实验二、进程调度实验

1. 实验目的

用高级语言编写和调试一个进程调度程序，以加深对进程的概念及进程调度算法的理解

1. 实验内容

任务：设计一个有 N 个进程并行的进程调度程序进程调度算法：采用最高优先数优先的调度算法（即把处理机分配给优先数最高的进程）和同优先级条件下先来先服务算法。每个进程有一个进程控制块（ PCB）表示。进程控制块可以包含如下信息：进程名、优先数、需要运行时间、已用 CPU 时间、进程状态等等。进程的优先数及需要的运行时间可以事先人为地指定（也可以由随机数产生）。进程的运行时间以时间片为单位进行计算。每个进程的状态可以是就绪 W（Wait）、运行 R（Run）、或完成 F（Finish）三种状态之一。 就绪进程获得 CPU 后都只能运行一个时间片。用已占用 CPU时间加 1 来表示。如果运行一个时间片后，进程的已占用 CPU 时间已达到所需要的运行时间，则撤消该进程，如果运行一个时间片后进程的已占用CPU 时间还未达所需要的运行时间，也就是进程还需要继续运行，此时应将进程的优先数减 1（即降低一级），然后把它插入就绪队列等待 CPU。每进行一次调度程序都打印一次运行进程、就绪队列、以及各个进程的 PCB，以便进行检查。重复以上过程，直到所有进程都完成为止。

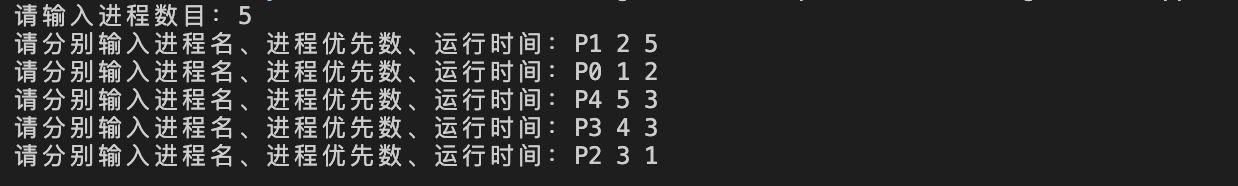
**题目 ：有5个进程：P0 P1 P2 P3 P4；进程进入顺序、优先级、运行时间如下：**

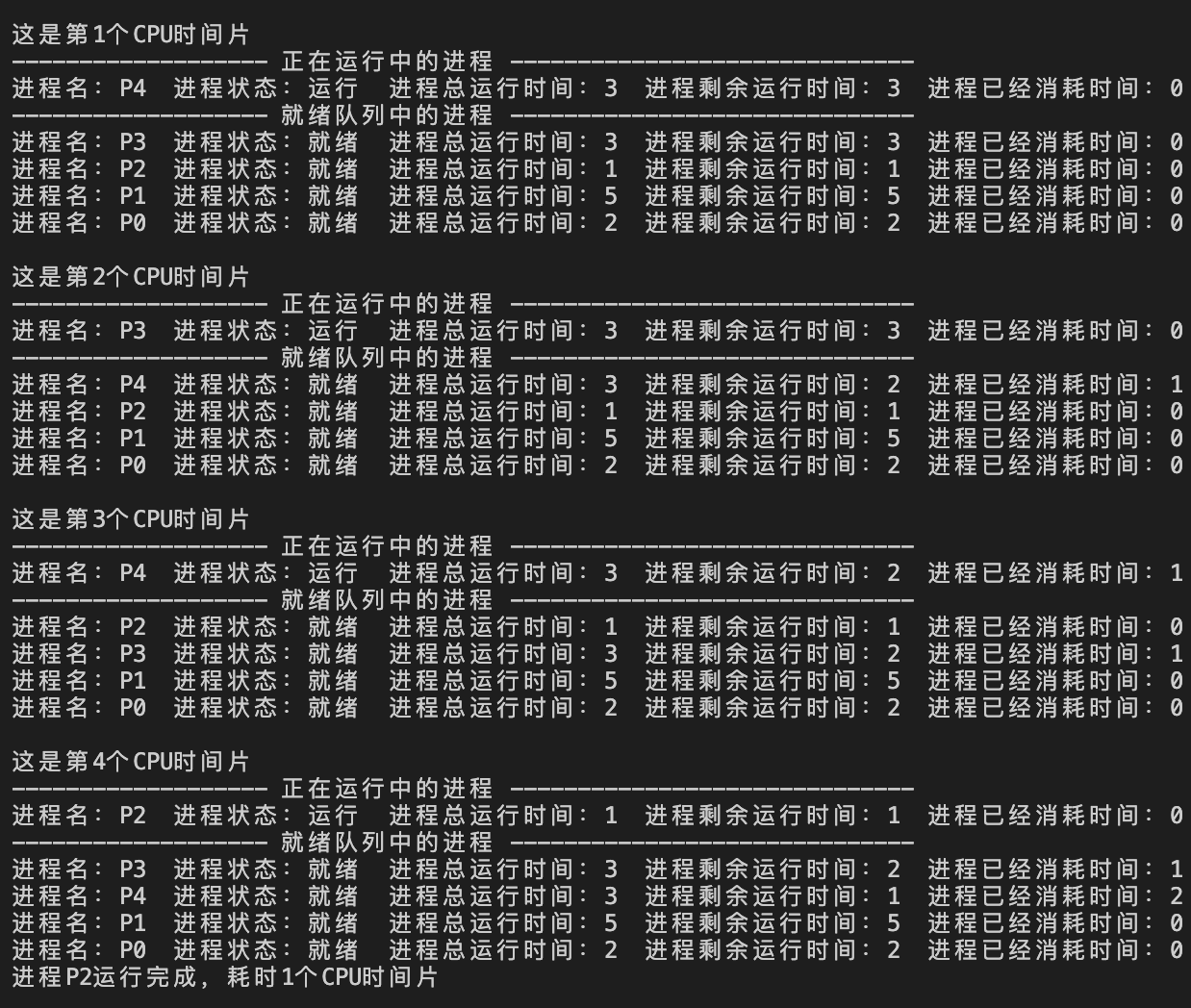
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **进程** | **P1** | **P0** | **P4** | **P3** | **P2** |
| **优先级** | 2 | 1 | 5 | 4 | 3 |
| **运行时间** | 5 | 2 | 3 | 3 | 1 |

