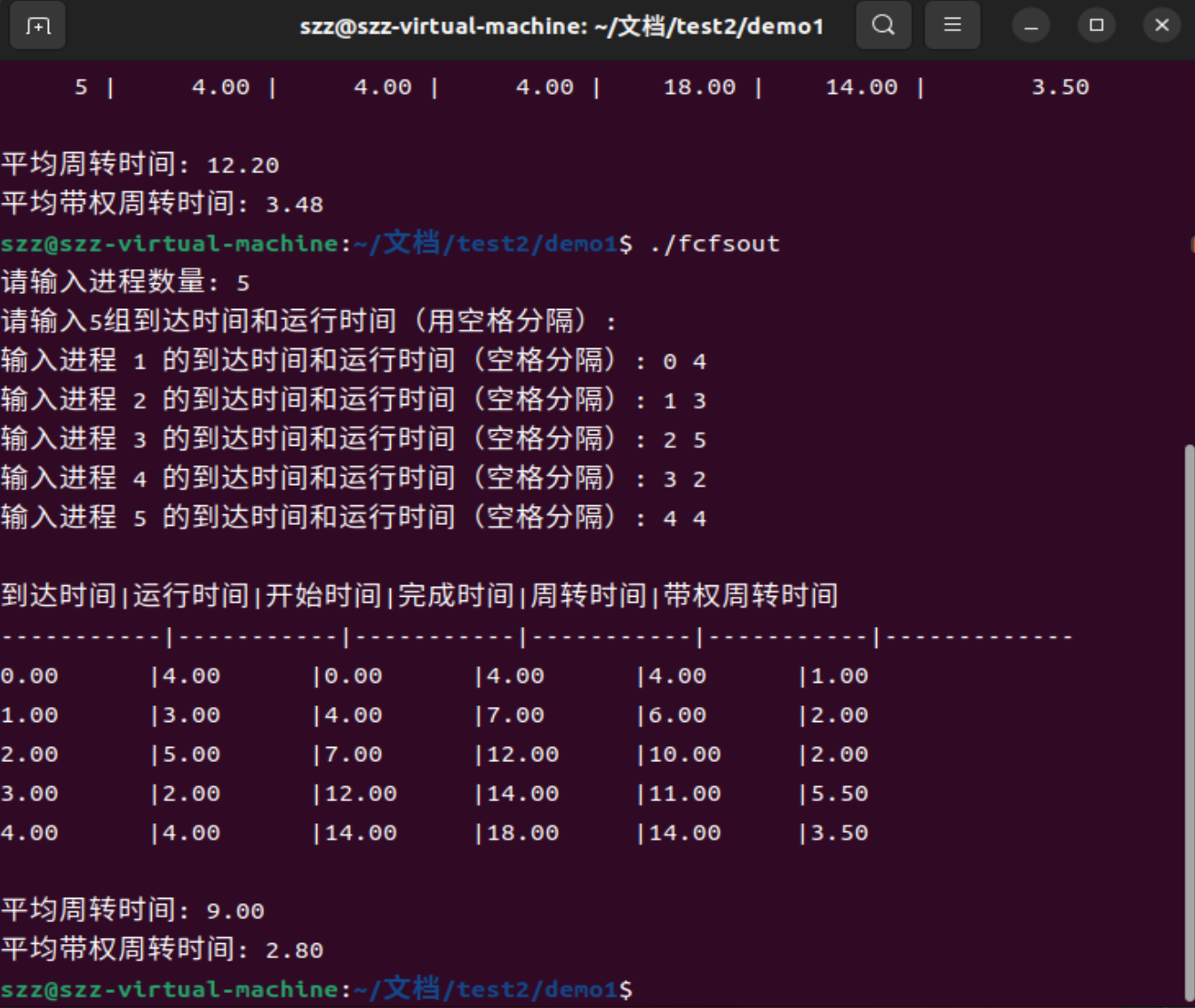
FCFS先来先服务



根据到达时间和上一进程的完成时间决定下一进程的开始时间

进程1到达时间为0，开始时间为0，运行时间为4，完成时间为4

进程2到达时间为1，开始时间为4，运行时间为3，完成时间为7

虽然进程2到达时间为1，但此时先来的进程1在运行，进程1完成后才可进程2

进程3到达时间为2，开始时间为7，运行时间为5，完成时间为12

进程4到达时间为3，开始时间为12，运行时间为2，完成时间为14

进程5到达时间 为4，开始时间我14，运行时间为4，完成时间为18

总结：上一进程的完成时间是下一进程的开始时间。FCFS调度策略下，进程按到达顺序执行，虽然实现简单，但可能导致较晚到达的短进程面临较长等待时间，影响整体效率

SJF短作业优先图片包含 图形用户界面

AI 生成的内容可能不正确。

进程的执行顺序由其运行时间决定，优先执行当前可运行进程中运行时间最短的进程。

进程1：到达时间为0，开始时间为0，运行时间为4，完成时间为4。它是最先到达的进程，因此立即开始执行。

进程4：到达时间为3，但由于它的运行时间（2）是所有可运行进程中最短的，因此在进程1完成后（时间点4）立即开始执行，完成时间为6。

进程2：到达时间为1，运行时间为3。在时间点6时，它仍然是当前可运行进程中运行时间最短的，因此开始执行，完成时间为9。

进程5：到达时间为4，在时间点9时已到达且运行时间为4，因此开始执行，完成时间为13。

进程3：到达时间为2，运行时间为5。最后只剩下它和已完成的进程，因此开始执行，完成时间为18。

总结：在SJF调度策略下，进程按照运行时间从小到大的顺序执行，而不是按到达时间顺序。这种策略减少了平均等待时间和周转时间，提高了系统效率。尽管实现稍复杂，但相较于FCFS，SJF能有效避免较晚到达的短进程长时间等待的问题，从而优化整体性能。

