操作系统实验报告：实验二

姓名 王喆 学号 13061074

沈明飞 13061067 赵乐 13061076

郭奕鑫 13061088 余旭 13061090

# 1需求说明

## 1.1基本要求和提高要求

**基本要求**

本实验要求实现一个作业调度程序，通过该程序可以完成作业的入队、出队、查看和调度。具体要求如下：

（1）实现作业调度程序scheduler，负责整个系统的运行。

这是一个无限循环运行的进程，其任务是响应作业的入队、出队以及状态查看请求，采用适当的算法调度各作业运行。

（2）实现作业入队命令。

用户通过该命令给scheduler发送入队请求，将作业交给系统运行。每一个作业提交以后，若创建成功，scheduler都将为其分配一个唯一标识jid。Scheduler调度程序为每个作业创建一个进程，并将其状态置为READY，然后放入就绪队列中，打印作业信息。

（3）实现作业出队命令

用户通过该命令给scheduler发送出队请求，scheduler将使该作业出队，然后清除相关的数据结构。若该作业正在运行，则必须先终止其运行。每个用户都只能杀掉（kill）自己提交的作业。

（4）实现作业状态查看命令。

在标准输出上打印出就绪队列中各作业的信息。状态信息应该包括：

* + 作业的jid
  + 作业提交者用户名
  + 作业执行的时间
  + 在就绪队列中的等待时间
  + 作业创建的时刻
  + 此时作业的状态（READY\RUNNING\DONE）

（5）实现多级反馈的轮转调度算法。

每个作业有其动态的优先级，在用完分配的时间片后，可以被优先级更高的作业抢占运行。就绪队列中的进程等待时间越长，其优先级越高。每个作业都具有以下两种优先级：

* + - * 初始优先级 在作业提交时指定，将保持不变，直至作业结束。
      * 当前优先级 由scheduler调度更新，用以调度作业运行。Scheduler总是选择当前优先级最高的那个作业来运行。

作业当前优先级的更新主要取决于以下两种情况：

* + 一个作业在就绪队列中等待了若干各时间片，则将它的当前优先级+1。
  + 若当前运行的作业时间片到，则中止该作业的运行（抢占式多任务），将其放入就绪队列中，它的当前优先级也恢复为初始优先级。

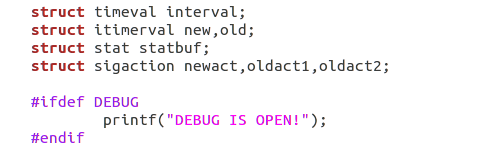
通过这样的反馈处理，使得每个作业都有执行的机会，避免了使低优先级的作业拖延而不能执行的情况发生。

出于简单的目的，假设只考虑作业的以下三种状态：

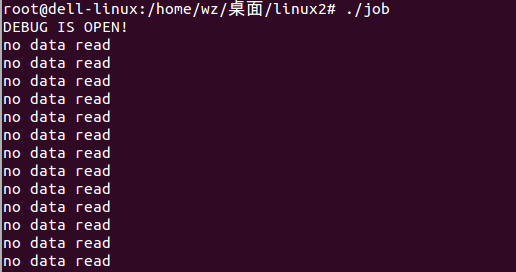
* READY 就绪状态，该作业在就绪队列（ready queue）中等待调度。
* RUNNING 运行状态，该作业正在运行。
* DONE 运行结束，该作业已完成运行。

**调试任务**

**任务一**

在job.c的main函数中，所有的struct声明之后，添加调试代码，如下所示：

重新编译job.c文件，再运行./job，得到下图结果



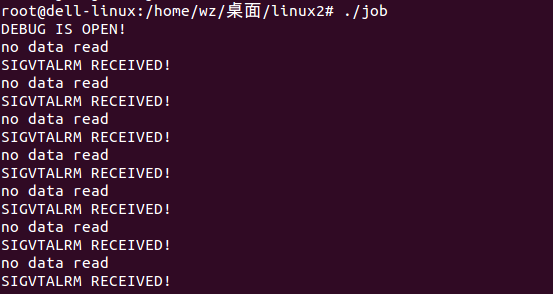
任务二

定时处理函数sig\_handler()调试;

定时处理函数sig\_handler()每一秒中接收来自“时钟函数setitimer()的信号SIGVTALRM”或者“子进程结束时给父进程发出的信号SIGCHLD”，根据接受信号的不同进行不同的case语句操作，例如进行进程调度函数scheduler()等。

在case SIGVTALRM的执行语句加入调试代码，如下所示

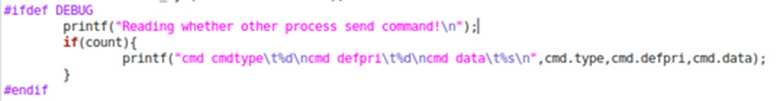
重新编译文件并执行，得到结果如下：每一秒钟会接受到一个SIGVTALRM信号



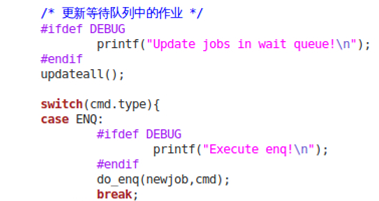
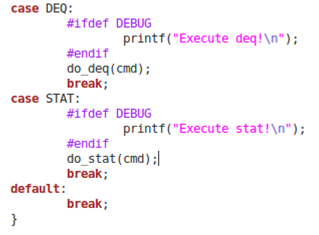
任务三

调度程序scheduler()调试；

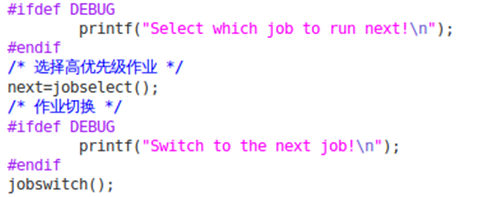
进程通信阶段的调试代码：



更新作业阶段的调试代码:

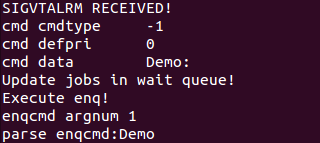
 

选择和切换作业阶段的调试代码:

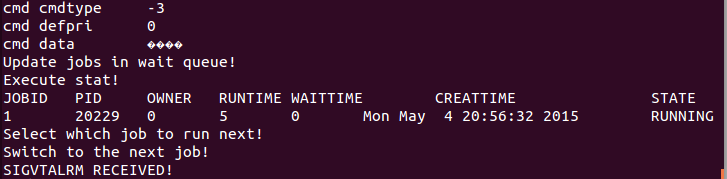


重新编译文件并执行，得到结果如下：可以查看到每一秒作业调度程序的执行结果。

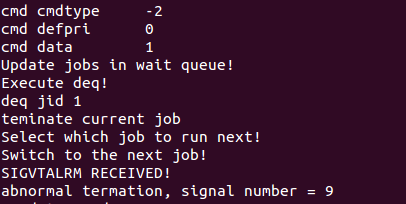
执行enq：



执行stat：



执行deq：

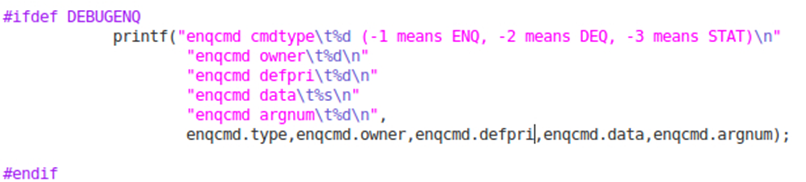


任务四

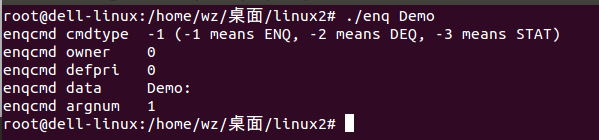
作业入队命令调试;