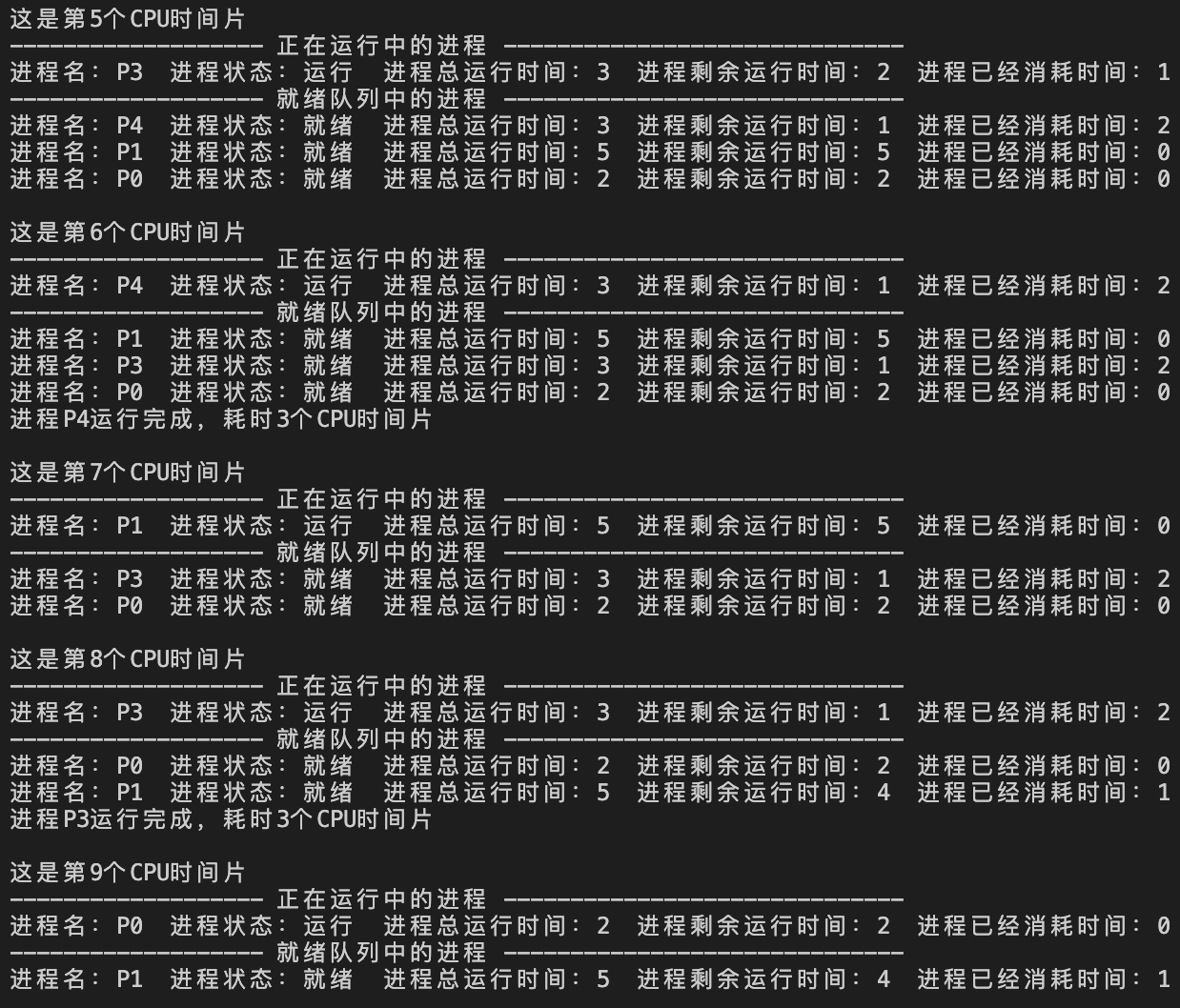
**实验过程：**

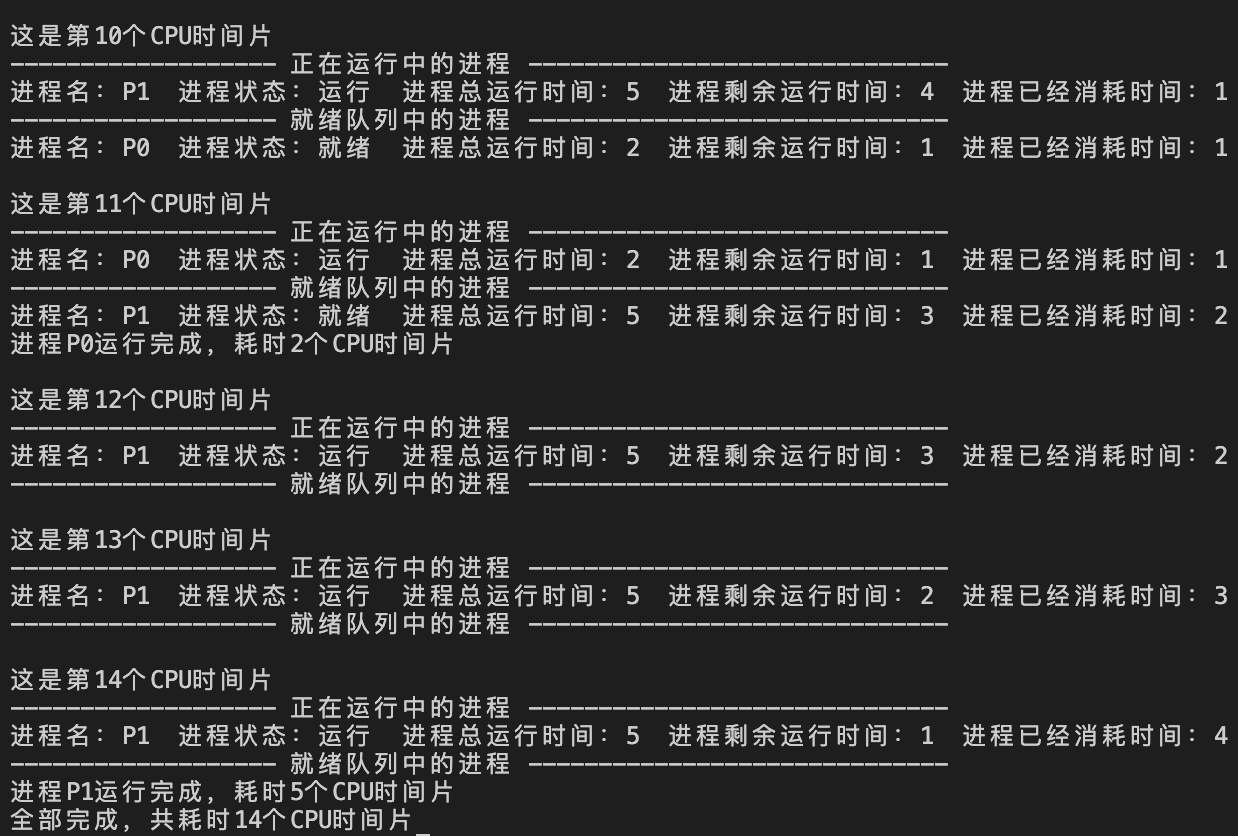
1. #include <iostream>
2. #include <stdlib.h>
3. #include <time.h>
4. **using** **namespace** std;
6. #define getpch(type) (type \*)malloc(sizeof(type))
8. **int** t = 0;  // 记录时间片
9. **struct** pcb {  // 定义进程控制块PCB
10. **char** name[10];  // 进程名
11. **char** state;     // W/R/F (等待/运行/完成)
12. **int** super;      // 优先数
13. **int** ntime;      // 总运行时间
14. **int** rtime;      // 已耗时
15. **struct** pcb\* link;  // 指向下一个进程
16. } \*ready = NULL, \*p;
17. **typedef** **struct** pcb PCB;
19. // 排序函数，按照优先数从高到低排序，同优先级先来先服务
20. **void** sort() {
21. p->link = NULL;
22. **if** (ready == NULL)
23. ready = p;
24. **else** {
25. **if** (p->super > ready->super) {  // 如果当前进程的优先数比队列中的第一个进程高
26. p->link = ready;
27. ready = p;
28. }
29. **else** {
30. PCB\* f = ready;
31. **while** (1) {
32. **if** (f->link == NULL) {  // 如果是最后一个进程
33. f->link = p;
34. **return**;
35. }
