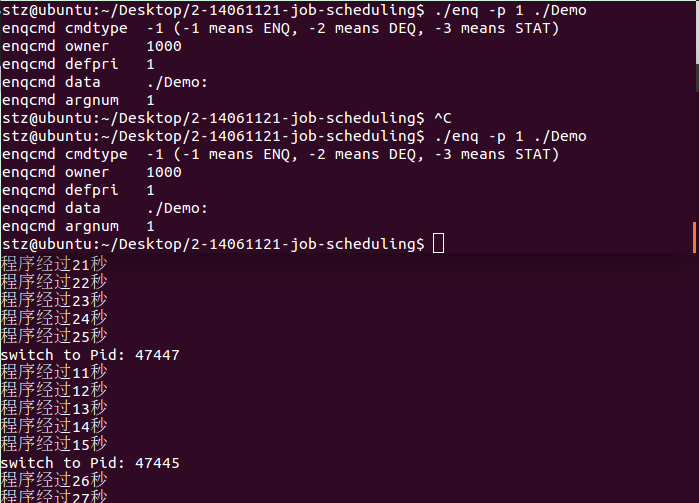
参数说明：

-p num:设定作业的优先级，默认为0（0-3）。

e\_file:启动作业执行的可执行文件

args:e\_file的运行参数

运行结果如下图：



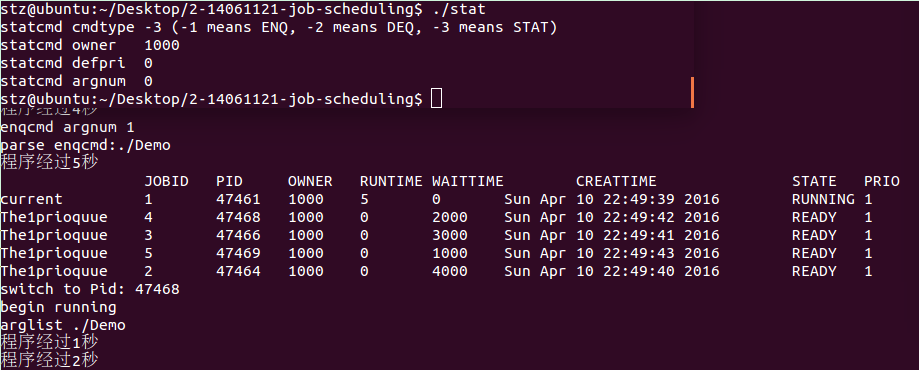
## 2. 实现作业状态命令查看

实现作业状态命令查看 stat:

打印出以下状态信息 作业的jid

作业的提交者 用户名 作业的执行时间 在就绪队列的等待时间 作业的创建时刻 此时作业的状态(READY,RUNNING,DONE)