先删除job.c文件中的#define DEBUG，关闭之前的调试方法；

在enq.c的头部加入#define DEBUG，并将源码中“输出调试信息”的代码修改如下：

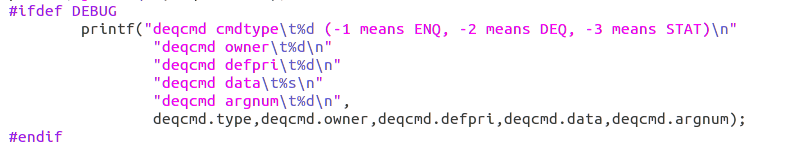


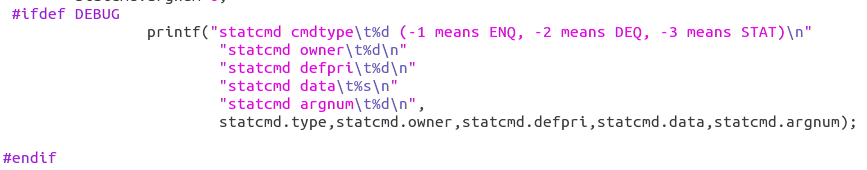
重新编译，并运行作业入队 命令，结果如下

****

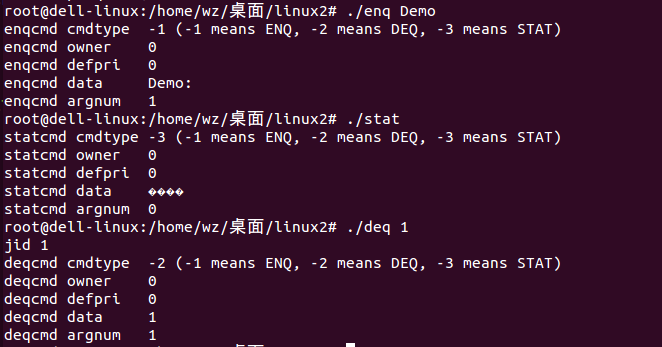
任务五

DEQ命令和STAT命令调试：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”



