Submitted Files

Results | Code

cpu.c Download

```c
#include <stdio.h>;
#include <stdlib.h>
#include "oslabs.h"

int is_NULLpcb(struct PCB inpcb)
{
    if (inpcb.process_id == 0 && inpcb.arrival_timestamp == 0 && inpcb.total_bursttime == 0 && inpcb.execution_starttime == 0 && inpcb.execution_endtime == 0 && inpcb.remaining_bursttime ==0 && inpcb.process_priority == 0)
        return 1;
    else
        return 0;

}


struct PCB handle_process_arrival_pp(struct PCB ready_queue[QUEUEMAX], int *queue_cnt, struct PCB current_process, struct PCB new_process, int timestamp)
{
    if(is_NULLpcb(current_process))
    {
        new_process.execution_starttime = timestamp;
        new_process.execution_endtime = timestamp + new_process.total_bursttime;
        new_process.remaining_bursttime = new_process.total_bursttime;
        return new_process;
    }
    else if(new_process.process_priority >= current_process.process_priority)
    {
        new_process.execution_starttime=0;
        new_process.execution_endtime=0;
        new_process.remaining_bursttime=new_process.total_bursttime;
        ready_queue[*queue_cnt]=new_process;
        (*queue_cnt)++;
        return current_process;
    }
    else
    {
        new_process.execution_starttime=timestamp;
        new_process.execution_endtime=timestamp + new_process.total_bursttime;
        new_process.remaining_bursttime=new_process.total_bursttime;
        current_process.execution_endtime=0;
```

```
39
current_process.remaining_bursttime=current_process.remaining_bursttime -
1;
40          ready_queue[*queue_cnt]=current_process;
41          (*queue_cnt)++;
42          return new_process;
43      }
44  }
45
46  struct PCB handle_process_completion_pp(struct PCB ready_queue[QUEUEMAX],
int *queue_cnt, int timestamp)
47  {
48      if (*queue_cnt > 0)
49      {
50          struct PCB next_process;
51          int priorty = ready_queue[0].process_priority;
52          int temp_queue = 0;
53          for (int i = 1 ; i <= *queue_cnt - 1; i++)
54          {
55          if (priorty > ready_queue[i].process_priority)
56          {
57              priorty = ready_queue[i].process_priority;
58              temp_queue = i;
59          }
60          }
61          next_process = ready_queue[temp_queue];
62          if (*queue_cnt == 1)
63          {
64              ready_queue[0].process_id = 0;
65              ready_queue[0].arrival_timestamp = 0;
66              ready_queue[0].total_bursttime = 0;
67              ready_queue[0].execution_starttime = 0;
68              ready_queue[0].execution_endtime = 0;
69              ready_queue[0].remaining_bursttime = 0;
70              ready_queue[0].process_priority = 0;
71          }
72          else
73              ready_queue[temp_queue] = ready_queue[*queue_cnt - 1];
74              *queue_cnt = *queue_cnt - 1;
75              next_process.execution_starttime = timestamp;
76              next_process.execution_endtime = timestamp +
next_process.total_bursttime;
77              return next_process;
78      }
79      else
80      {
81          struct PCB null_PCB;
82          null_PCB.process_id = 0;
83          null_PCB.arrival_timestamp = 0;
```

```
84          null_PCB.total_bursttime = 0;
85          null_PCB.execution_starttime = 0;
86          null_PCB.execution_endtime = 0;
87          null_PCB.remaining_bursttime = 0;
88          null_PCB.process_priority = 0;
89          return null_PCB;
90
91      }
92  }
93  struct PCB handle_process_arrival_srtp(struct PCB ready_queue[QUEUEMAX],
    int *queue_cnt, struct PCB current_process, struct PCB new_process, int
    time_stamp)
94  {
95      if(is_NULLpcb(current_process))
96      {// NULLPCB , New process priority
97          new_process.execution_starttime = time_stamp;
98          new_process.execution_endtime = time_stamp +
    new_process.total_bursttime;
99          new_process.remaining_bursttime = new_process.total_bursttime;
100         return new_process;
101     }
102     else if ( new_process.total_bursttime >
    current_process.remaining_bursttime) { //new process goes to the queue
103         new_process.execution_starttime = 0;
104         new_process.execution_endtime = 0;
105         new_process.remaining_bursttime = new_process.total_bursttime;
106         ready_queue[*queue_cnt] = new_process;
107         *queue_cnt = *queue_cnt + 1;
108         return current_process;
109     }
110     else
111     { //current process goes to the queue