36. **else** **if** (p->super > f->link->super) {  // 如果当前进程的优先数比下一个进程的优先数高
37. PCB\* s = f->link;
38. f->link = p;
39. p->link = s;
40. **return**;
41. }
42. f = f->link;
43. }
44. }
45. }
46. }
48. // 输入进程信息
49. **void** input() {
50. srand((unsigned)time(NULL));
51. **int** n;
52. cout << "请输入进程数目：";
53. cin >> n;
54. **for** (**int** i = 0; i < n; i++) {
55. p = getpch(PCB);
56. cout << "请分别输入进程名、进程优先数、运行时间：";
57. cin >> p->name >> p->super >> p->ntime;
59. p->state = 'W';
60. p->rtime = 0;
61. sort();
62. }
63. }
65. // 显示进程信息
66. **void** disp(PCB\* pr) {
67. cout << "进程名：" << pr->name << "  ";
68. cout << "进程状态：" << (pr->state == 'W' ? "就绪" : (pr->state == 'R' ? "运行" : "完成")) << "  ";
69. cout << "进程总运行时间：" << pr->ntime << "  ";
70. cout << "进程剩余运行时间：" << (pr->ntime - pr->rtime) << "  ";
71. cout << "进程已经消耗时间：" << pr->rtime << endl;
72. }
74. // 查看当前进程和就绪队列
75. **void** check() {
76. cout << "-------------------" << " 正在运行中的进程 " << "------------------------------" << endl;
77. disp(p);