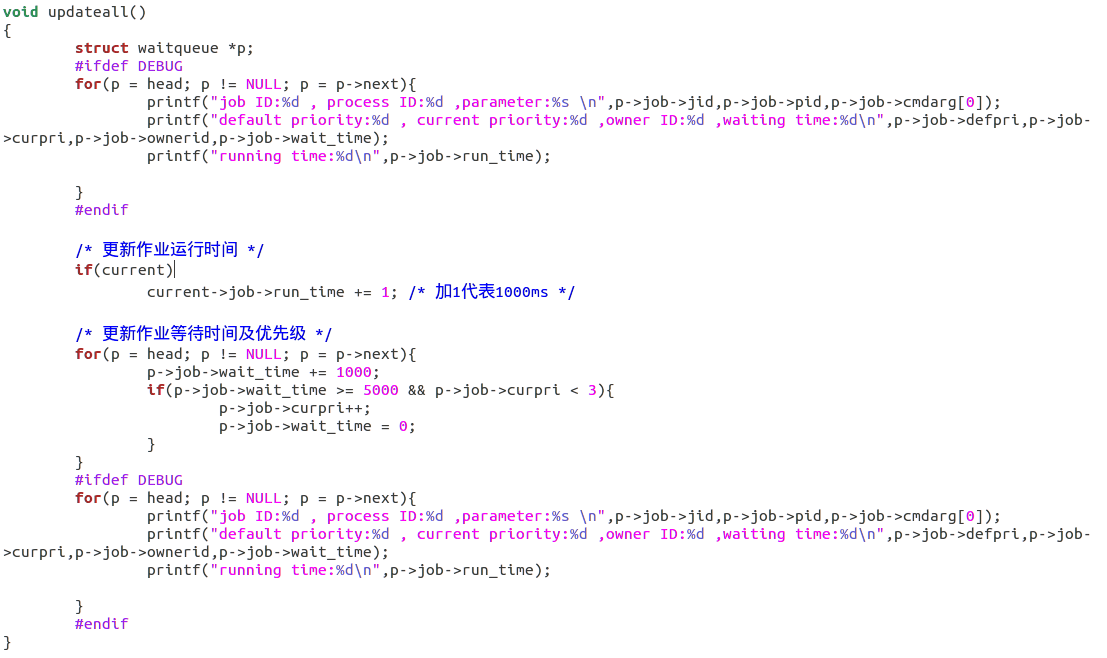


重新编译并执行，结果如下：

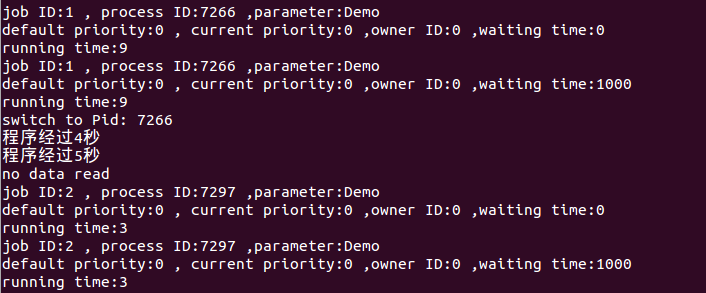


任务六

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进程的相关信息”；



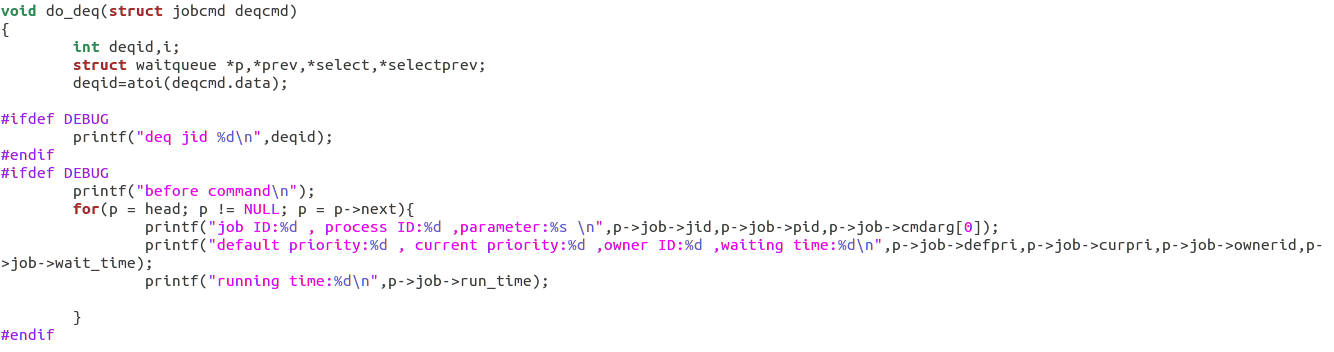
重新编译并执行，结果如下：

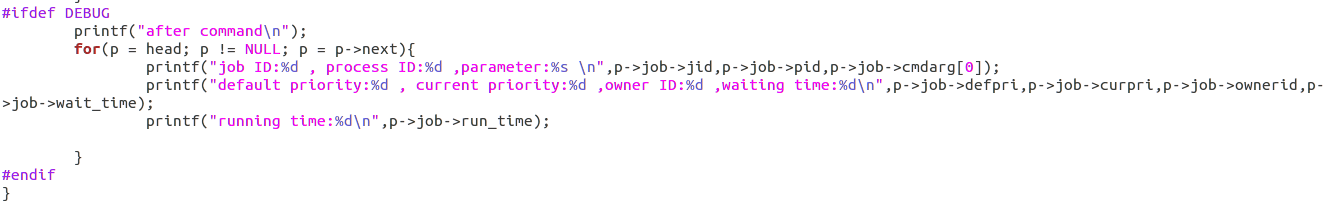


任务七

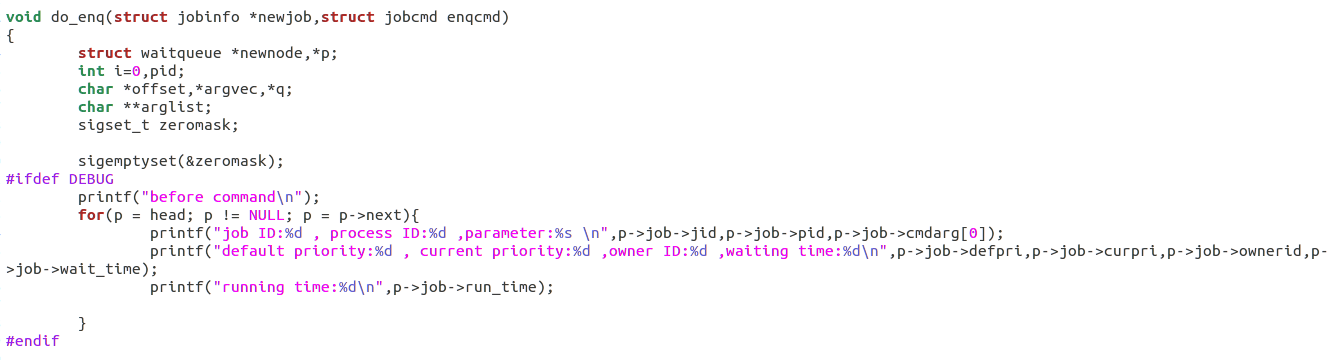
命令调试：在job.c里，实现“显示ENQ，DEQ，STAT命令执行前后的队列里所有进程的相关信息”；