112         new_process.execution_starttime = time_stamp;
113         new_process.execution_endtime = time_stamp +
    new_process.total_bursttime;
114         new_process.remaining_bursttime = new_process.total_bursttime;
115         current_process.remaining_bursttime =
    current_process.execution_endtime - time_stamp;
116         current_process.execution_endtime = 0;
117         current_process.execution_starttime = 0;
118         ready_queue[*queue_cnt] = current_process;
119         *queue_cnt = *queue_cnt + 1;
120         return new_process;
121     }
122 }
123
124
125
```

```
126 struct PCB handle_process_completion_srtp(struct PCB
    ready_queue[QUEUEMAX], int *queue_cnt, int timestamp) {
127
128 }
129
130
131
132
133 struct PCB handle_process_arrival_rr(struct PCB ready_queue[QUEUEMAX],
    int *queue_cnt, struct PCB current_process, struct PCB new_process, int
    timestamp, int time_quantum)
134 {
135     if ((current_process.process_id == 0) &&
    (current_process.total_bursttime == 0) &&
    (current_process.execution_endtime == 0) &&
    (current_process.remaining_bursttime == 0) &&
    (current_process.execution_starttime == 0) &&
    (current_process.arrival_timestamp == 0) &&
    (current_process.process_priority == 0)) {// NULLPCB , New process
    priorty
136         new_process.execution_starttime = timestamp;
137         if (time_quantum <= new_process.total_bursttime){
138             new_process.execution_endtime = timestamp + time_quantum;
139 }
140         else {
141             new_process.execution_endtime = timestamp +
    new_process.total_bursttime;
142 }
143     new_process.remaining_bursttime = new_process.total_bursttime;
144     return new_process;
145 }
146     else {
147         new_process.execution_starttime = 0;
148         new_process.execution_endtime = 0;
149         new_process.remaining_bursttime = new_process.total_bursttime;
150         ready_queue[*queue_cnt] = new_process;
151         *queue_cnt = *queue_cnt + 1;
152         return current_process;
153 }
154 }
155
156 struct PCB handle_process_completion_rr(struct PCB ready_queue[QUEUEMAX],
    int *queue_cnt, int timestamp, int time_quantum) {
157     if (*queue_cnt > 0) {
158         struct PCB next_process;
159         int arr_timestamp = ready_queue[0].arrival_timestamp;
160         int temp_queue = 0;
161         for (int i = 1 ; i <= *queue_cnt - 1; i++){
162             if (arr_timestamp > ready_queue[i].arrival_timestamp){
```

```
163                 arr_timestamp = ready_queue[i].arrival_timestamp;
164                 temp_queue = i;
165             }
166         }
167         next_process = ready_queue[temp_queue];
168         if (*queue_cnt == 1) {
169             ready_queue[0].process_id = 0;
170             ready_queue[0].arrival_timestamp = 0;
171             ready_queue[0].total_bursttime = 0;
172             ready_queue[0].execution_starttime = 0;
173             ready_queue[0].execution_endtime = 0;
174             ready_queue[0].remaining_bursttime = 0;
175             ready_queue[0].process_priority = 0;
176         }
177         else
178             ready_queue[temp_queue] = ready_queue[*queue_cnt - 1];
179             *queue_cnt = *queue_cnt - 1;
180             next_process.execution_starttime = timestamp;
181             if (time_quantum <= next_process.remaining_bursttime){
182                 next_process.execution_endtime = timestamp +
time_quantum;
183             }
184             else {
185                 next_process.execution_endtime = timestamp +
next_process.remaining_bursttime;
186             }
187         return next_process;
188     }
189     else {
190         struct PCB null_PCB;
191         null_PCB.process_id = 0;
192         null_PCB.arrival_timestamp = 0;
193         null_PCB.total_bursttime = 0;
194         null_PCB.execution_starttime = 0;
195         null_PCB.execution_endtime = 0;
196         null_PCB.remaining_bursttime = 0;
197         null_PCB.process_priority = 0;
198         return null_PCB;
199     }
200 }
201
202
203
```

CPU Scheduling Lab

● Graded

Select each question to review feedback and grading details.

Student

Jahan Amrin

Total Points

5 / 6 pts

Autograder Score

5.0 / 6.0

Failed Tests

handle_process_completion_srtp (0/1)

Passed Tests

handle_process_arrival_pp (1/1)

handle_process_arrival_rr (1/1)

handle_process_arrival_srtp (1/1)

handle_process_completion_pp (1/1)

handle_process_completion_rr (1/1)