**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 № 9

по дисциплине «Операционные системы»

на тему: «Простейшие схемы управления памятью»

Выполнил: студент гр. ИТП-11

Ковтун А.В.

Принял: преподаватель

Карась О.В.

Гомель 2022

**Цель работы:** изучение алгоритмов управления памятью, разработка программы менеджера памяти.

**Задание:** разработать программу, реализующую заданный алгоритм выделения памяти. Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданного алгоритма (таблица). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объём свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворенных запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаѐтся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемого блока и имя процесса.

4. Реализовать возможность последовательной записи/чтения информации в/из выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.

5. Организовывать циклическое выделение и освобождение памяти. При этом случайным образом задается количество выделяемых блоков и их размер.

Сегментная схема организации памяти. Для каждого процесса создавать 2 сегмента: сегмент кода (3 кб) и сегмент данных.



Рисунок 1 – Результат выполнения программы

По нажатию на кнопку «выбрать случайный процесс» выбирается случайный процесс для «исполнения», если его нет в физической памяти, то запускается функция которая освобождает память нужное количество памяти и загружает выбранный процесс.



Рисунок 2 – Выбор случайного процесса

**Вывод:** изучены алгоритмы управления памятью, разработана программа менеджера памяти.

Листинг:

JavaScript:

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

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

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

const spaceCounter = document.getElementById("freeSpace");

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

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

let memorySegmentCounter = 0;

function drawMainMenu() {

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

const nameLabel = document.createElement("label");

const sizeLabel = document.createElement("label");

const sizeInput = document.createElement("input");

const nameInput = document.createElement("input");

nameLabel.innerText = `Название`;

addBtn.innerText = "Добавить процесс";

sizeLabel.innerText = "Размер сегмента данных"

addBtn.addEventListener("click", () =>addProcess(nameInput.value, sizeInput.value));

nameLabel.appendChild(nameInput);

sizeLabel.appendChild(sizeInput);

main.appendChild(sizeLabel);

main.appendChild(nameLabel);

main.appendChild(addBtn);

}

drawMainMenu();

const physicMemory = [];

physicMemory.length = 40;

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

physicMemory[i]=0;

class segment {

name;

address;

size;

type;

apeals=0;

}

let processes = [];

function addProcess(name, size) {

processes.push(new segment());

processes[processes.length-1].name = name;

processes[processes.length-1].size = +size;

processes[processes.length-1].type = "data";

findSpace(processes[processes.length-1]);

processes.push(new segment());

processes[processes.length-1].name = name;

processes[processes.length-1].size = 3;

processes[processes.length-1].type = "code";

findSpace(processes[processes.length-1]);

redrawSegmentTable();

redrawPhysicTable();

spaceCounter.innerText = freeSpaceCounter();

satisfied.innerText = percentSatisfiedSegments();

}

function findSpace(process) {

let i=0;

let freeMemory;

let iStart;

while (i<physicMemory.length) {

freeMemory = true;

iStart=i;

while (i<process.size+iStart) {

if(physicMemory[i]===0) {

i++;

}

else {

freeMemory = false;

break;

}

}

if(i===process.size+iStart && freeMemory === true) {

i-=process.size;

process.address = i;

for(i;i<process.size+iStart;i++) {

physicMemory[i]=process.name;

}

break;

}

i++;

}

}

function percentSatisfiedSegments() {

memorySegmentCounter=0;

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

if(Number.isInteger(processes[i].address))

memorySegmentCounter++;

return `${(memorySegmentCounter/processes.length).toFixed(2)\*100}% сегментов размещено`;

}

function redrawSegmentTable() {

if(document.getElementById("mainDiv"))

segmentTable.removeChild(document.getElementById("mainDiv"));

const mainDiv = document.createElement("div");

mainDiv.id = "mainDiv";

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

mainDiv.appendChild(createSegment(processes[i]));

}

segmentTable.appendChild(mainDiv);

}

function createSegment(process) {

const div = document.createElement("div");

div.innerText = `Сегмент ${process.name} (${process.type}) - ${process.size}Кб`;

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

deleteButton.id = process.name;

deleteButton.innerText = "удалить процесс";

deleteButton.addEventListener("click", () => deleteProcess(deleteButton))

div.appendChild(deleteButton);

return div

}

function deleteProcess(btn) {

let deleteId;

let btnId = btn.id;

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

if(btnId===processes[i].name) {

deleteId = i;

for(let i = processes[deleteId].address; i<processes[deleteId].address+processes[deleteId].size; i++)

physicMemory[i]=0;

processes.splice(deleteId, 1);

i--;

}

}

redrawPhysicTable();

redrawSegmentTable();

spaceCounter.innerText = freeSpaceCounter();

satisfied.innerText = percentSatisfiedSegments();

}

function redrawPhysicTable() {

if(physicTable.childElementCount>0)

physicTable.removeChild(document.getElementById("physicMainDiv"));

const mainDiv = document.createElement("div");

mainDiv.id = "physicMainDiv";

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

drawPhysicDiv(i, mainDiv);

}

physicTable.appendChild(mainDiv)

}

function drawPhysicDiv(i, mainDiv) {

const div = document.createElement("div");

div.className = "physicDiv"

if(physicMemory[i]===0) {

div.style.backgroundColor = "white";

}

else {

div.style.backgroundColor = "yellow";

div.innerText = `${physicMemory[i]}`;

}

mainDiv.appendChild(div);

}

function freeSpaceCounter() {

let free=0;

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

if (physicMemory[i]===0)

free++;

return `${free}Кб из ${physicMemory.length}Кб свободно`

}

function chooseRandomProcess() {

let index = Math.round(Math.random()\*(processes.length-1));

if(Number.isInteger(processes[index].address))

processes[index].apeals++;

else {

loadProcess(index)

redrawPhysicTable();

processes[index].apeals++;

spaceCounter.innerText = freeSpaceCounter();

satisfied.innerText = percentSatisfiedSegments();

}

}

function loadProcess(index) {

let processType;

if(processes[index].type === "code")

processType = -1;

else processType = 1;

while(Number.isInteger(processes[index].address) === false) {

clearPhysicMemory();

findSpace(processes[index]);

}

while((Number.isInteger(processes[index+processType].address) === false)) {

clearPhysicMemory();

findSpace(processes[index+processType]);

}

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

if(Number.isInteger(processes[i].address) === false)

findSpace(processes[i]);

}

}

function clearPhysicMemory() {

let min = Infinity;

let index;

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

if(processes[i].apeals<min && Number.isInteger(processes[i].address)) {

min=processes[i].apeals;

index = i;

}

}

for(let i = processes[index].address; i<processes[index].address+processes[index].size; i++)

physicMemory[i]=0;

processes[index].address = undefined;

}

document.getElementById("clearMemory").addEventListener("click", chooseRandomProcess)

HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Сегментная схема управления памятью</title>

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

</head>

<body>

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

<h3>Список сегментов:</h3>

<div id="segmentTable"></div>

<h3>Физическая память:</h3>

<div id="physicTable"></div>

<p id="freeSpace"></p>

<p id="satisfied"></p>

<button id="clearMemory">очистить память</button>

<p id="choosenProcess"></p>

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

</body>

</html>

CSS:

#main {

display: grid;

justify-content: center;

}

label {

display: grid;

justify-content: space-between;

}

#physicMainDiv {

display: grid;

grid-template-columns: repeat(40, 2vw);

}

.physicDiv {

width: 2vw;

height: 2vh;

display: grid;

justify-content: center;

align-content: center;

border: black 2px solid;

}