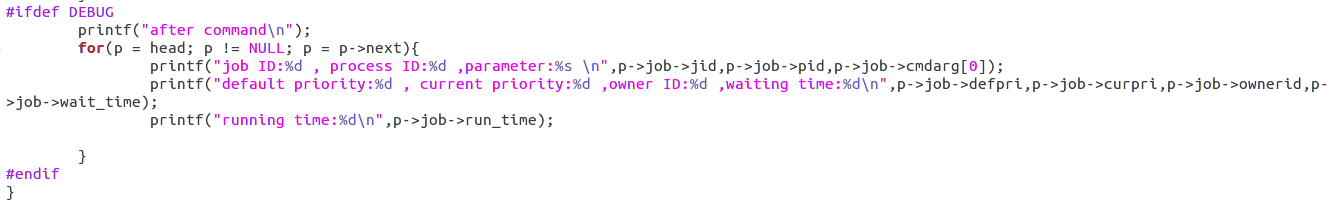
实现deq相关：



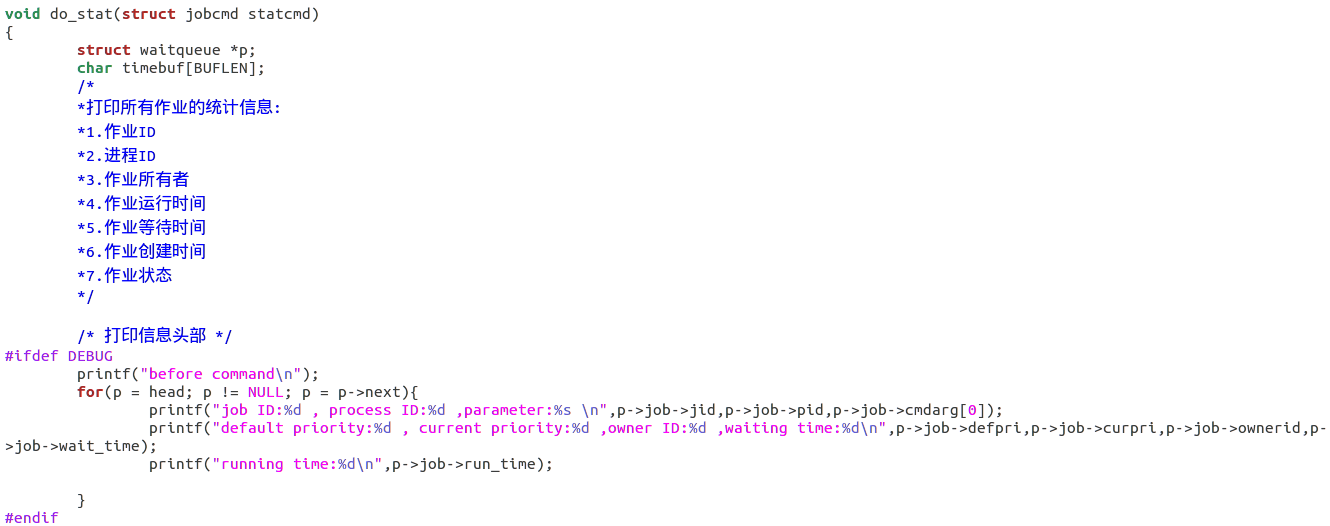


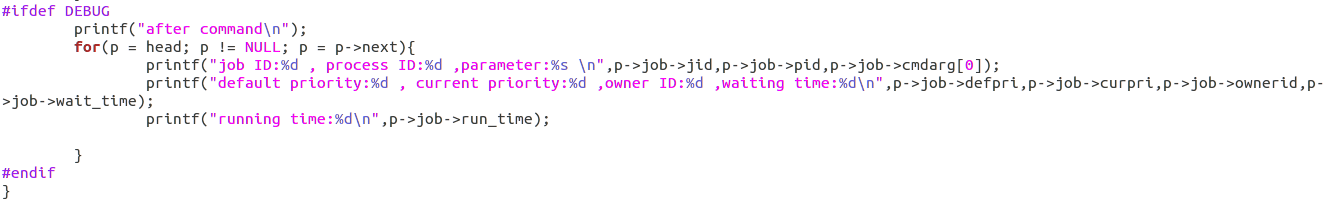
实现enq相关：





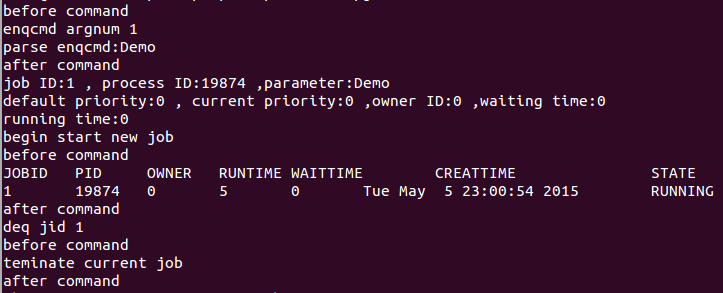
实现stat相关：



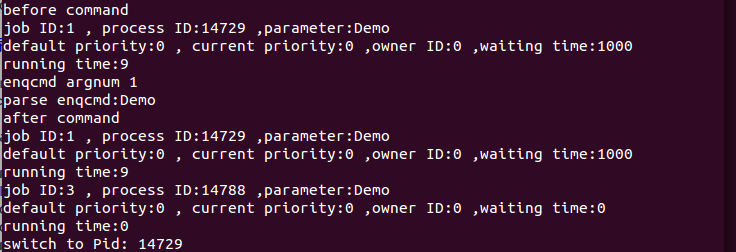


重新编译并执行，结果如下：

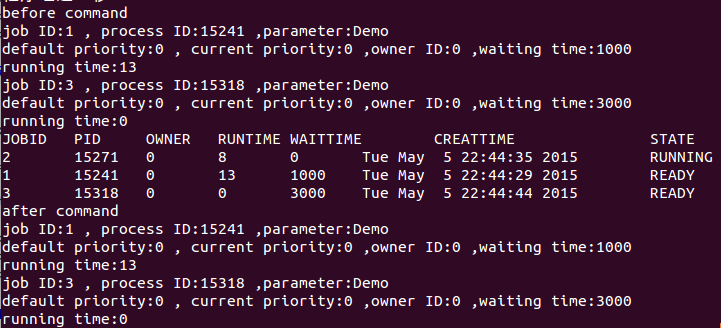
执行deq：



执行enq：

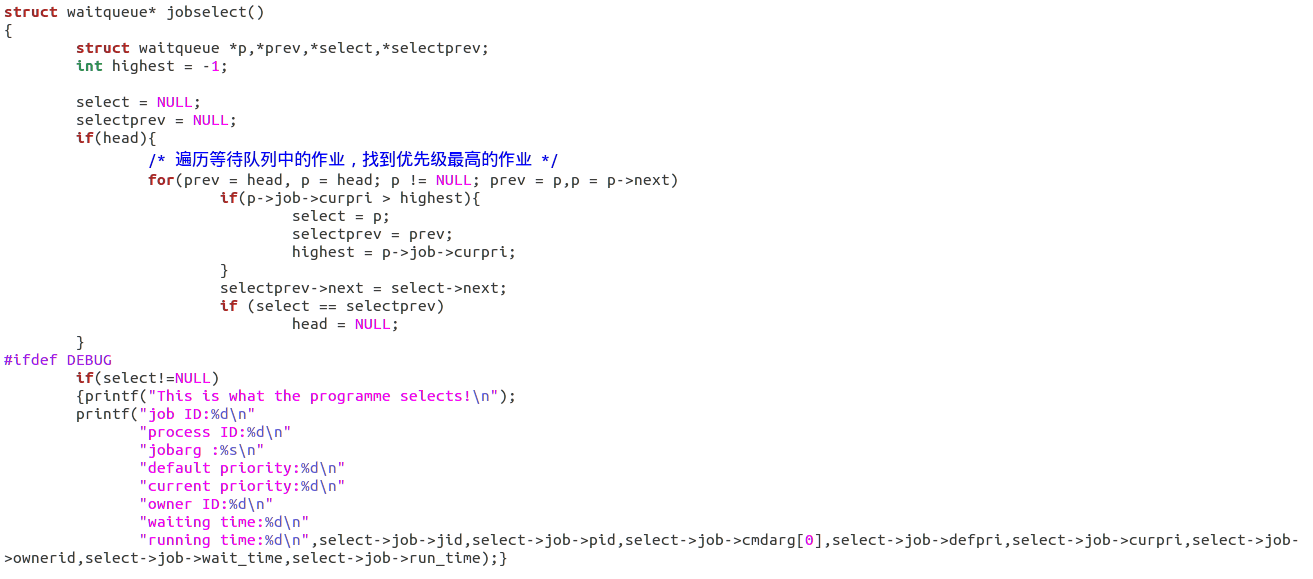


执行stat：

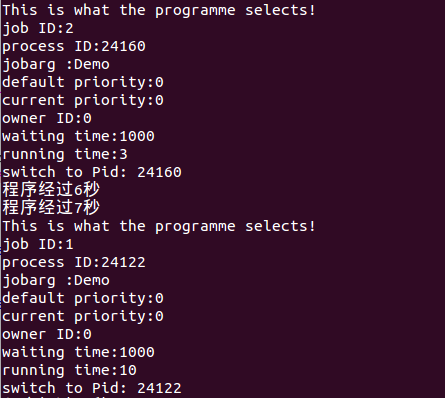


任务八

jobselect函数调试：在job.c里，实现“显示jobselect函数选择的进程的信息”；

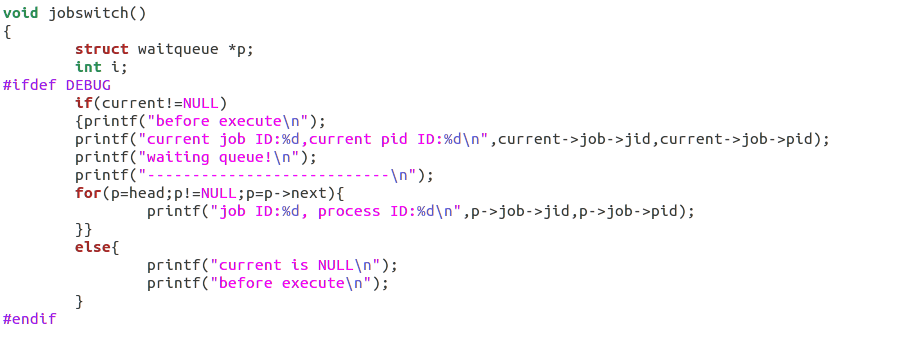


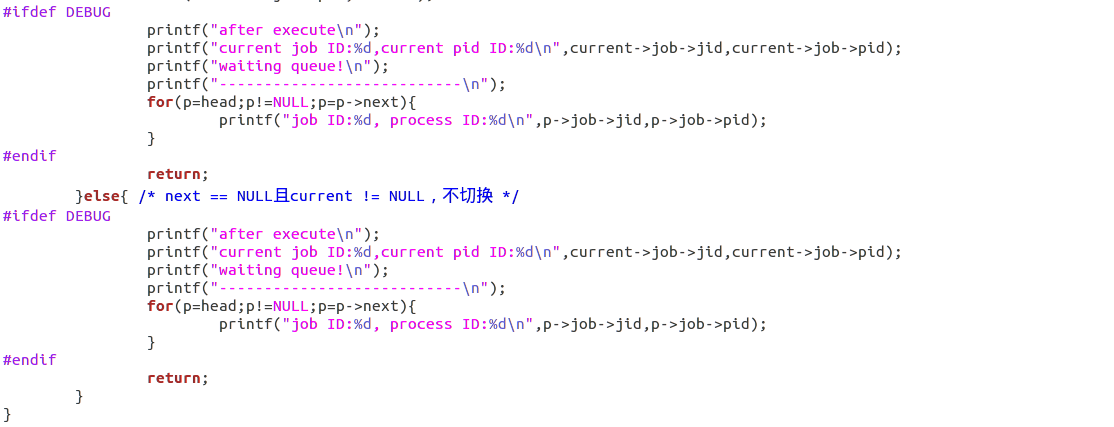
重新编译并执行，结果如下：