运行结果如图：

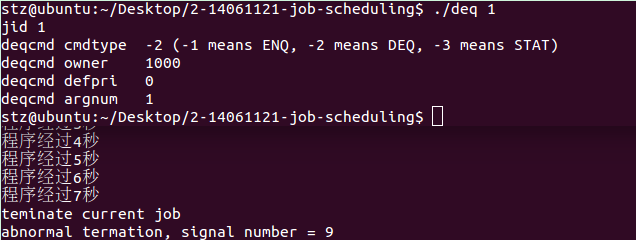


## 3. 实现出队命令操作

实现出队命令操作

deq jid

参数说明: Jid:由作业调度程序分配的作业号

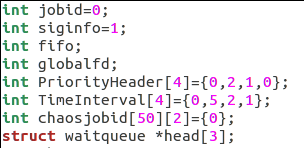


## 3.3 提高要求实现说明

1. 实现多级反馈的轮转调度算法

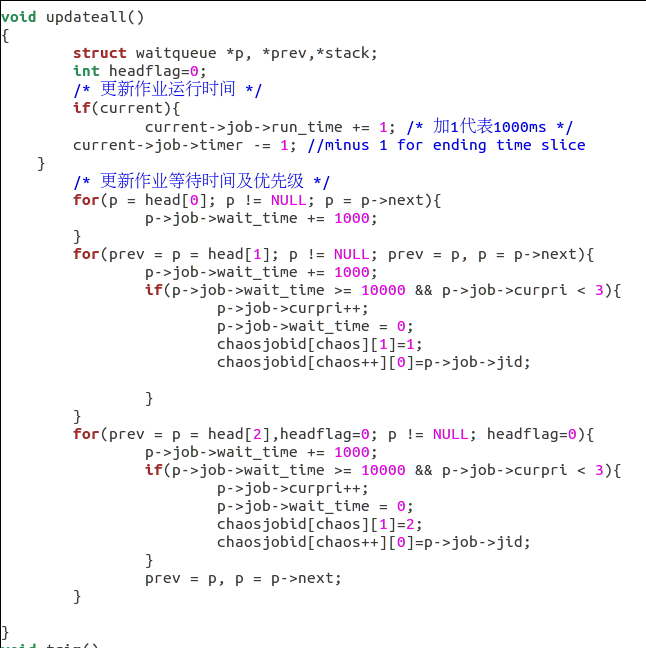
多种优先级的实现：

首先按照要求为等待队列设置3种不同的优先级队列如下图，其中PriorityHeader中的2，1，0为3个优先级，TimeInterval中的5，2，1为三种优先级对应的时间片长度，5秒，2秒，1秒。



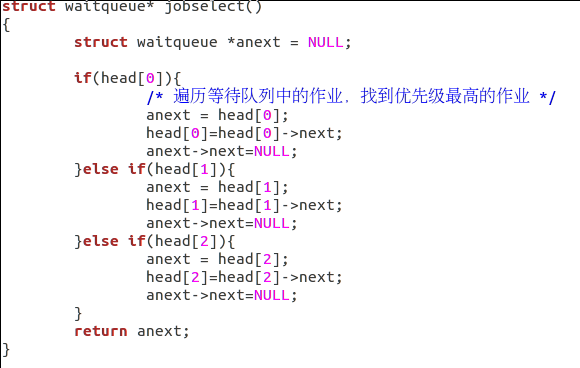
反馈的实现：

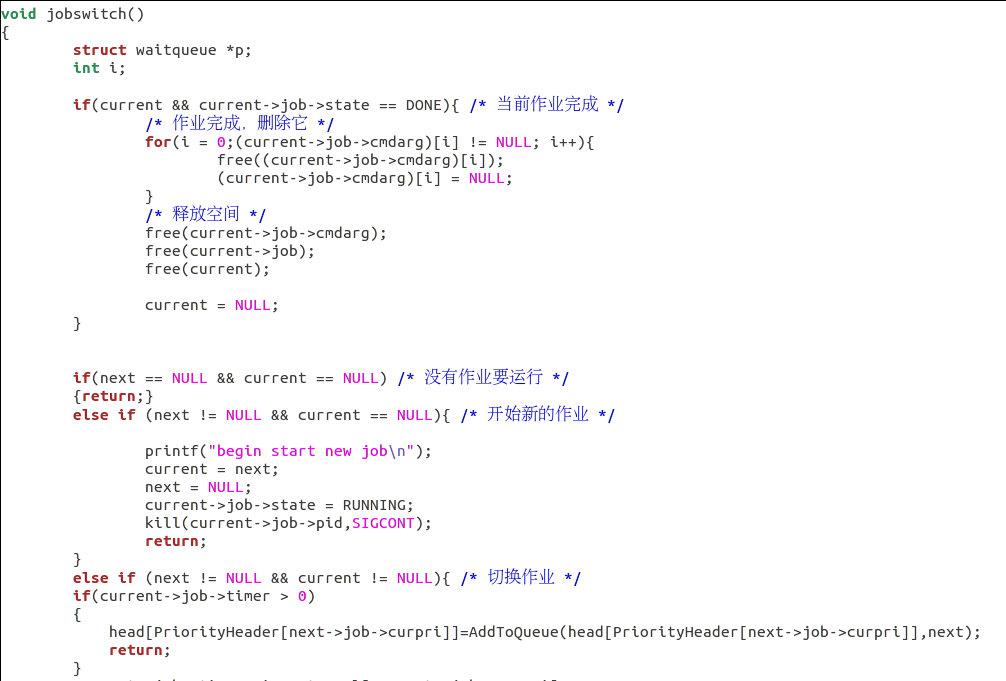
如下图，在updateall函数中，作业的等待时间逐秒增加，每当一个作业的等待时间超过10000ms时，该作业的优先级上升一级。

