УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

**“ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО”**

Кафедра «Информационные технологии»

Лабораторная работа №6

**«Программирование планировщика процессов»**

Выполнил: студент гр. ИТП-11

Воробьев В.А.

Принял: преподаватель

Карась О. В.

Гомель 2022

**Цель**: разработать программу, осуществляющую моделирование режима работы с разделение времени.

**Ход работы**

В соответствии с вариантом (выдается преподавателем), разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов. Моделирование режима разделения времени выполняется в соответствии с заданием (таблица 2.1). На экран выводится следующая информация:

1.Номер текущего кванта времени процессора;

2. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления;

3. Таблица планирования процессов с отображением текущего состояния процессов.

После запуска, программа должна диалоговом режиме, ввести информацию процессах – имя, длительность, приоритет, время появления. Для алгоритмов RR число квантов времени. Выполнение должно производиться в пошаговом режиме (по нажатию на кнопку). По окончанию работы процесса на экране должно выводится сообщение о его завершении «Процесс такой-то завершен».

В соответствии с вариантом надо сделать SJF не вытесняющий приоритетный, таблица процессов приведена ниже:

|  |  |  |  |
| --- | --- | --- | --- |
| Процесс | Продолжительность | Время появления в очереди | Приоритет |
| P0 | 3 | 3 | 4 |
| P1 | 9 | 0 | 1 |
| P2 | 2 | 0 | 2 |
| P3 | 3 | 4 | 3 |

Текст программы, написанной на языке *JavaScript*, представлен в листинге **А**. Так как у нас алгоритм SJF не вытесняющий, приоритетный, то чтобы выполнить процессы в правильном порядке, нам надо просто отсортировать их по времени продолжительности по возрастанию, но при этом учитывать и процесс. То есть если у процесса 1 время выполнения меньше, чем у процесса 2, но при этом у процесса 1 выше приоритет, то тогда он выполнится первым. После того, как мы их отсортировали, мы можем спокойно выполнить процессы в новом порядке. Если процесс выполняется, его состояние — “E”; ожидает своей очереди — “W”, исполнен — “C”. В результате мы увидим следующее. Согласно программе, нам необходимо отображать, когда процесс начался и когда закончился, и при выполнении мы увидим следующее:

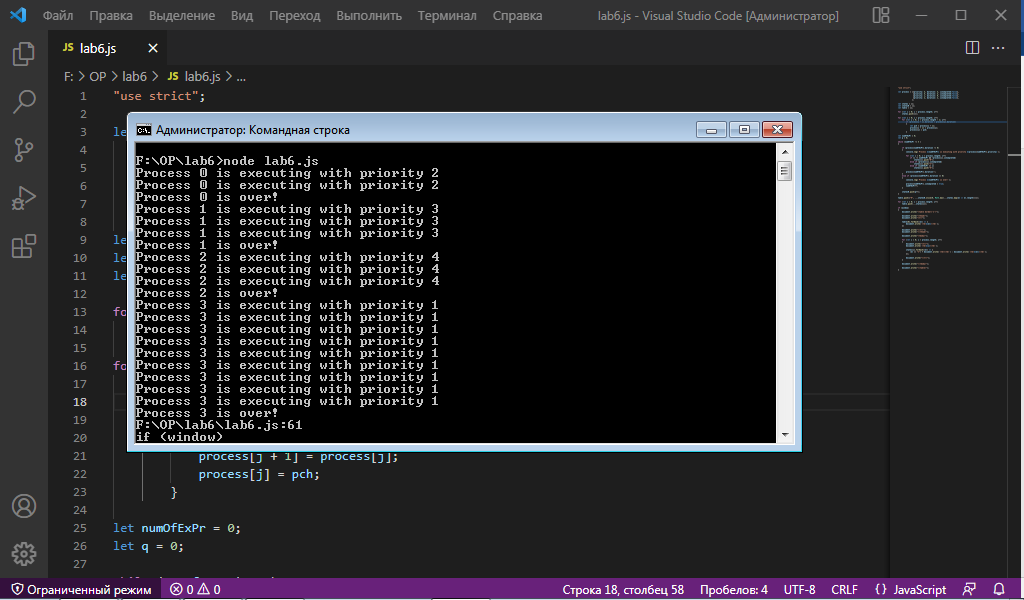


Рисунок 1 — Результат выполнения программы

Также мы увидим таблицу состояний процессов:

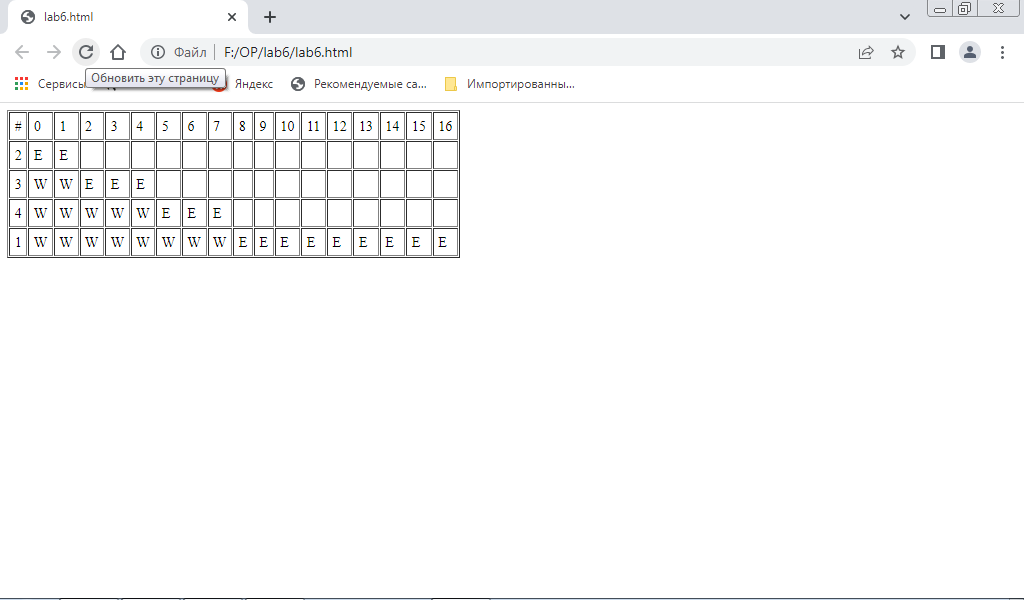


Рисунок 2 — Таблица состояний процессов

**Вывод**: в ходе работы была разработана программа, осуществляющая моделирование режима работы с разделение времени.

**ЛИСТИНГ А**

**КОД ПРОГРАММЫ, МОДЕЛУЮЩАЯ**

**РЕЖИМ РАБОТЫ РАЗДЕЛЕНИЯ ВРЕМЕНИ**

"use strict";

let process = [{priority: 3, duration: 3, isCompleted:false},

               {priority: 4, duration: 3, isCompleted:false},

               {priority: 1, duration: 9, isCompleted:false},

               {priority: 2, duration: 2, isCompleted:false},

              ];

let states = [];

let statesN = [];

let table = [];

for (let i = 0; i < process.length; i++)

    states.push([]);

for (let i = 0; i < process.length; i++)

    for (let j = 0; j < process.length - 1; j++)

        if (process[j + 1].duration < process[j].duration)

        {

            let pch = process[j + 1];

            process[j + 1] = process[j];

            process[j] = pch;

        }

let numOfExPr = 0;

let q = 0;

while (numOfExPr != 4 )

{

    if (process[numOfExPr].duration != 0)

    {

        console.log(`Process ${numOfExPr} is executing with priority ${process[numOfExPr].priority}`);

        for (let i = 0; i < process.length; i++)

            if (i != numOfExPr && !process[i].isCompleted)

                states[i].push(["W", process[i].priority]);

            else if (process[i].isCompleted)

                states[i].push(["C", process[i].priority]);

            else if (numOfExPr == i)

                states[i].push(["E", process[i].priority]);

        process[numOfExPr].duration--;

    }

    else if (process[numOfExPr].duration == 0)

    {

        console.log(`Process ${numOfExPr} is over!`);

        process[numOfExPr].isCompleted = true;

        numOfExPr++;

    }

    statesN.push(q++);

}

table.push(["#", ...statesN.slice(0, Math.max(...states.map(el => el.length)))]);

for (let i = 0; i < process.length; i++)

    table.push(...states[i]);

if (window)

{

    document.write("<table border='1'>");

    document.write("<thead>");

    document.write("<tr>");

    table[0].forEach((el) => {

        document.write(`<td>${el}</td>`);

    })

    document.write("</tr>");

    document.write("</thead>");

    document.write("<tbody>");

    for (let i = 0; i < process.length; i++)

    {

        document.write("<tr>");

        document.write(`<td>${states[i][0][1]}</td>`);

        states[i].forEach((el) => {

            (el[0] == "C") ? document.write(`<td></td>`) : document.write(`<td>${el[0]}</td>`);

        });

        document.write("</tr>");

    }

    document.write("</tbody>");

    document.write("</table>");

}