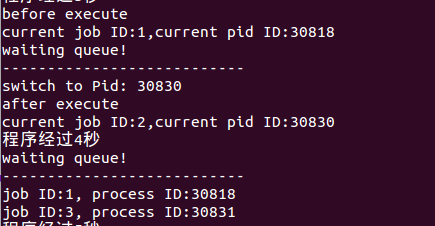
任务九

jobswitch函数调试：在job.c里，实现“显示jobswitch执行前后的正在执行的进程和等待队列中的分别的情况”；





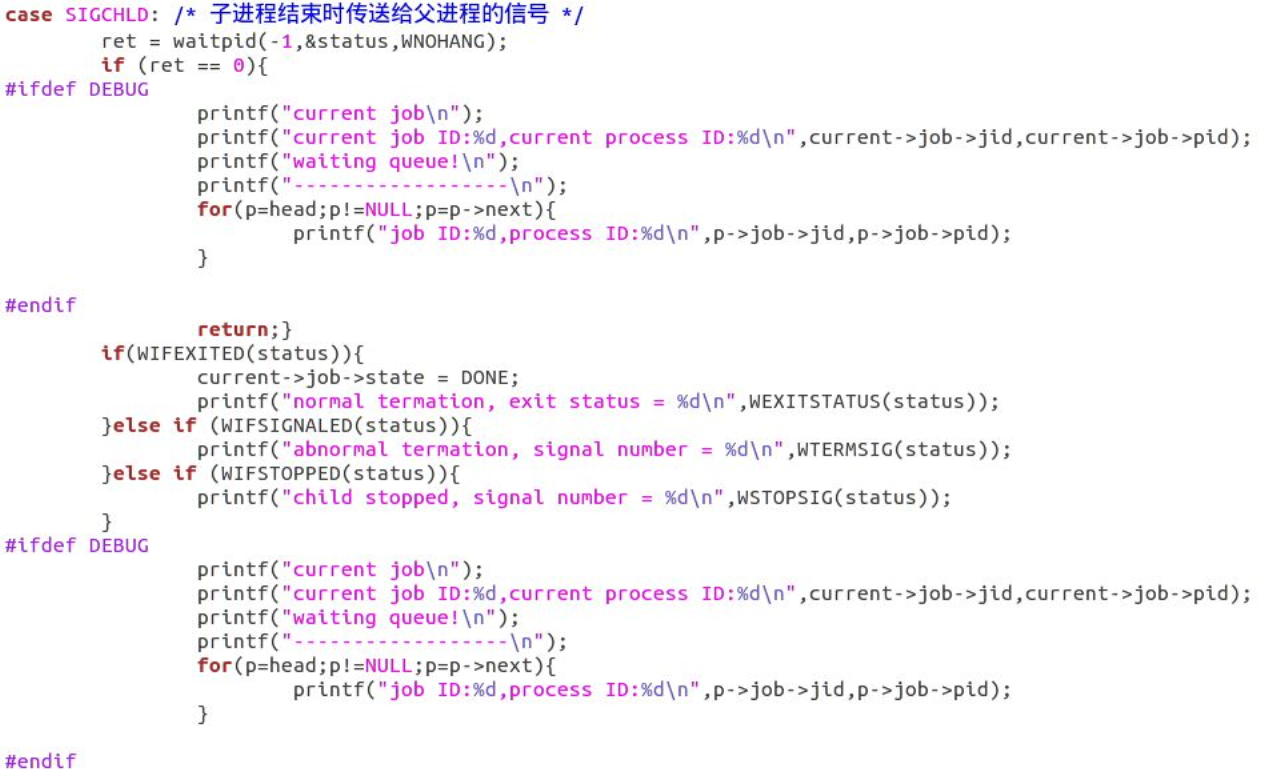
重新编译并执行，结果如下：



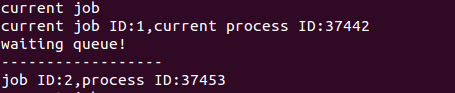
任务十

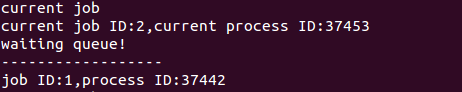
进程运行完成调试：在job.c里，实现“显示当进程自己运行完成后发送SIGCHLD给sig\_handler后的正在执行的进程和等待队列中的状态”；

考虑是否需要修改Demo.c：不需要



重新编译并执行，结果如下





**提高要求**

实现多级反馈轮转调度算法

* 多级——多个队列
* 反馈——优先级会根据等待时间变化
* 轮转——同优先级的作业轮流进行

**算法要求**

* 实现多级反馈轮转调度算法
* 要求实现3个队列，每个队列时间片不一样：最高优先级队列的轮转时间为1s，次高优先级为2s，最低优先级为5s；
* 当有新作业加入，抢占式运行，而不是等待当前时间片结束；
* 高优先级队列结束前，不轮转低优先级队列，等待时间超过10s自动升高优先级一级。