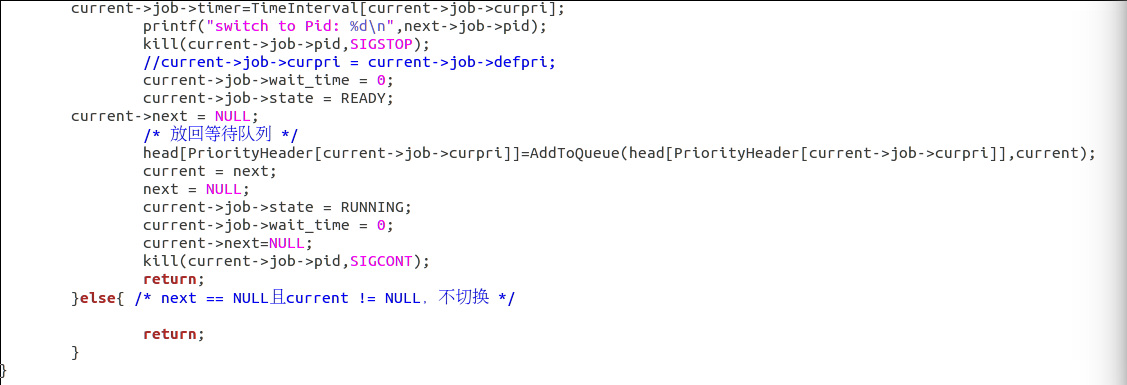


多级轮转：

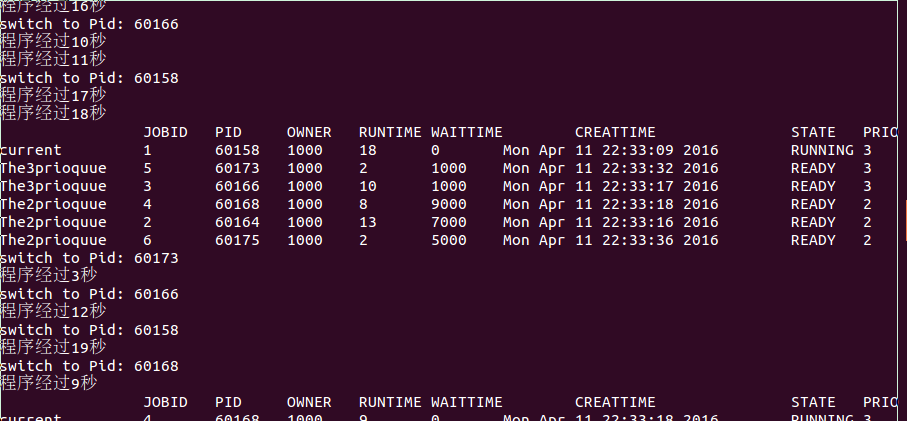
重写后的jobselect如下，由于是多优先级，所以每次轮转优先寻找优先级高的作业，我们还重写了jobswitch函数，通过这两个函数实现了实现多级反馈轮转调度算法