MFQ多级反馈队列调度



时间片设置的为第0级为1，第1级为2，… i+1

按照先来先服务进入队列，队列设置最大为100，因为根本用不完。

所有进程进入队列1等待。

进程1先运行第0级时间片，没运行完进入队列2。

进程2运行第0级时间片，没运行完进入队列2。

（只要队列1不空，队列2内进程就不能开启时间片）

…队列1空

进程1运行第1级时间片，没运行完进入队列3

依次类推，直到所有进程运行完。

时间点 0：

进程1到达，进入队列0。

开始执行进程1，分配时间片1（队列0）。执行从0到1。

时间点 1：

进程2到达，进入队列0。

进程1继续执行，因为它的时间片还未用完。执行从1到2。

时间点 2：

进程3到达，进入队列0。

进程1继续执行，执行从2到3。

时间点 3：

进程4到达，进入队列0。

进程1继续执行，执行从3到4。进程1完成，总完成时间为4。

时间点 4：

接下来选择队列0中的进程执行。由于进程4的剩余时间最短（2），它开始执行。执行从4到6。进程4完成，总完成时间为6。

时间点 6：

队列0现在有进程2和进程5等待。

进程2开始执行，分配时间片2（队列0）。执行从6到8。

进程2还需要1个单位时间才能完成，因此被移到队列1。

时间点 8：

