İ.T.Ü.
Computer Engineering Department

10.04.2013

BLG 312E – Computer Operating Systems

Homework 3

Submission Deadline: **01.05.2013**, **Wednesday**, until **17.00**.

- Homeworks submitted after the deadline and/or not submitted via the Ninova system will NOT be accepted.
- You are required to submit all 3 homeworks. Please note that you have to achieve at least 20 out of 100 points on a homework for its submission to be accepted. Homeworks with lower grades will NOT be considered as submitted. Submitting ONLY parts of the codes provided in class will NOT be sufficient to achieve a grade of 20.
- You are expected to work individually on all exams and homeworks. All forms of collaboration are discouraged and will be treated as cheating.

Where to submit: You should submit your homework program code via the Ninova system.
What to submit: You should submit a zip file of your homework via the Ninova system. The zip file should contain all your source files and one set of sample input and output files. No additional report file is required; however, it is expected from you to include code comments and explanations in your source files.
Program: You are required to write a program in C to implement the following simulation.

Problem definition: In a painting shop, boxes are being painted. Assume that each box is painted only in one color and that all boxes are of the same size. Changing colors in the painting process is expensive due to the added cost of wasted paint and solvents involved with each color change in addition to the extra time required for the changing process. Therefore, the objective is to order the boxes in a way to minimize the number of paint changeovers.

When boxes arrive at the painting shop, they are placed in a line based on the time of their arrival, until it is their turn to be painted. Painting starts only after all boxes have arrived and have been stored in the storage area. The box painting unit has a capacity of two, which means that two boxes of the same color can be painted at the same time. In order to minimize the number of paint changeovers, when a painting resource becomes available, a box having the same color as the previous boxes has to be painted. Boxes of the same color are painted as first-come-first-serve based on the order of their arrival. After both painting stations (resources) become empty, if there are no boxes of the same color, then a new box with another color should be painted. You can choose this box color using a first-come-first-serve approach based on the order of their arrival. Use the following values to simulate the time a box occupies a painting resource:
Red: 1 second, Green: 2 seconds, Blue: 2 seconds, Yellow: 1 second, Purple: 1 second, Orange: 2 seconds

Program implementation:

- You should use semaphores and shared memory where necessary for IPC. You should NOT use signals for synchronization.
- Each box should be modeled as a separate process. No threads will be used.

- A process (parent) creates all the required resources and IPC primitives, prepares an array showing the color for each box in a shared memory location and creates the box processes using the fork system call.
- At the end, after all boxes have been painted, the parent process removes all resources and IPC primitives.
- The parent process reads the sequence of box colors from an input file *inFile*. The first line of the input file shows the total number of boxes to be painted. Each line after this, contains one color code, showing the color of the corresponding box.
- Each box process will read its color from the array implemented on a shared memory, created and filled in by the parent process. The index of the box process will be the index with which it was created by the parent process. The color code on the array can be one of the following. R: red, G: green, B: blue, Y: yellow, P: purple, O: orange. You can assume that the colors are provided correctly. No error checking is required.
- Your program should output the results to an output file *outFile*. The first line of the output file will show the total number of paint color changes and the remaining lines will show the numbers and colors of the boxes in the order in which they have been painted. There should be one entry on each line. The pid of the corresponding process can be used as the box number.

Test: Your program will be tested in the form: ./program.exe <**`inFile`**> <**`outFile`**>

Please test and make sure to achieve expected results.

Example:
inFile:

```
12
R
R
G
B
G
B
O
Y
O
O
P
R
```

This means that 12 boxes will be painted. The first and second boxes will be painted red, the third box will be painted green, etc

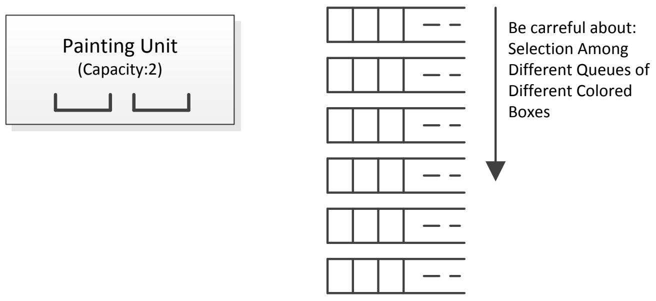
outFile: (It is assumed that the box processes are numbered consecutively starting from 100).

```
6
100 R
101 R
111 R
102 G
104 G
103 B
105 B
106 O
108 O
109 O
107 Y
110 P
```

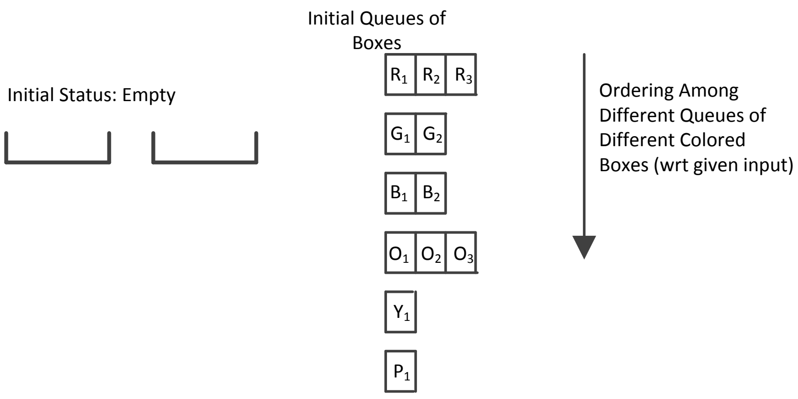
This means that the color has been changed 6 times, including the first time a color was loaded and the rest of the file shows the order in which the boxes were painted.

Demonstration:

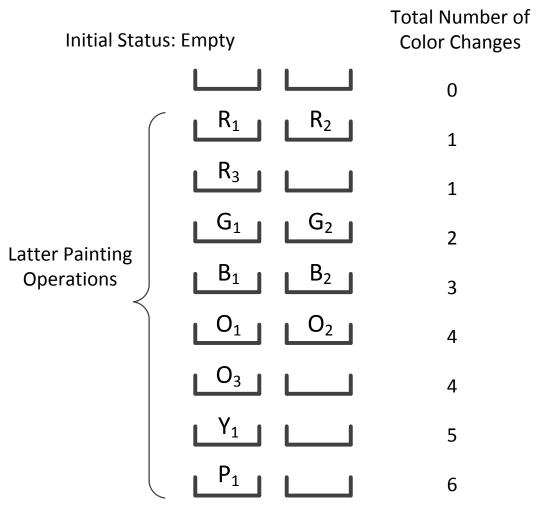
Initially, painting unit is available and queues are empty:


Painting Unit
(Capacity:2)
Be carreful about: Selection Among Different Queues of Different Colored Boxes

Boxes have come to the system (as in the example input file: inFile):


Initial Queues of Boxes
Initial Status: Empty
R1 R2 R3
G1 G2
B1 B2
O1 O2 O3
Y1
P1
Ordering Among Different Queues of Different Colored Boxes (wrt given input)

System started to work after all boxes have arrived and behaved as described and boxes are painted accordingly: (6 color changes are made in total)


Initial Status: Empty
Total Number of Color Changes
0
R1 R2 1
R3 1
G1 G2 2
B1 B2 3
O1 O2 4
O3 4
Y1 5
P1 6
Latter Painting Operations