多级轮换的运行截图如下：



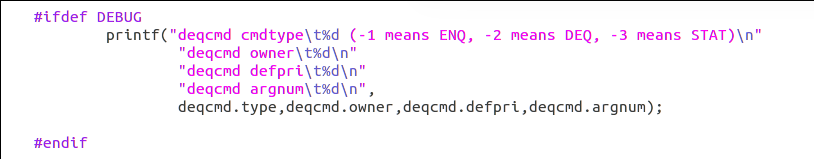
2. 实现调试任务：

调试任务的1~4已经在ppt中给出，这里我们给出任务5到10的实现过程。

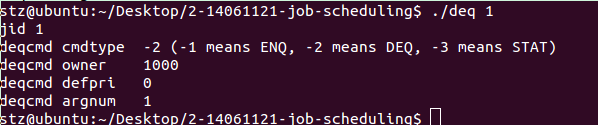
任务五：DEQ命令和STAT命令调试:与enq类似在deq.c和stat.c的适当位置添加调试信息,实现启动命令时可以在当前终端查看对应的cmdtype和owner信息;

1. Deq：

在deq.c的头部加入#define DEBUG, 然后在下边加入如下调试代码：

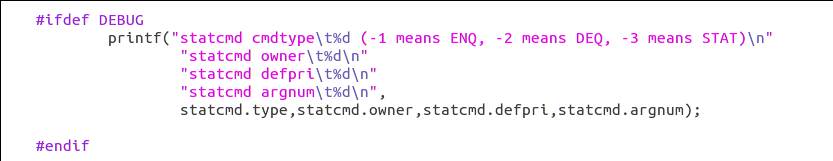


编译后运行deq指令后，结果如下：

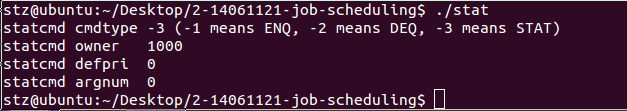


2．Stat

在stat.c的头部加入#define DEBUG, 然后在下边加入如下调试代码：

****

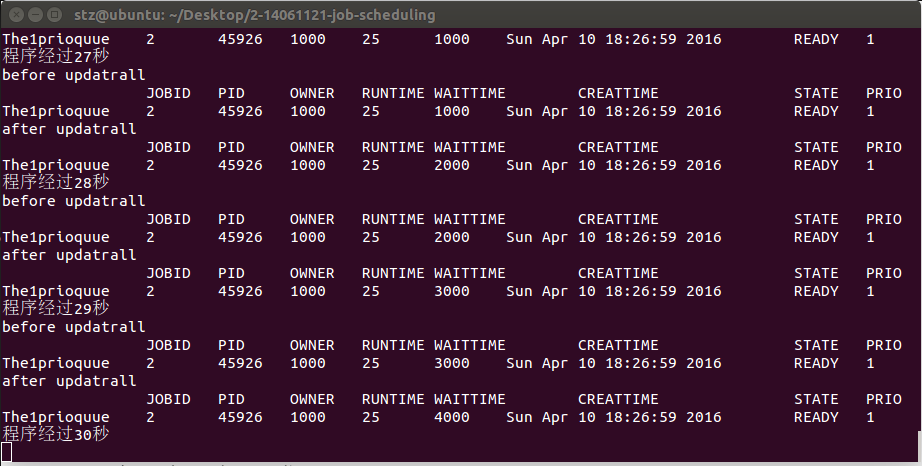
编译后运行stat指令，结果如下：



任务六：updateall函数调试:在job.c里,实现查看updateall函数执行前后队列里所有作业的相关信息;

在updateall函数内，函数段运行前后，各加入一段如下图所示代码，即可查看每次updateall函数执行前后队列里各级作业的相关信息，为了区分updateall函数运行前后，在两段第一句分别加上了printf("before updatrall\n");和printf("after updatrall\n");两句输出，用以区分，执行结果如下图。