78. cout << "-------------------" << " 就绪队列中的进程 " << "------------------------------" << endl;
79. PCB\* pr = ready;
80. **while** (pr != NULL) {
81. disp(pr);
82. pr = pr->link;
83. }
84. }
86. // 销毁已完成的进程
87. **void** destroy() {
88. cout << "进程" << p->name << "运行完成，耗时" << p->ntime << "个CPU时间片" << endl;
89. free(p);
90. p = NULL;
91. }
93. // 进程就绪函数
94. **void** running() {
95. p = ready;
96. ready = ready->link;
97. t++;
98. cout << endl << "这是第" << t << "个CPU时间片" << endl;
99. p->state = 'R';
100. check();
101. p->rtime++;
102. **if** (p->ntime == p->rtime) {
103. destroy();
104. }
105. **else** {
106. p->super -= 1;  // 优先数减 1
107. p->state = 'W';
108. sort();  // 重新排序
109. }
110. **if** (p == NULL && ready == NULL)
111. cout << "全部完成，共耗时" << t << "个CPU时间片" << endl;
112. }
114. // 主函数
115. **int** main() {
116. input();  // 输入进程信息
117. **while** (p != NULL || ready != NULL)  // 循环直到所有进程都完成
118. running();
119. **return** 0;
120. }

**运行结果：**

****

****

****

****