**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине **«Операционные системы»**

на тему: **«Программирование планировщиков процессов»**

Выполнила студентка гр. ИТИ-11:  
Кононович Е.М.  
 Принял преподаватель:  
 Карась О.В.

Гомель 2022

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

**Задание**

Разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов.**(Алгоритм SJF. Не вытесняющий)** Моделирование режима разделения времени выполняется в соответствии с заданием 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, время появления.

**Ход выполнения работы**

Листинг программы:

<html>

<head>

<title>Лаба 6</title>

<style>

td, th, table {

border: 2px solid black;

}

\*{

margin-left:100;

margin-top: 20;

font-size: 16px;

font-family: "Arial";

background: #f2efc4;

}

</style>

<script>

var readyProcess = [];

var plannedProcess = [];

var cProcess;

var cTime = 0;

function addProcess(name, time)

{

var flag = 0;

if (!name || !time)

{

return;

}

if(cProcess && name == cProcess['name'])

{

alert('Process exist!');

return;

}

readyProcess.forEach(function(item){

if (item['name'] == name)

{

alert('Process exist!');

flag = 1;

}

});

if (!flag) {

readyProcess.push({ name: name, time: time });

readyProcess.sort(function (a, b) {

return a['time'] - b['time'];

});

}

}

function addPlanned(apirTime, name, time)

{

var flag = 0;

if(cProcess && name == cProcess['name'])

{

alert('Process exist!');

return;

}

readyProcess.forEach(function(item){

if (item['name'] == name)

{

alert('Process exist!');

flag = 1;

}

});

plannedProcess.forEach(function(item){

if (item['name'] == name)

{

alert('Process exist!');

flag = 1;

}

});

if (!flag) {

plannedProcess.push({ apirTime: apirTime, name: name, time: time });

}

}

function tick()

{

let frame = [];

checkPlanned();

if (cProcess == undefined)

{

if (readyProcess.length)

{

cProcess = readyProcess[0];

readyProcess.splice(0, 1);

}

else

{

return draw(frame);

}

}

if (!cProcess['time'])

{

cProcess = undefined;

return tick();

}

cProcess['time']--;

readyProcess.forEach(function(item){

frame.push({name: item['name'], status: 'Г'});

});

frame.push({name: cProcess['name'], status: 'И'});

draw(frame);

}

function checkPlanned()

{

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

{

if(plannedProcess[i]['apirTime']==cTime)

{

addProcess(plannedProcess[i]['name'], plannedProcess[i]['time']);

}

}

}

function draw(frame)

{

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

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

th.innerHTML = cTime;

document.getElementById('header').appendChild(th);

frame.forEach(function(item)

{

if (document.getElementById(item['name']))

{

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

th.innerHTML = item['status'];

document.getElementById(item['name']).appendChild(th);

}

else

{

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

tr.id = item['name'];

tr = table.appendChild(tr);

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

name.innerHTML = item['name'];

tr.appendChild(name);

for (var i = 0; i < cTime; i++)

{

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

blank.innerHTML = ' ';

tr.appendChild(blank);

}

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

th.innerHTML = item['status'];

tr.appendChild(th);

}

});

table.childNodes.forEach(function(item)

{

if (item.id)

{

if(!frame.filter(function (p) { return p['name'] == item.id}).length)

{

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

blank.innerHTML = ' ';

item.appendChild(blank);

}

}

});

cTime++;

}

</script>

</head>

<body>

<table id="table">

<tr id="header">

<th></th>

</tr>

</table>

<button onclick="tick()">Далее</button><br><br><br>

Время появления процесса: <input type="text" id="apirTime"><br>

Название процесса: <input type="text" id="name"><br>

Время процесса: <input type="text" id="time"><br>

<button onclick="document.getElementById('apirTime').value ? addPlanned(document.getElementById('apirTime').value, document.getElementById('name').value, document.getElementById('time').value) : addProcess(document.getElementById('name').value, document.getElementById('time').value);">Добавить процесс</button>

</body>

</html>

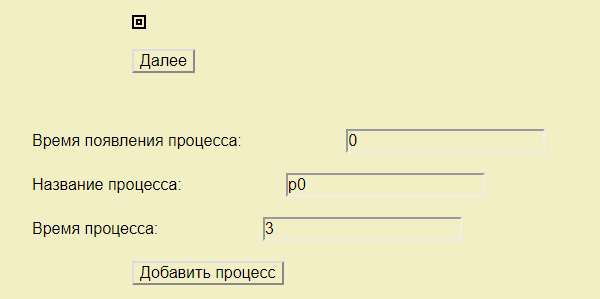


Рисунок 1 – Ввод данных процессора p0

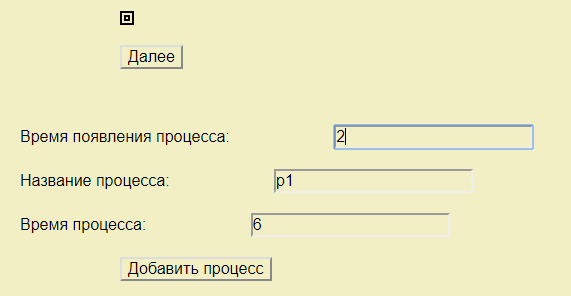


Рисунок 2 – Ввод данных процессора p1

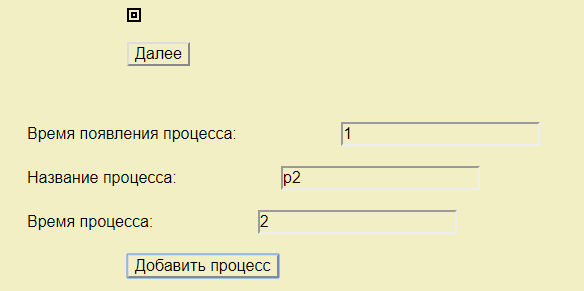


Рисунок 3 – Ввод данных процессора p2

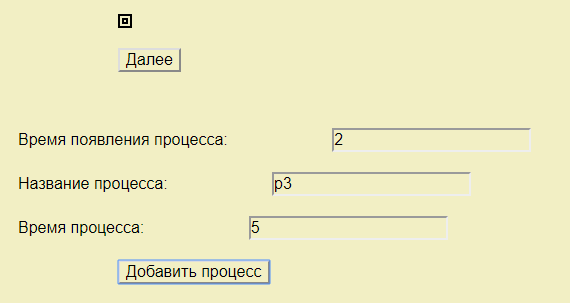


Рисунок 4 – Ввод данных процессора p3

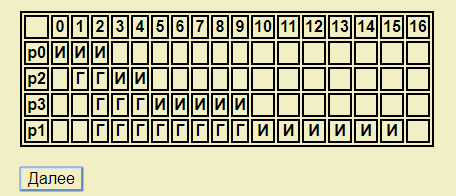


Рисунок 5 – Таблица планирования процессов SJF(не вытисняющий)

**Вывод:** Разработали программу, осуществляющую моделирование режима работы с разделением времени.