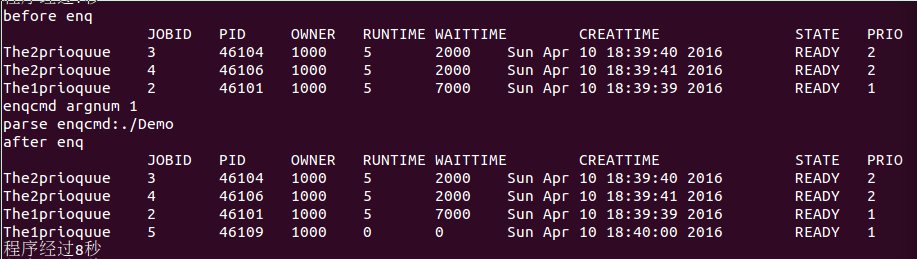




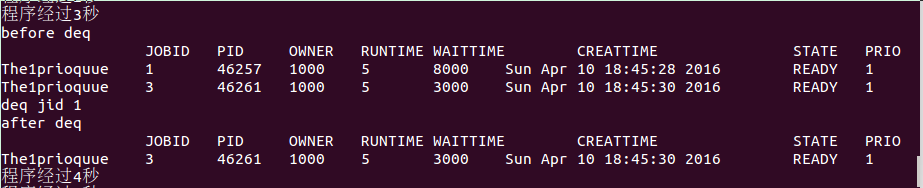
任务七：命令调试:在job.c里,实现显示ENQ,DEQ,STAT命令执行前后的队列⾥里所有作业的相关信息;