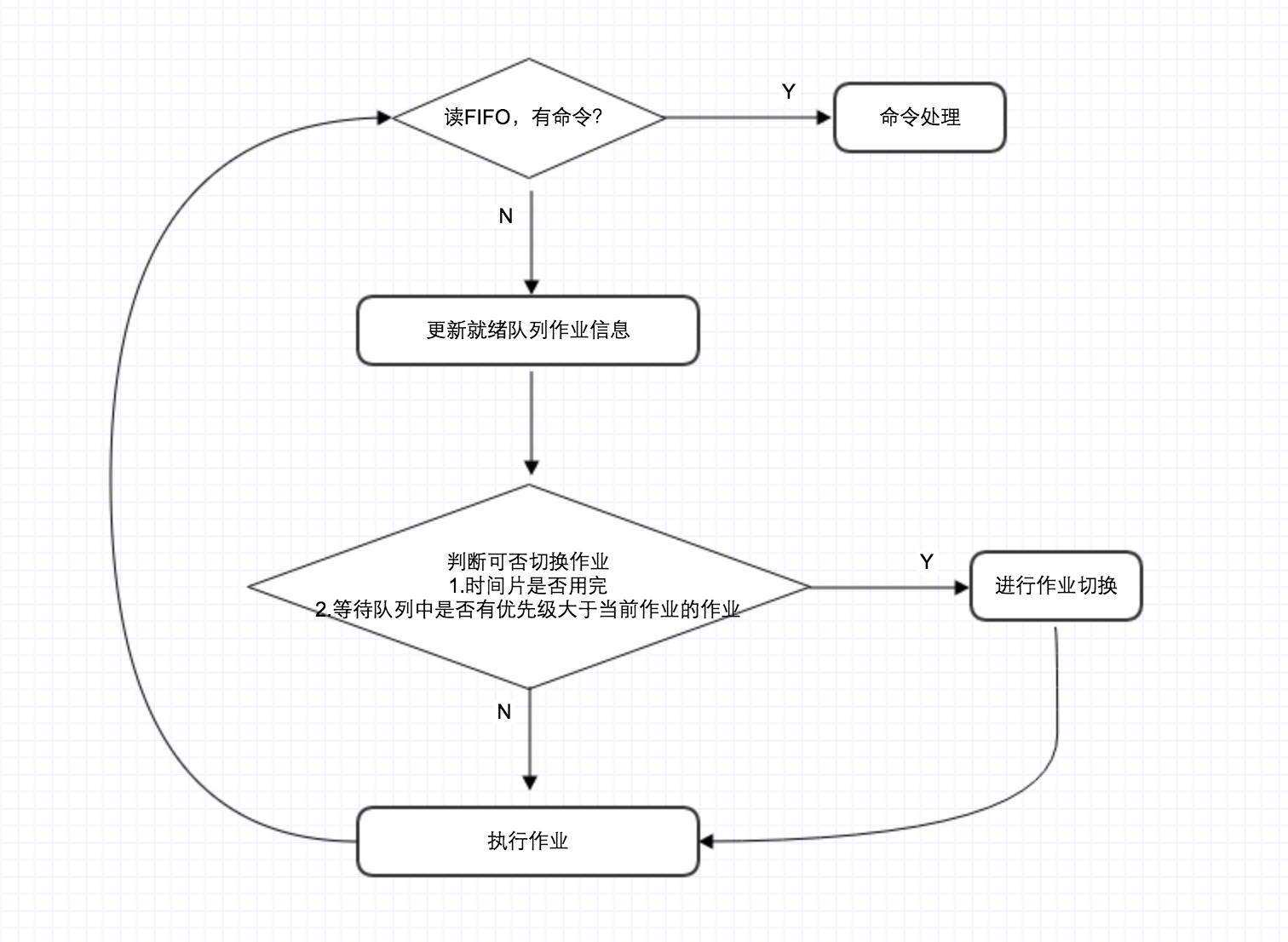
## 1.2完成情况

我们组本次完成了所有基本要求，调试任务以及提高要求。

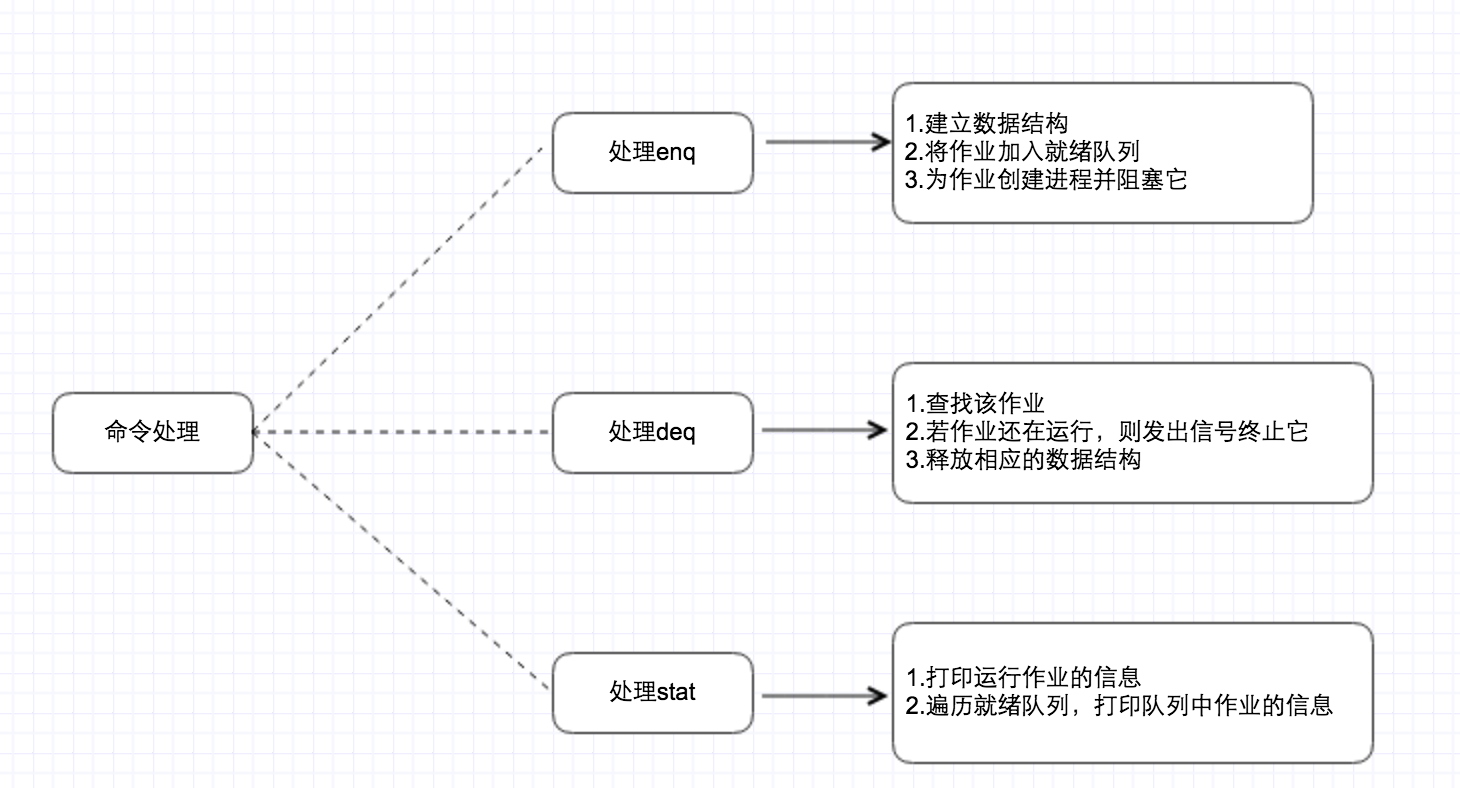
# 2设计说明

## 2.1流程示意图

程序流程图：



命令处理示意图：



## 2.2所使用的系统调用的列表

|  |  |  |
| --- | --- | --- |
| 名称 | 作用 | 返回值 |
| bzero（void \*s, int n） | 置字节字符串s的前n个字节为零且包括‘\0’。 |  |
| read(int fd,void \* buf ,size\_t count) | read()会把参数fd 所指的文件传送count个字节到buf指针所指的内存中。文件读写位置会随读取到的字节移动。 | 若参数count为0，则read为实际读取到的字节数，如果返回0，表示已到达文件尾或是无可读取的数据 |
| kill(pid\_t pid, int sig) | 用于向任何进程组或进程发送信号。 |  |
| waitpid(pid\_t pid,int \* status,int options) | waitpid()会暂时停止目前进程的执行，直到有信号来到或子进程结束。 | 如果执行成功则返回子进程识别码(PID) ,如果有错误发生则返回  返回值-1。 |
| fork() | 一个现有进程可以调用fork函数创建一个新进程。由fork创建的新进程被称为子进程（child process）。 | 若成功调用一次则返回两个值，子进程返回0，父进程返回子进程标记；否则，出错返回-1。 |
| getppid() | 用来取得目前进程的父进程识别码。 | 目前进程的父进程识别码。 |
| dup2(int oldhandle,int newhandle) | 复制文件句柄 |  |
| execv(const char \*pathname, char \* const argv[]) | 装入并运行其它程序的函数 | 如果应用程序正常执行完毕，那么execv是永远不会返回的；出错了，它的返回值应该是-1 |
