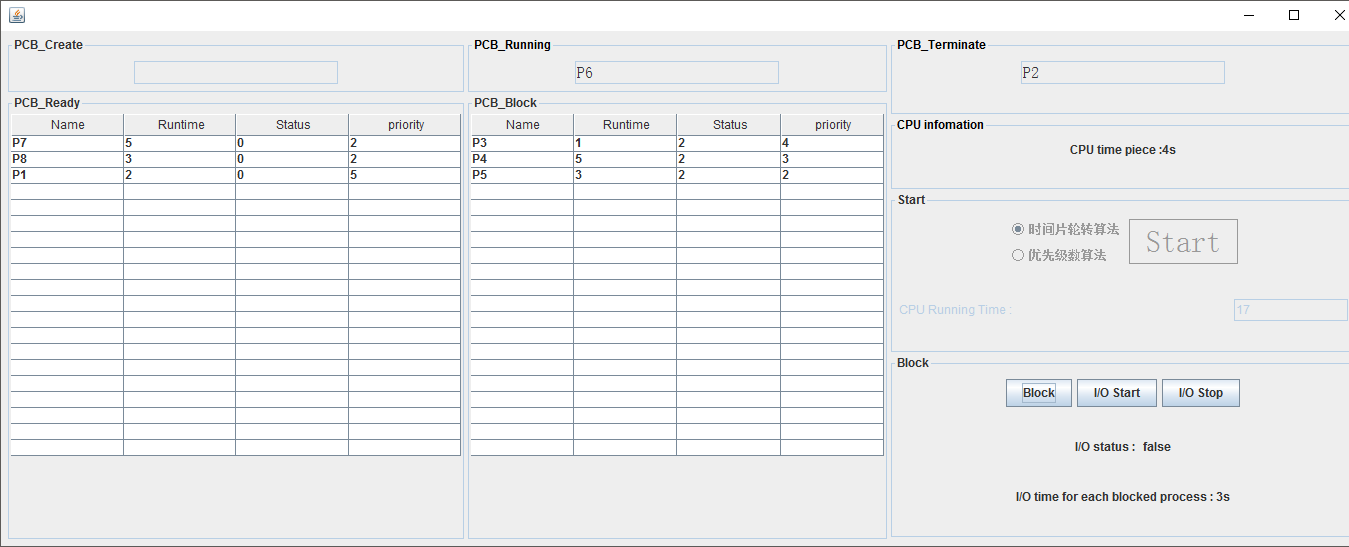
1. **多道程序系统中程序并发模拟实验**

我将第一次作业的第一题和第四题以及第三次作业的第一题三个部分写在了一起。

（该程序需要读写文件，请保证该项目放置的目录，普通程序有读写权限，建议放在C盘外的盘符）



程序截图

整个程序使用Java写的，初学Java，这是我的第一个Java图形化界面。虽然丑了点~~因为对很多控件的属性不是很清楚。

程序的原理并不难。

分为四个线程，同步运行：

* CheckIODevice：从进程创建开始到所有进程运行结束。负责一直检测I/0设备是否开启。以便于让阻塞队列中的进程以3s为时间周期回到就绪队列。
* CreatPCBQueue：创建pcbs数组的进程。从JOB.txt中导入进程信息到PCBs2的数组中，并显示在界面中。
* RereshData：刷新界面中的组件的数据
* ScanningReadyQueue：从就绪队列中根据算法选择合适的进程，使其运行

分为两个算法：

* 时间片轮转算法：本质是先来先服务 + 4s的运行周期
* 优先数算法：本质是4s的运行周期 + 按优先数排序运行进程

这两个算法非常简单。

优先数算法是没经过一个运行周期，当前进程的优先数会减1。

1. **LINUX下的编程**

用U盘装的双系统，Linux发行版使用的是deepin系统。

使用code::blocks 软件编写的。和Windows基本没有差别。

#include <stdio.h>

void main (){

int a ;

int b;

scanf("%d %d", &a, &b);

if(a>=b){

printf("%d" ,a);

}else {

printf("%d", b);

}

}

1. **创建子进程**

#include <stdio.h>

/\*两个进程中，原先就存在的那个被称作“父进程”，新出现的那个被称作“子进程”。父子进程的区别除了进程标志符（process ID）不同外，变量pid的值也不相同，pid存放的是fork的返回值。fork调用的一个奇妙之处就是它仅仅被调用一次，却能够返回两次，它可能有三种不同的返回值：

在父进程中，fork返回新创建子进程的进程ID；

在子进程中，fork返回0；

如果出现错误，fork返回一个负值；

fork出错可能有两种原因：

（1）当前的进程数已经达到了系统规定的上限，这时errno的值被设置为EAGAIN。

（2）系统内存不足，这时errno的值被设置为ENOMEM。

\*/

int main (){

int pid;

printf("PID before fork() : %d\n", getpid());

pid = fork();

if(pid < 0){

printf("error in fork()!");

}else if(pid == 0){

printf("I am the child, my pid is %d\n", getpid());

}else{

printf("I am the parent, my pid is %d\n",getpid());

}

printf("hello linux!");

}