进程5开始执行，分配时间片1（队列0）。执行从8到9。

时间点 9：

进程5继续执行，分配时间片1（队列0）。执行从9到10。

进程5仍需3个单位时间才能完成，因此被移到队列1。

时间点 10：

队列1中有进程2等待。

进程2继续执行，分配时间片2（队列1）。执行从10到12。进程2完成，总完成时间为12。

时间点 12：

队列1中有进程5等待。

进程5继续执行，分配时间片2（队列1）。执行从12到14。

时间点 14：

进程5继续执行，分配时间片2（队列1）。执行从14到16。

进程5仍需2个单位时间才能完成，因此被移到队列2。

时间点 16：

队列2中有进程3和进程5等待。

进程3开始执行，分配时间片3（队列2）。执行从16到21。进程3完成，总完成时间为21。

计算周转时间和带权周转时间

周转时间 = 完成时间 - 到达时间

平均周转时间 = (4 + 11 + 19 + 3 + 14) / 5 = 10.2

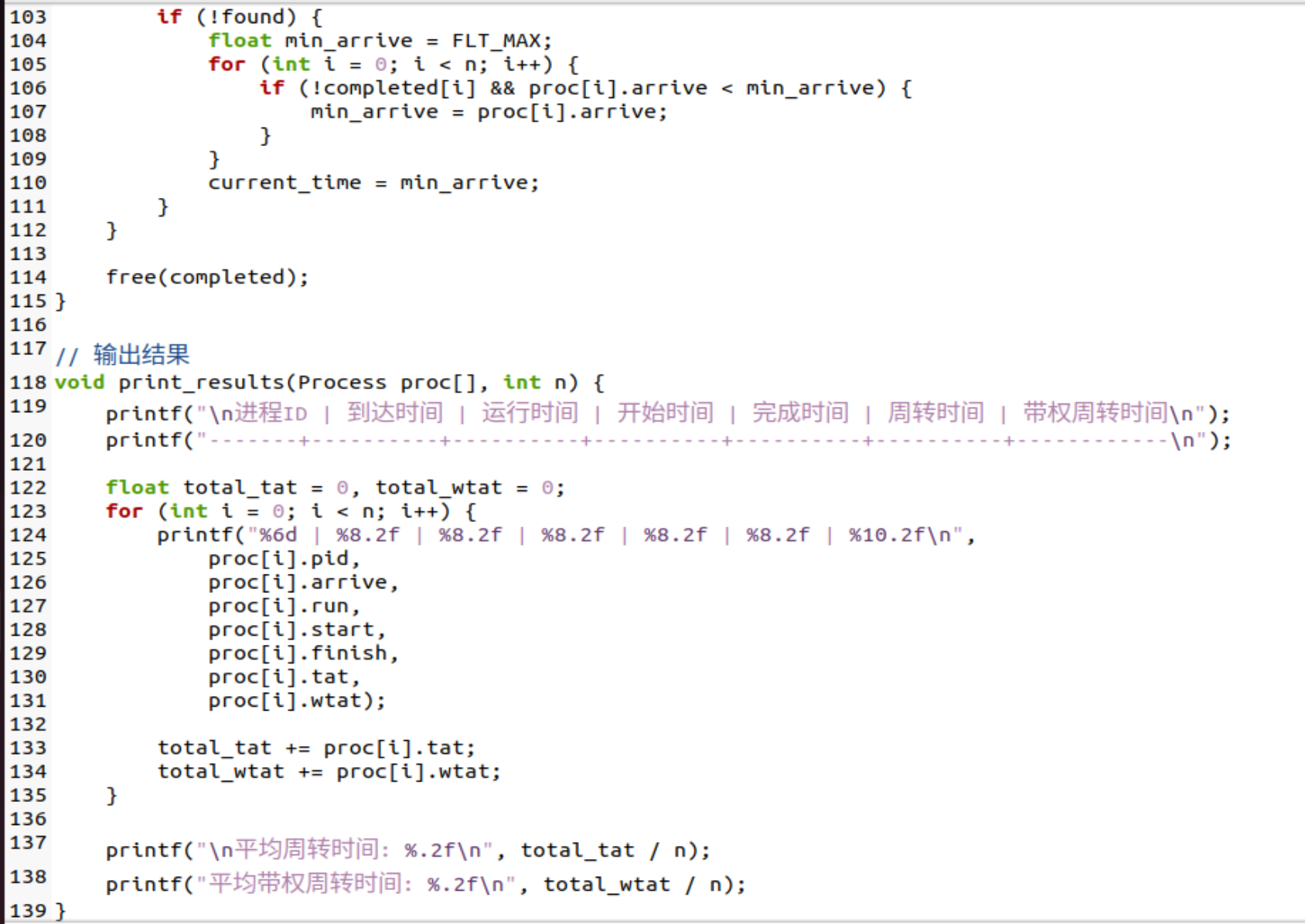
带权周转时间 = 周转时间 / 运行时间

平均带权周转时间 = (1 + 3.67 + 3.8 + 1.5 + 3.5) / 5 ≈ 2.69









文本, 信件

AI 生成的内容可能不正确。