**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 № 5

по дисциплине «Операционные системы»

на тему: «Планирование процессов»

Выполнил: студент гр. ИТП-11

Ковтун А.В.

Принял: преподаватель

Карась О.В.

Гомель 2022

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

**Задание:** моделирование алгоритма *RR*. Не вытесняющий, приоритетный.



Рисунок 1.1 – Ввод данных о процессах



Рисунок 1.2 – Ввод данных о процессах



Рисунок 2 – Вывод таблиц с результатом

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

**Приложение:**

*JavaScript*:

const main = document.getElementById("main");

let cols;

let testColsLabel = document.createElement("label");

testColsLabel.innerText = "введите количество процессов";

let buttonCols = document.createElement("input");

buttonCols.innerText = "ввести";

testColsLabel.appendChild(buttonCols);

buttonCols.addEventListener("keydown", drawProcessEnter);

main.appendChild(testColsLabel);

function drawProcessEnter(e) {

if(e.code === "Enter") {

main.removeChild(testColsLabel);

cols = +this.value+1;

createTable(main, 4, cols, drawTable, "chooseTable", drawReadyButton);

fillTable();

}

}

function createTable(parent, cols, rows, drawTableFunc, id, drawBtnFunc) {

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

table.id=`${id}`;

for (let i = 0; i < rows; i++) {

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

for (let j = 0; j < cols; j++) {

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

tr.appendChild(td);

drawTableFunc(j, td, i);

}

table.appendChild(tr);

}

parent.appendChild(table);

if(drawBtnFunc)

drawBtnFunc();

}

function drawReadyButton() {

const btn = document.createElement("button");

btn.innerText = "Готово";

btn.addEventListener("click", start);

main.appendChild(btn);

return true

}

function drawTable(j, td, i) {

if (i === 0) {

let p = document.createElement("p");

switch (j) {

case 0:

p.innerText = `Имя`;

td.appendChild(p);

break;

case 1:

p.innerText = `Длительность`;

td.appendChild(p);

break;

case 2:

p.innerText = `Приоритет`;

td.appendChild(p);

break;

case 3:

p.innerText = `Появление`;

td.appendChild(p);

break;

default:

break;

}

} else {

switch (j) {

case 0:

let p = document.createElement("p");

p.innerText = `P${i}`;

td.appendChild(p);

break;

default:

let inp = document.createElement("input");

inp.type = "number";

td.appendChild(inp);

break;

}

}

}

class Process {

id = 0;

duration = 0;

priority = 0;

over;

appearance = 0;

given = 0;

}

let processes = [];

let dataPresses = [];

function start() {

let t = document.getElementsByTagName("input");

for (let i = 0; i < cols - 1; i++) {

processes[i] = new Process();

dataPresses[i] = new Process();

let j = i \* 3;

processes[i].id = i;

processes[i].given = +t[j].value;

processes[i].duration = +t[j].value;

processes[i].priority = +t[j + 1].value;

processes[i].appearance = +t[j + 2].value;

dataPresses[i].id = i;

dataPresses[i].given = +t[j].value;

dataPresses[i].duration = +t[j].value;

dataPresses[i].priority = +t[j + 1].value;

dataPresses[i].appearance = +t[j + 2].value;

}

make();

}

function make() {

let tick = 0;

let queue = createQueue(tick);

let i = 0;

while (processes.length > 0) {

for (let q = 0; q < 3; q++) {

addData(queue, i, tick);

if (processes.length === 0) {

break;

}

if (i < queue.length && processes[queue[i]]) {

runProcess(queue, i, tick);

} else {

queue = createQueue(tick);

i = 0;

runProcess(queue, i, tick);

}

if (processes[queue[i]]) {

console.log(

`p${processes[queue[i]].id + 1}: `,

processes[queue[i]].duration, `, `, tick

);

}

tick++;

}

queue = createQueue(tick);

if (i < queue.length - 1) i++;

else i = 0;

}

console.log(dataSet);

createTable(main, dataSet.now.length+1, cols, logTable, "logTable");

createTable(main, 5, cols, slowLogTable, "slowTable", drawNextButton);

}

function drawNextButton() {

const btn = document.createElement("button");

btn.id="nextBtn";

btn.innerText = `q=${nexttmp}. Далее`;

btn.addEventListener("click", next);

main.appendChild(btn);

return true

}

class data {

now = [];

}

let dataSet = new data;

let nexttmp = 0;

function next() {

main.removeChild(document.getElementById("slowTable"));

main.removeChild(document.getElementById("nextBtn"));

console.log(nexttmp);

dataPresses[dataSet.now[nexttmp]].given--;

nexttmp++;

if(nexttmp===dataSet.now.length) {

nexttmp=0;

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

dataPresses[i].given = dataPresses[i].duration;

}

createTable(main, 5, cols, slowLogTable, "slowTable", drawNextButton);

}

function addData(queue, i, tick) {

if(processes[queue[i]] && processes[queue[i]].duration>=1)

dataSet.now.push(processes[queue[i]].id);

}

function runProcess(queue, i, tick) {

if (processes[queue[i]].duration === 0) {

processes.splice(queue[i], 1);

queue.splice(i, 1);

i++;

} else {

processes[queue[i]].duration--;

if (processes[queue[i]].duration === 0)

dataPresses[processes[queue[i]].id].over = tick;

}

}

function createQueue(tick) {

let maxPriority = -Infinity;

let queue = [];

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

if (

maxPriority < processes[i].priority &&

tick >= processes[i].appearance

) {

maxPriority = processes[i].priority;

}

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

if (

maxPriority === processes[i].priority &&

tick >= processes[i].appearance

) {

queue.push(i);

}

return queue;

}

function logTable(j, td, i) {

if (i === 0) {

let p = document.createElement("p");

switch (j) {

case 0:

p.innerText = `Имя`;

td.appendChild(p);

break;

default:

p.innerText = `${j}`;

td.appendChild(p);

break;

}

}

else {

switch (j) {

case 0:

let p2 = document.createElement("p");

p2.innerText = `P${i}`;

td.appendChild(p2);

break;

default:

let p1 = document.createElement("p");

if(i-1===dataSet.now[j-1])

p1.innerText = "И";

else {

if(dataPresses[i-1].appearance<=j && j<dataPresses[i-1].over)

p1.innerText = "Г";

}

td.appendChild(p1);

break;

}

}

}

function slowLogTable(j, td, i) {

if (i === 0) {

let p = document.createElement("p");

switch (j) {

case 0:

p.innerText = `Имя`;

td.appendChild(p);

break;

case 1:

p.innerText = `Длительность`;

td.appendChild(p);

break;

case 2:

p.innerText = `Приоритет`;

td.appendChild(p);

break;

case 3:

p.innerText = `Появление`;

td.appendChild(p);

break;

case 4:

p.innerText = `Осталось`;

td.appendChild(p);

break;

default:

break;

}

}

else {

let p = document.createElement("p");

switch (j) {

case 0:

p.innerText = `P${i}`;

td.appendChild(p);

break;

case 1:

p.innerText = `${dataPresses[i-1].duration}`;

td.appendChild(p);

break;

case 2:

p.innerText = `${dataPresses[i-1].priority}`;

td.appendChild(p);

break;

case 3:

p.innerText = `${dataPresses[i-1].appearance}`;

td.appendChild(p);

break;

case 4:

p.innerText = `${dataPresses[i-1].given}`;

td.appendChild(p);

break;

}

}

}

*HTML*:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

<link href="style.css" rel="stylesheet">

</head>

<body>

<div id="main"></div>

<script src="main.js"></script>

</body>

</html>

*CSS*:

table td {

padding-left: 7px;

padding-right: 7px;

height: 15px;

border: 2px solid black;

}

table {

border-collapse: collapse;

}

input {

width: 30px;

}

#main {

display: grid;

justify-content: center;

gap: 10px;

}