| signal(int signum,sig\_t handler) | signal()会依参数signum 指定的信号编号来设置该信号的处理函数。当指定的信号到达时就会跳转到参数handler指定的函数执行。 |  |

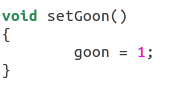
## 2.3提高要求实现说明

因为没有具体检测，所以在这里详细说明提高要求

首先对于do\_enq时创建子进程先后的问题，我们的做法是设置同步锁，代码如下：



其中setGoon为



**提高要求：**

1.首先设置三个链表头，分别为head1，head2，head3，

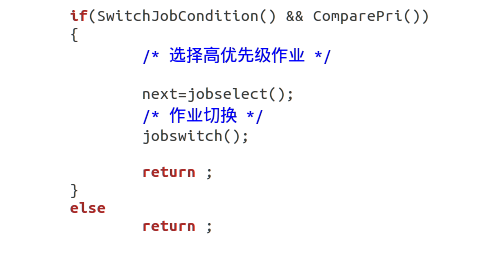
head1表示优先级为1的队列，

head2表示优先级为2的队列，

head3表示优先级为3的队列，

在数据结构jobinfo添加整型变量turn\_time表示每个作业的轮转时间，设置检查轮转时间的函数，改变作业切换条件，在每次调度时检查轮转时间，若当前作业的轮转时间大于当前优先级的时间片时间，并且在三个等待队列中存在大于等于当前作业优先级的作业，即可进行作业选择及切换。

将原来的直接选择作业并切换改为有条件的切换，代码如下：



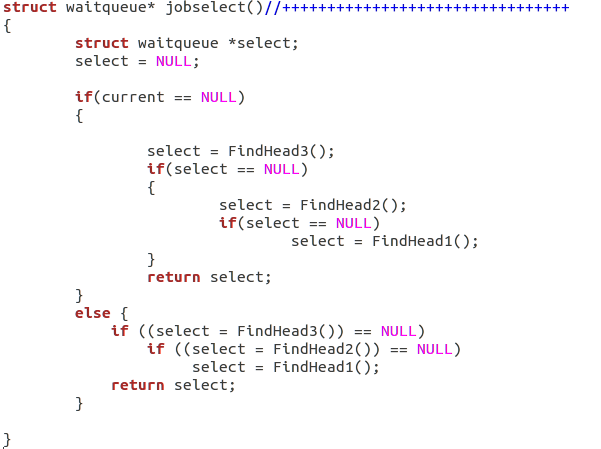
SwitchJobCondition()是对当前运行的时间片进行检查，运行是否超过时间片，返回值表示能否切换



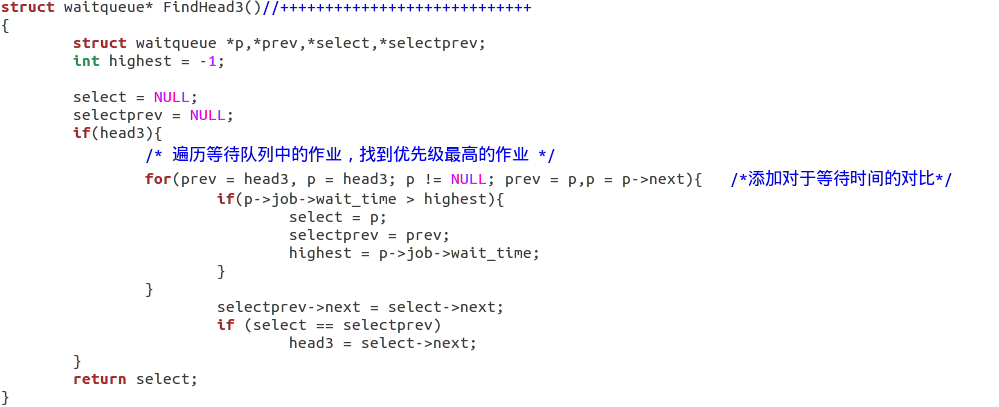
ComparePri()表示当前队列中是否存在比当前任务优先级还高的作业，返回值表示能否切换



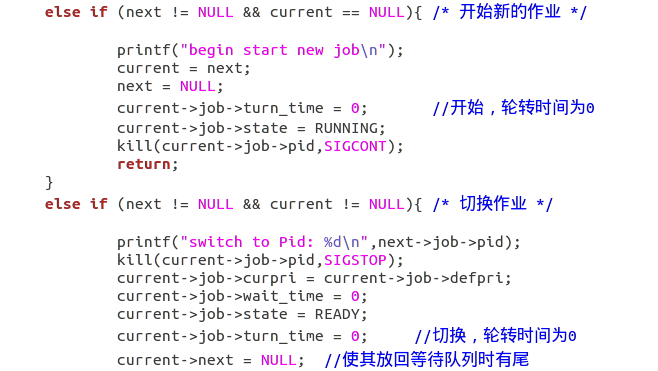
作业选择部分改为：



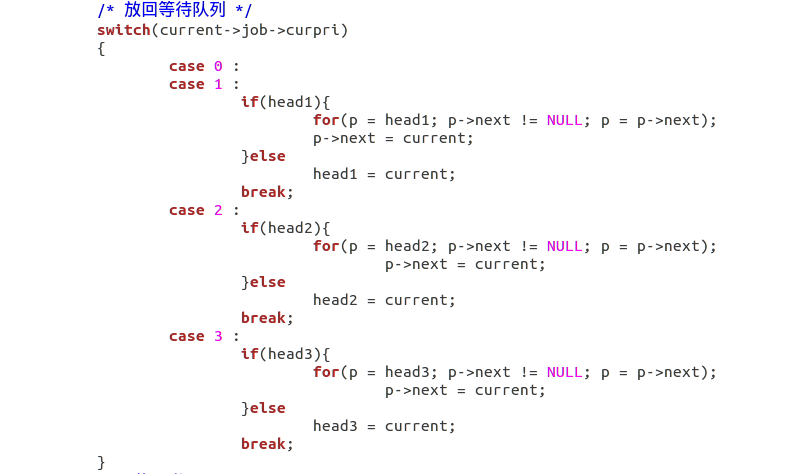
其中FindHead1(),FindHead2(),FindHead3()类似，在同一优先级对比等待时间，时间越长的，越先运行。同时对源代码进行修改，使得本程序可以同时切换优先级相同的3个以上的作业(即head3 = select->next)



