Учреждение образования Республики Беларусь

«Гомельский государственный технический университет им. П. О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчет по лабораторной работе №9

по дисциплине «Операционные системы»

Выполнил: студент группы ИТП-11

Половцев М. С.

Принял: преподаватель-стажер

Карась О. В.

Гомель, 2022

**Цель:** 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 удовлетворённых запросов (%).

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

Вариант задания:

|  |  |
| --- | --- |
| Вариант | Задание |
| 6 | Сегментная схема организации памяти. Для каждого процесса  создавать 3 сегмента: сегмент стека (1 кб), сегмент кода  (1 кб) и сегмент данных. |

На рисунке 1 представлен результат работы менеджера памяти:



Рисунок 1 – Результат работы менеджера памяти (добавление, удаление, вывод информации о процессе)

**Вывод:** изучил алгоритмы управления памятью, разработал программу менеджера памяти.

ЛИСТИНГ

let memory = [];

memory.length = 50;

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

{

memory[i] = 0;

}

function segment(name, size, start, number) {

this.name = name;

this.size = size;

this.start = start;

this.number = number;

}

let processes = [];

let freeSize = 50;

let req = 0;

let sucreq = 0;

let per = 0;

let n = 0;

function addProcess() {

if (parseInt(document.getElementById('size').value) + 2 <= freeSize)

{

let name = document.getElementById('name').value;

let size = parseInt(document.getElementById('size').value);

let freeSpace;

let start;

let j;

let f = 0;

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

{

if (memory[i] == 0)

{

freeSpace = 0;

start = i;

j = i;

while (memory[j] == 0)

{

freeSpace++;

j++;

}

if (size + 2 <= freeSpace)

{

for (let k = start; k < start + size + 2; k++)

{

memory[k] = name;

}

f = 1;

break;

}

}

}

if (f == 0)

{

alert('Недостаточно места для добавления процесса');

req++;

per = Math.round(sucreq / req \* 100);

document.getElementById('span2').innerText = ` ${req}`;

document.getElementById('span3').innerText = ` ${sucreq} (${per} %)`;

return;

}

processes.push(new segment);

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

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

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

processes[processes.length - 1].number = n;

freeSize -= (parseInt(processes[processes.length - 1].size) + 2);

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

span.innerHTML = `<p>Процесс ${processes[processes.length - 1].name} (данные) - ${processes[processes.length - 1].size} Кб</p>`;

span.innerHTML += `<p>Процесс ${processes[processes.length - 1].name} (стек) - 1 Кб</p>`;

span.innerHTML += `<p>Процесс ${processes[processes.length - 1].name} (код) - 1 Кб</p>`;

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

div.setAttribute('id', `div${n}`);

div.append(span);

let parElem = document.getElementById('list');

parElem.append(div);

req++;

sucreq++;

per = Math.round(sucreq / req \* 100);

document.getElementById('span1').innerText = ` ${freeSize} Кб`;

document.getElementById('span2').innerText = ` ${req}`;

document.getElementById('span3').innerText = ` ${sucreq} (${per} %)`;

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

createTable();

n++;

}

else

{

alert('Недостаточно места для добавления процесса');

req++;

per = Math.round(sucreq / req \* 100);

document.getElementById('span2').innerText = ` ${req}`;

document.getElementById('span3').innerText = ` ${sucreq} (${per} %)`;

return;

}

}

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

add.addEventListener('click', addProcess);

function deleteProcess() {

let name = document.getElementById('delname').value;

let f = 0;

let n;

let div;

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

{

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

{

freeSize += (parseInt(processes[i].size) + 2);

for (let j = parseInt(processes[i].start); j < parseInt(processes[i].start) + parseInt(processes[i].size) + 2; j++)

{

memory[j] = 0;

}

n = processes[i].number;

div = document.getElementById(`div${n}`);

div.innerHTML = '';

processes.splice(i, 1);

f = 1;

document.getElementById('span1').innerText = ` ${freeSize} Кб`;

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

createTable();

break;

}

}

if (f == 0)

{

alert('Процесса с таким именем не существует');

}

}

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

del.addEventListener('click', deleteProcess);

function createTable() {

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

table.setAttribute('border', '1');

table.innerHTML = '<tr>';

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

{

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

td.setAttribute('width', `15px`);

if (memory[i] != 0)

{

td.setAttribute('bgcolor', `#00FF00`);

td.innerText = memory[i];

}

table.append(td);

}

table.innerHTML += '</tr>';

document.getElementById('table').append(table);

}