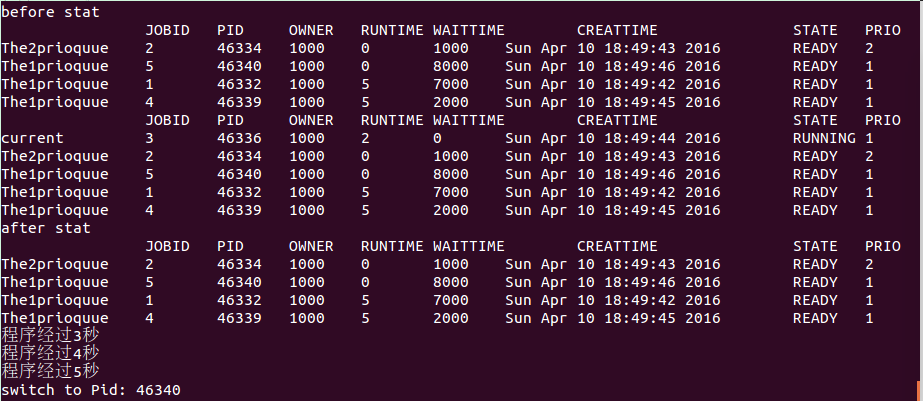
ENQ：

与任务六同理，这次在do\_enq函数内的函数段前后分别加入同任务六的代码，重新编译后，执行结果如下图。



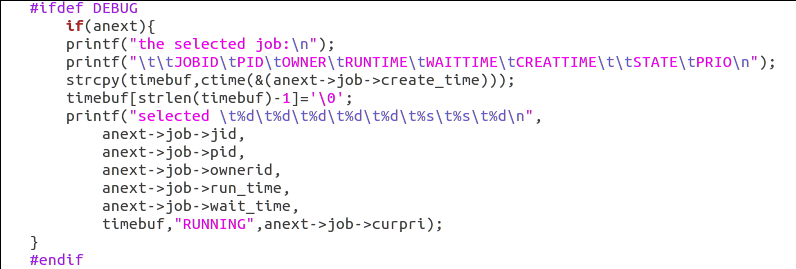
DEQ：

还是同理，这次在do\_deq函数内的函数段前后分别加入同任务六的代码，重新编译后，执行结果如下图。

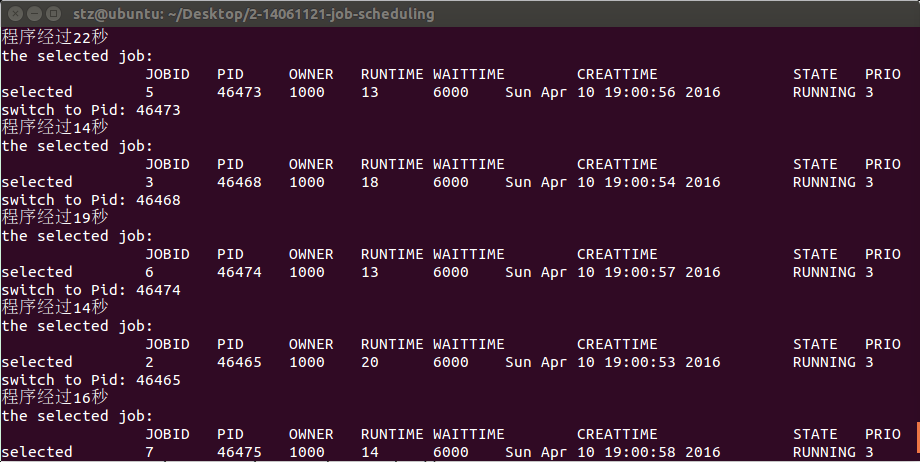
STAT：这次在do\_stat函数内的函数段前后分别加入同任务六的代码，重新编译后，执行结果如下图。

任务八：jobselect函数调试:在job.c里,实现显示jobselect函数选择的作业的信息;

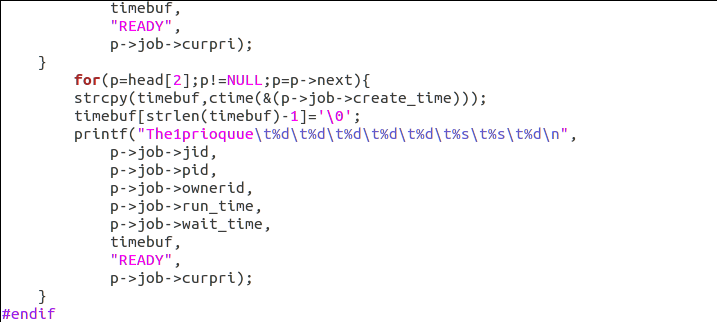
在struct waitqueue\* jobselect()内，在select代码后，加上如下代码。

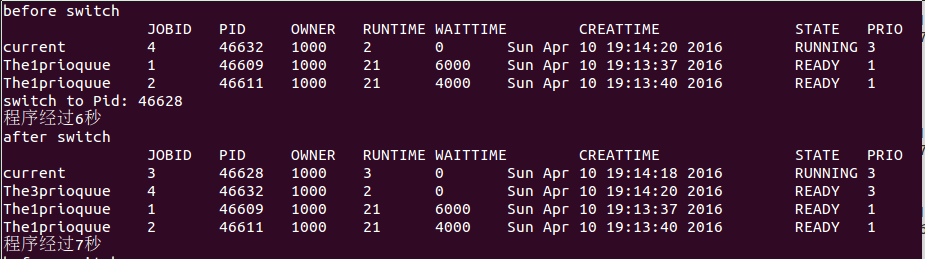


重新编译后，即可显示每次选择的作业信息，结果如下图：



任务九：jobswitch函数调试:在job.c里,实现显示jobswitch执行前后的正在执行的作业和等待队列中的分别的情况；

先在jobswitch中可能切换job的部分开始处加入如下调试代码，结尾处再加一遍，即可看到swtich前后的所有作业的信息。

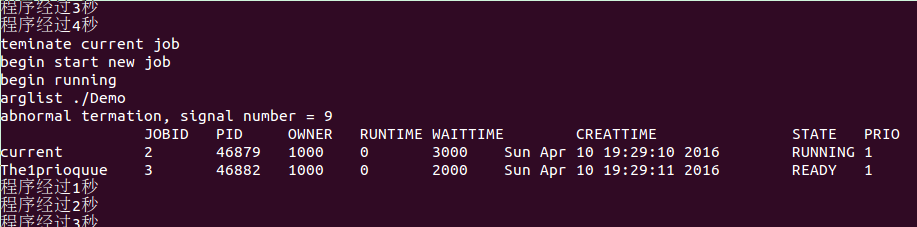
重新编译后，运行结果如下图：

任务十：进程运行完成调试:在job.c里,实现显示当进程自己运行完成后发送 SIGCHLD给sig\_handler后的正在执行的作业和等待队列中的状态;

