****

2014

**Project 4: Memory Allocation and Garbage Collection**

**Bahadir Can Yildiz 100301034**

1 – Problem Definition:

Simulating working principal of Memory Allocation and Garbage Collection that includes allocating memories of processes, and collecting garbage when there is no place to put next process in memory.

2 - Explanation of the algorithms used to solve that problem:

For dealing with this problem, many algorithms were planned for reaching the solution. Program is developed as a Client-Side Web Application which is coded with HTML5 and Javascript (also used Jquery Framework). It works with many browsers yet I strongly recommend to work it with Firefox.

!!! Complete Source Code with Explanations:

Index.html(this is the coding of our interface):

<!DOCTYPE html>

<html>

<head>

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

<script type="text/javascript" src="jquery-2.0.3.js"></script>

<script type="text/javascript" src="scripts.js"></script>

<title>Welcome to the John`s Secrecy</title>

</head>

<body>

<div id="header">Memory Allocation and Garbage Collection, Bahadir Can Yildiz - 100301034</div>

<div id="regcount" class="quote">

<p>Process Quantity:</p>

<input id="processtotal" type="number" name="points" class="bar"/><br/>

<p>Total Memory:</p>

<input type="number" id="totalmem" class="bar"/><br/><br/>

<center><input type="button" value="Memory Allocation" onclick="return olustur();"/></center>

</div>

<div id="guide" class="quote">

<p>Total Cpu Time:<span id="cputime">0</span> ms</p>

</div>

<div id="malloc" class="quote">

<p>Memory Allocation Logs:</p>

<p>Allocated Memory Space:<span id="ams">0</span> kb</p>

<p>Free Memory Space:<span id="fms"></span> kb</p>

</div>

<div id="gclog" class="quote">

<p>Garbage Collection Logs:</p>

</div>

<div id="processes" class="quote">

<p>All Processes:</p>

<div id='memory'>

</div>

<center><input type='button' onclick='return calistir();' value='Stepper'></input></center>

</div>

</body>

</html>

Scripts.js(Programming Side):

//Memory Allocation and Garbage Collection, Bahadir Can Yildiz - 100301034

$(document).ready(function(){

$("#header").slideDown(1000);//this is for making my application fabulous.

});

var GC = new Object(); //(Garbage Collect)this is our Main Object that contains methods, processes and information about that processes.

GC.P = new Array(); //under GC, there are array objects called P, which defines every process.

function yaz(){ //for printing Process quantity value dynamically above the "create" button.

$("#sayi").html($("#processtotal").val());

}

function olustur(){//for creating memory and it`s informations, instructions and their elements for giving commands.

if ($("#totalmem").val()=="" || $("#processtotal").val()==""){

alert("Enter values properly please!");

}

else{

GC.count = parseInt($("#processtotal").val()); //for getting values we wrote to the program

GC.mem = parseInt($("#totalmem").val());

$("#fms").html(GC.mem);

GC.stepcount = 0; //this counts for every step we made. We use it for putting our processes.

GC.totalcpu = 0; // this calculates the cpu time of total processes.

GC.usedmem = 0; // for calculating our used memory. We use it for finding remaining memory.

GC.arrayP = new Array(); // array that shows us which processes are in memory.

for(var i=0;i<GC.count;i++){ // creates all the processes you demanded and gives them their random values

GC.P[i] = new Object();

GC.P[i].color = getRandomColor(); //gets random color for every process

GC.P[i].cpu = getRandomCpu(); //gets random cpu for every process

GC.P[i].mem = getRandomMem(); //gets random memory for every process

}

$("#memory").css("width","1000px");

GC.stepper(); //Makes the first step automatically after you give parameters to the program.

}

}

function calistir(){ //algorithm that captures all methods to make a step.

GC.stepper();

}

function getRandomColor() { //for giving random colors for every process. Combines 6 hexadecimal characters to make one.

var letters = '0123456789ABCDEF'.split('');

var color = '#';

for (var i = 0; i < 6; i++ ) {

color += letters[Math.round(Math.random() \* 15)];

}

return color;

}

function getRandomMem(){ // for getting random memory value for every process.

for (var i=0;i<0.1\*GC.mem || i>0.4\*GC.mem ;){

i = Math.round(Math.random()\*0.4\*GC.mem);

}

return i;

}

function getRandomCpu(){ // for getting random cpu value for every process.

var i = Math.round(Math.random()\*10)+1;

return i;

}

GC.stepper = function(){ // this is our stepper function. Everytime you press the "Stepper" button, this works

if(this.mem>=this.P[this.stepcount].mem+this.usedmem){ // if memory is sufficient for next process, do this.

var divcode = "<div id='P"+this.stepcount+"' class='process'>P"+this.stepcount+"</div>"

$("#memory").append(divcode);

$("#P"+this.stepcount).css("background-color",this.P[this.stepcount].color);

$("#P"+this.stepcount).css("width",this.P[this.stepcount].mem/GC.mem\*1000);

this.arrayP.push(this.stepcount); // whenever a process enters to the memory, we push the process label into that array.

this.usedmem += this.P[this.stepcount].mem; // used memory increases in every insertion.

this.stepcount++;

}

else{

var mincpu = 11; //our cpu values are randomed between 1 and 10. as a worst case scenario, this is the max one.

var log = "<p>Garbage Collection Logs:</p>";

for(var i=0;i<this.arrayP.length;i++){ //gets the minimum cpu time value between processes in memory.

if (mincpu>this.P[this.arrayP[i]].cpu) mincpu = this.P[this.arrayP[i]].cpu;

}

this.totalcpu += mincpu; //adds it to total cpu.

$("#cputime").html(this.totalcpu);

for(var i=0;i<this.arrayP.length;i++){ //subtracts minimum cpu value from every processes` cpu time in memory.

this.P[this.arrayP[i]].cpu -= mincpu;

}

for(var i=0;i<this.arrayP.length;i++){ //removes processes that has cpu time as 0 or lesser.

if (this.P[this.arrayP[i]].cpu<=0){

this.usedmem -= this.P[this.arrayP[i]].mem; //removes memory values from main memory

log += "<p>Process "+this.arrayP[i]+" is removed."

$("#P"+this.arrayP[i]).remove(); //removes the block that shows removed process

this.arrayP.remove(this.arrayP.indexOf(this.arrayP[i])); //also removed from array.

}

}

}

$("#gclog").html(log);

$("#ams").html(this.usedmem);

$("#fms").html(this.mem-this.usedmem);

}

Array.prototype.remove = function(from, to) { //In JS, there is no built-in array removal function, so i wrote one.

var rest = this.slice((to || from) + 1 || this.length);

this.length = from < 0 ? this.length + from : from;

return this.push.apply(this, rest);

};

Jquery-v2.0.3.js: framework for Javascript (<http://jquery.com/>)

Flowchart:

