УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Кафедра «Информационные технологии»

Лабораторная работа №6

**«Программирование планировщика процессов»**

Выполнил: студент гр. ИТП-11

Клевцов А.О.

Принял: преподаватель-стажёр

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 алгоритм RR не вытесняющий приоритетный, таблица процессов приведена ниже:

|  |  |  |
| --- | --- | --- |
| Процесс | Продолжительность | Приоритет |
| P0 | 9 | 3 |
| P1 | 8 | 2 |
| P2 | 7 | 1 |

Текст программы, написанной на языке *JavaScript*, представлен в листинге **А**. Если у процессов одинаковый приоритет, то первым выполнится тот процесс, время выполнения которого меньше. По нажатию на кнопку вывода планирования выводится строка, состоящая из имён процессоров, которые по времени кванта, а затем передают очередь другому. В случае, когда время исполнения процесса меньше кванта, процесс прекращает работу и следует следующий процесс.

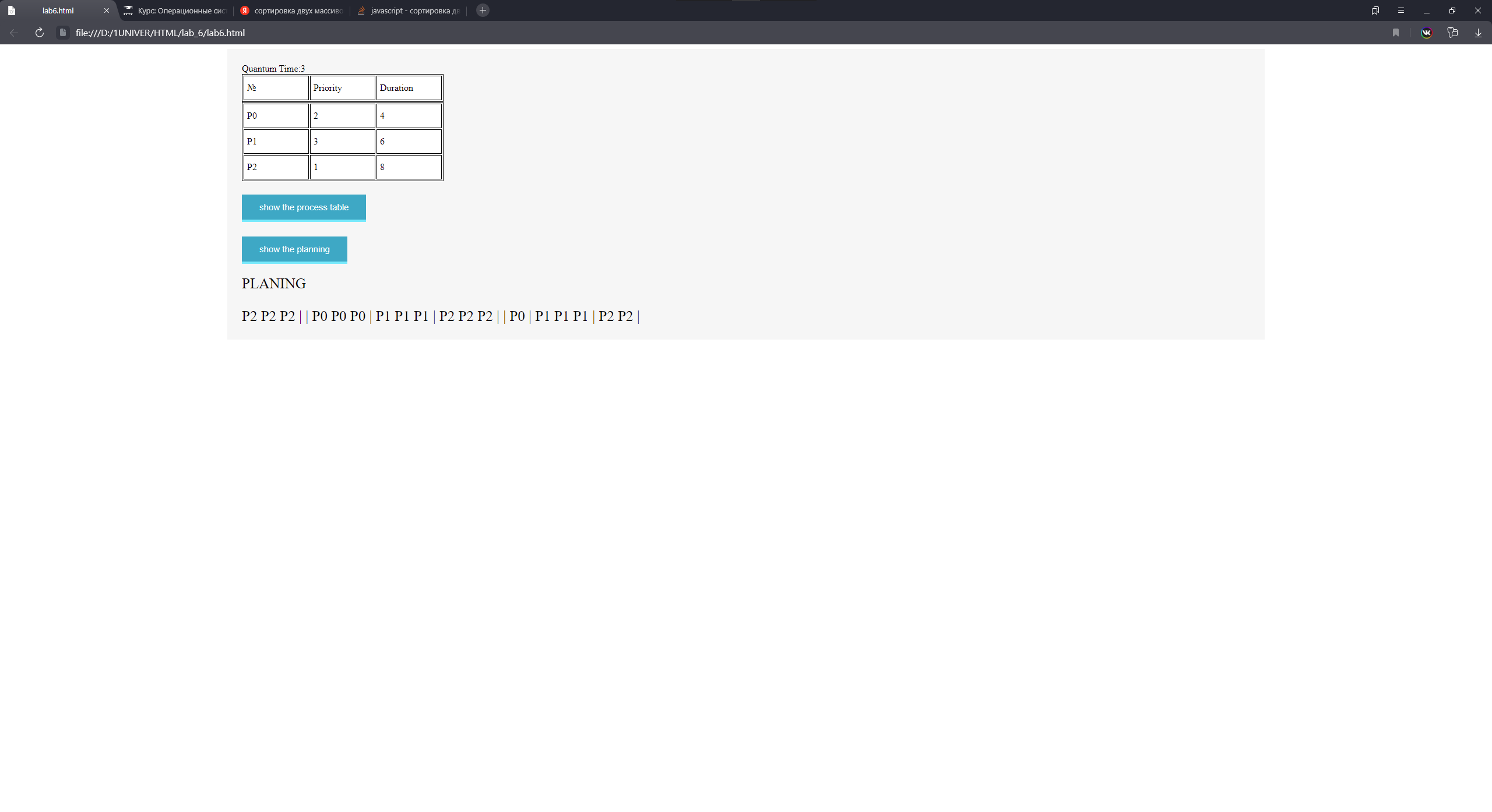


Рисунок 1 — Пример состояния на

текущий квант времени

После выполнения программы окончательно увидим следующее:

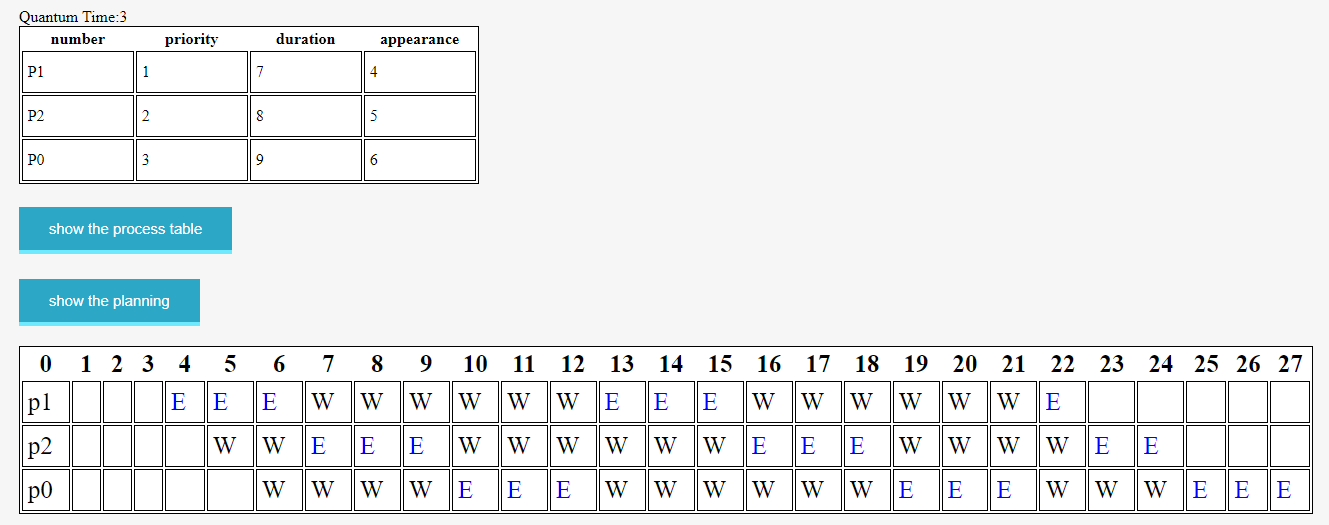


Рисунок 2 — Результат выполнения программы

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

**ЛИСТИНГ А**

**КОД ПРОГРАММЫ, МОДЕЛИРУЮЩЕЙ**

**РАБОТУ ПЛАНИРОВЩИК ЗАДАЧ**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title></title>

<link rel="stylesheet" type="text/css" href="lab6.css">

</head>

<body>

<script type="text/javascript">

"use strict";

let quantum = parseInt(prompt("Enter a time quantum: "));

let N = prompt("Enter the number of processes: ");

let number=[], priority = [], duration = [], appearance=[], outm = [];

let i,j,k,d,temp,burntime, min,c = 1;

for ( i = 0; i < N; i++) {

number.push(i);

priority.push(prompt("Enter priority of process: "));

duration.push(prompt("Enter duration of process: "));

appearance.push(prompt("Enter appearance of process: "));

}

for ( j = priority.length - 1; j > 0; j--) {

for ( i = 0; i < j; i++) {

if (priority[i] > priority[i + 1]) {

temp = number[i];

number[i] = number[i + 1];

number[i + 1] = temp;

temp = priority[i];

priority[i] = priority[i + 1];

priority[i + 1] = temp;

temp = duration[i];

duration[i] = duration[i + 1];

duration[i + 1] = temp;

temp = appearance[i];

appearance[i] = appearance[i + 1];

appearance[i + 1] = temp;

}

}

}

function ShowProcessTable() {

let table = document.createElement('table');

let thead = document.createElement('thead');

let tbody = document.createElement('tbody');

let trh = document.createElement('tr');

for ( j = 0; j < 4; j++) {

let th = document.createElement('th');

switch(j) {

case 0:th.innerHTML = 'number';break;

case 1:th.innerHTML = 'priority';break;

case 2:th.innerHTML = 'duration';break;

case 3:th.innerHTML = 'appearance';break;

}

trh.appendChild(th);

}

thead.appendChild(trh);

table.appendChild(thead);

for ( i = 0; i < N; i++) {

let tr = document.createElement('tr');

for ( j = 0; j < 4; j++) {

let td = document.createElement('td');

switch(j) {

case 0:td.innerHTML = 'P' + number[i];break;

case 1:td.innerHTML = priority[i];break;

case 2:td.innerHTML = duration[i];break;

case 3:td.innerHTML = appearance[i];break;

}

tr.appendChild(td);

}

tbody.appendChild(tr);

}

table.appendChild(tbody);

document.getElementById("tableDiv").appendChild(table);

document.getElementById("quantum").innerHTML = 'Quantum Time:' + quantum;

}

for ( i=0;i<number.length;i++) {

outm[i] = [];

}

min = appearance[0];

for ( i = 0; i < appearance.length; i++) {

if (appearance[i+1] < appearance[i]) {min = appearance[i+1];}

}

burntime = parseInt(min);

for ( i = 0; i < duration.length; i++) {

burntime += parseInt(duration[i]);

}

function Show() {

for (j = 0; j < burntime; j++) {

for (i = 0; i < number.length; i++) {

if (appearance[i] <=j && duration[i]>0) {

if (duration[i] >= quantum ) {

for (k = 0; k < quantum; k++) {

outm[i][j] = 'E';

duration[i]--;

for(d = 0; d<number.length; d++){

if (appearance[d] <=j && duration[d]>0 &&outm[d][j]!='E' ) {outm[d][j] = 'W';}

}

j++;

}

}

else {

for (duration[i]; duration[i]>0; duration[i]--) {

outm[i][j] = 'E';

for(d = 0; d<number.length; d++){

if (appearance[d] <=j && duration[d]>0 &&outm[d][j]!='E' ) {outm[d][j] = 'W';}

}

j++;

}

}

}

}

}

for ( i=0;i<number.length;i++) {

for (j = 0; j < burntime; j++) {

if (outm[i][j]=='E' && typeof outm[i][j+1] == "undefined") {

outm[i].splice(j+1,1)

}

if (outm[i][j]=='W' && typeof outm[i][j+1] == "undefined") {

outm[i].splice(j+1,1)

}

}

}

let table = document.createElement('table');

let tbody = document.createElement('tbody');

let thead = document.createElement('thead');

let trh = document.createElement('tr');

for ( j = 0; j < burntime; j++) {

let th = document.createElement('th');

th.innerHTML = j;

trh.appendChild(th);

}

thead.appendChild(trh);

table.appendChild(thead);

for ( i = 0; i < number.length; i++) {

let tr = document.createElement('tr');

for ( j = 0; j < burntime; j++) {

let td = document.createElement('td');

if (j==0) {td.innerHTML = 'p'+number[i]}

else if (outm[i][j]=='E') {td.innerHTML = '<font color="blue">' + outm[i][j] + '</font>';}

else if (outm[i][j]=='W') {td.innerHTML = outm[i][j];}

tr.appendChild(td);

}

tbody.appendChild(tr);

}

table.appendChild(tbody);

document.getElementById("Div").appendChild(table);

}

</script>

<div class="container">

<div id="quantum"></div>

<div id="tableDiv">

</div>

<br><input type="button" value="show the process table" class="dws" onclick="ShowProcessTable()">

<br><input type="button" value="show the planning" class="dws" onclick="Show()">

<div id="Div" ></div>

</div>

</body>

</html>.dws{

padding: 13px 30px;

margin: 5px 0 20px 0;

font-size: 15px;

color: #fff;

background-color: #2ca8c6;

border: none;

border-bottom: 4px solid #6ee9fd;

cursor: pointer;

}

.dws:hover{

transition: all 0.5s;

background: #fff;

color: #2c536c;

}

.container{

margin: 0 15% 0 15%;

background: #f6f6f6;

padding: 1%;

}

table{

background: white;

border: 1px solid;

}

td{

padding: 5px;

border: 1px solid;

width: 100px;

height: 30px;

}

tr{

padding: 5px;

border: 1px solid;

}

#Div{

font-size: 25px;

}