关于作业切换部分，进行修改的由，每次无论是开始新的作业，还是切换作业，都添加了轮转时间初始化为0的操作

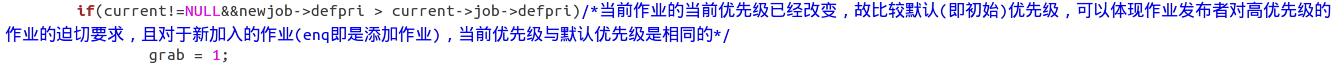


放回等待队列部分作出修改，使其能放回优先级适宜的队列

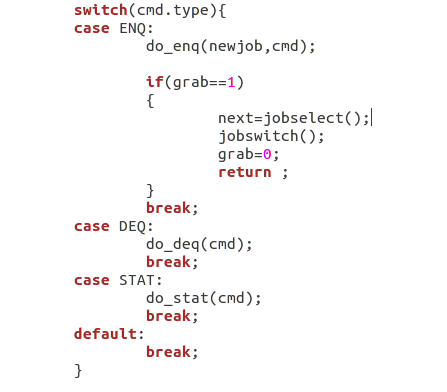


2.在每次执行enq指令，向等待队列中增加新的作业之前，判断当前作业是否为空，且新添加的作业的默认优先级是否大于当前作业的默认优先级，如果满足，设置标志grab。在do\_enq函数之后即判断是否设置grab，若满足，则进行作业选择及切换，以完成抢占。

抢占部分，我们理解为执行do\_enq()前后的一次作业选择与切换。

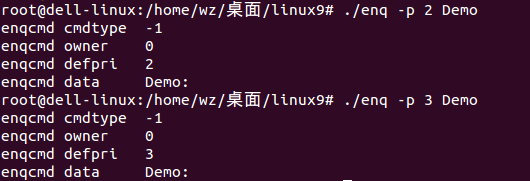
首先在do\_enq()中放入等待队列，前进行判断，入队作业的优先级是否大于等于当前作业的优先级(要求当前作业不为控)，grab为是否抢断的标志。

其次在执行完do\_enq()后，再次判断此时抢断是否成立，可否选择，切换作业

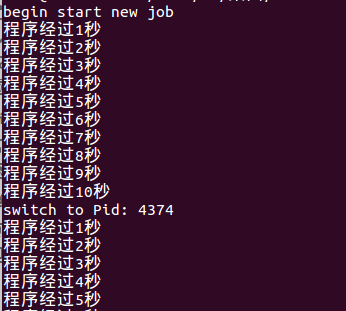


抢占效果

指令：

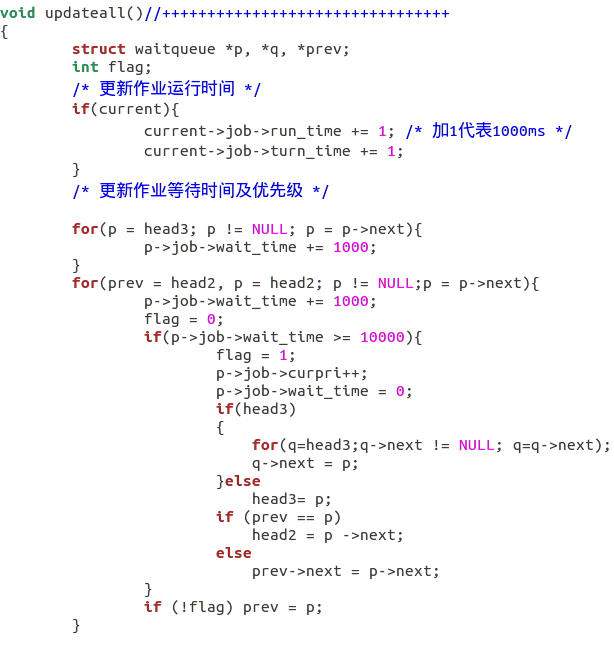


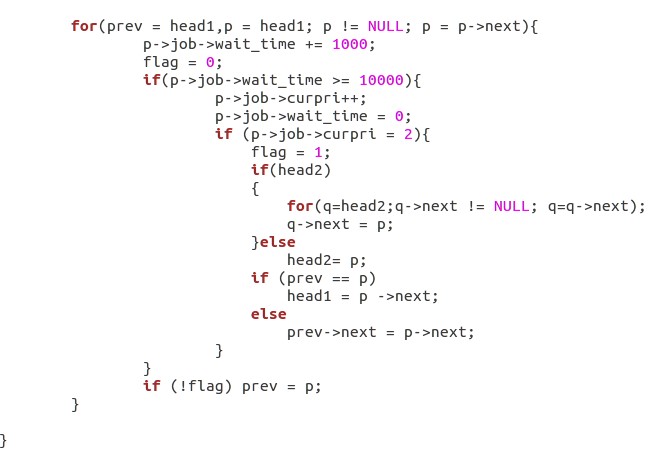
效果：



3.在更新队列时，使三个队列中每个作业的wait\_time累加，在head1和head2中，若等待时间超过10s，即使该作业的当前优先级+1，将其从当前队列删除并添加到相应的优先级队列，如果该作业得到机会运行，时间片用尽后，即使其变回原来的默认优先级并添加至相应队列。

updateall()更新队列修改，使得优先级提升后删除当前队列的作业，并添加到相应的优先级





# 3收获与感想

## 3.1给予你帮助的人

本组的郭奕鑫同学，对源码中的错误进行了一一修改，并完成了部分提高要求，受益良多。

本班的迟禄同学，给出Demo的限制。

## 3.2从实验中学到的东西

通过本次实验，我们初步了解了操作系统作业调度的机制，了解到操作系统的分时作业切换机制，以及每个作业内部所包含的数据结构。对作业的分进程运行机制，也有了充分的理解。

总的来说，我们认识到作业多级反馈轮转调度的整体机制，即在每一个时间单位，检查当前作业的时间片，并通过优先级队列、等待时间来选择下一个即将执行的作业。初步认识了作业切换所化费的开销，同时也认识到作业多级反馈轮转调度的优点，即对用户作业的及时响应，以及确保每个作业都能得到一定程度的运行。

同时，我体会到，本次作业对于我们加深对计算机操作系统的理解，以及锻炼按合理逻辑解决实际问题的能力，都有一定的帮助。

## 3.3任务分配

13061076赵乐：调试任务1,2,3,4，部分实验报告撰写

13061067沈明飞：调试任务5,7,9

13061090余旭：调试任务6,8,10

13061088郭奕鑫：提高任务：更新队列，以及源码所有插入，选择部分的更正，对作业选择以及切换的修改

13061074王喆：提高任务：3级队列、时间片轮转、作业抢占，对源码作业选择、切换、更新的初次修改，以及实验报告的部分撰写