在sig\_handler函数内如下图位置添加同任务九的代码。

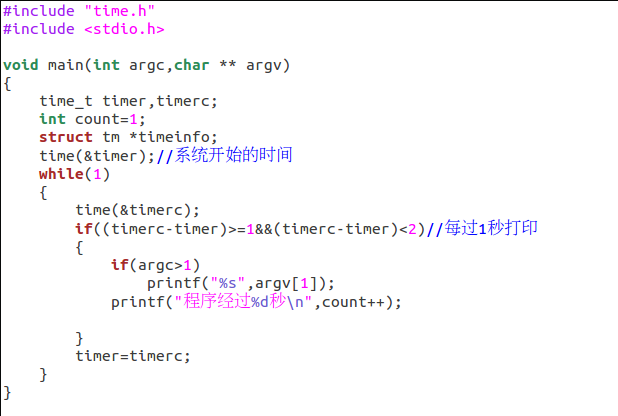


重新编译后，运行结果如下：



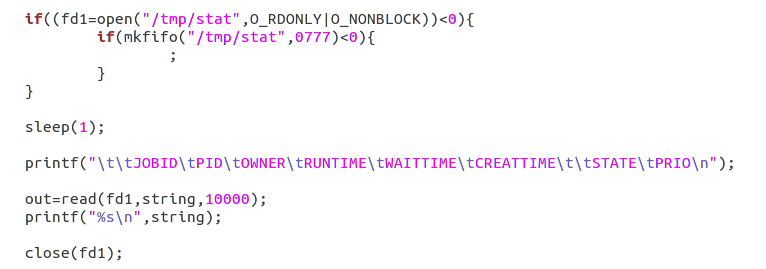
3．对Demo.c的修改：

我们在测试时发现，原来的Demo.c在进行作业轮转时，每次都会多运行1秒，经过阅读代码，我发现是Demo.c的设计有问题，它每次拿当前时间比较上次记录的时间，只要差大与1就输出，这样导致每次轮转回来刚开始时都要多输出一次，经过修改得到以下代码：保证每次只有时间差为1秒时才输出，这样就避免了多输出一秒的问题。



4. stat命令输出：

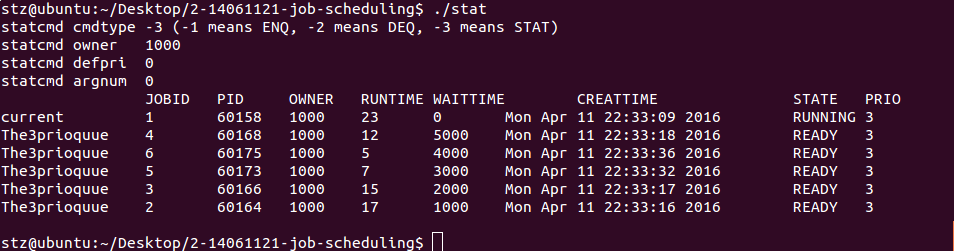
这里我还是按照PPT上的提示，采用FIFO有名管道来传递信息，修改了stat.c和job.c中的do\_stat函数,代码如下：







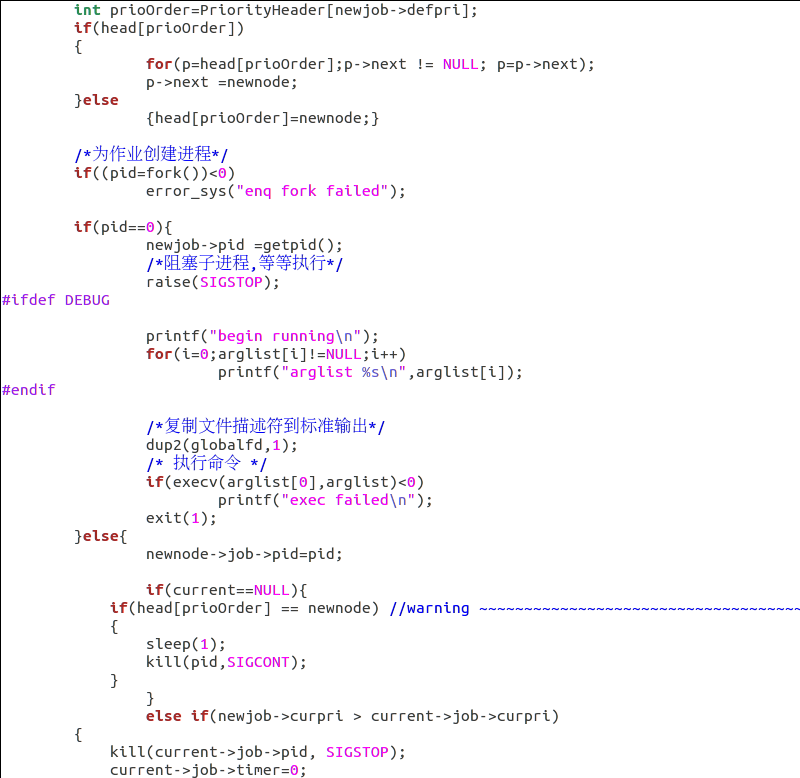
运行结果如下：



5.源程序的bug修改：

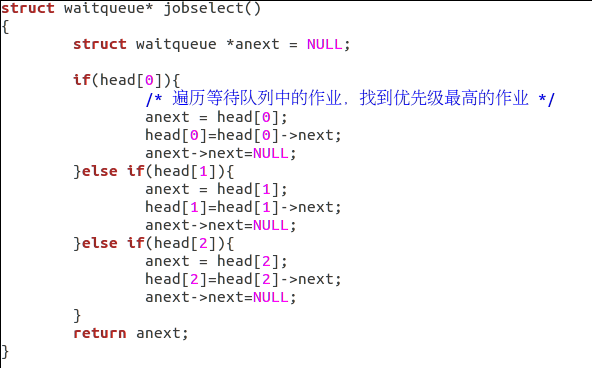
1.只入队一个作业时，作业不能执行：

这是因为在源程序的do\_enq函数中，当链表仅有第一个节点时，没有为其创建进程，所以第一个作业无法执行，先将其修改为如下程序：此时考虑了第一个节点的情况，第一个作业可以正常执行。



1. jobselect函数内，链表指针丢失
2. jobselect函数内，select->next未置空：

我们的jobselect改动很大，基本上是重写的，已经避免了源程序中的链表指针丢失问题，修改后的代码如下：



4.do\_deq函数内链表指针丢失：

修改后的代码如下图所示：





# 4.收获与感想

本次实验我们学习了linux的作业调度，理解了作业与进程的区别，并了解了进程调度和作业调度的区别和联系，在更改stat输出方式时，了解并实践了使用FIFO实现linux进程间的通信，尝试了对原来的调度程序进行更改，同时还能够使用linux给进程调度提供的系统函数。同时助教也为我们实验提供了很多帮助，感觉这次实验收获良多。

# 5.例会记录

第一次例会，4月7日，我们发现了enq和deq命令的一些bug，大家讨论了一下解决bug的方法。并且将其修改了，还在此探讨了一下本次实验的主要任务，分析了一下多级反馈轮转调度的实现方法。

第二次例会，4月11日，实现了多级反馈的轮转调度算法，主要实现了3个队列，优先级分别为1、2、3，输入的时候把优先级写上，没写的默认是1。调度的时候先从优先级高的而且排在前面的队列中拿作业，依次往后排，然后还写了更新时钟的函数。并且完成了将stat命令的就绪队列信息显示在命令终端，还实现了调试任务。实验已经基本做完。