

OPERATING sYSTEM

Assignment one



Name : Ramy Mamdouh

ID : 118371

# Round Robin (***Preemptive*** scheduling):

* Time slices are assigned to each process in equal percentages and in Sequenced order, handling all processes without priority .Round-robin scheduling is simple, easy to implement, and starvation-free
* Each process gets a small unit of CPU time (time quantum)
* Preemptive scheduling: a switch of threads will occur when its time slice is used up
* Done by passing arguments to the threads controlling threads by variables

# FCFS (***Non-preemptive)***

* First-In-First-Out (FIFO)
* Run-Until-Done
* Processes are dispatched according to their arrival time on the ready queue(random: according to arrival time/no priorities )
* Once the CPU has a Thread, it runs to completion
* Done by locking Threads using objects, lock: The lock keyword ensures that one thread does not enter a critical section of code while another thread is in the critical section.

Code FCFS (with comments):

Threads Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.IO;

//class "Threads" to obtain all threads(5 functions) and their helping functions

namespace assignment\_one

{

class Threads

{

//helping function : array containing from 1->25 for the functions to access it

public static string arr(int number)

{

string[] arr = { "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "tweleve", "therteen", "fourteen", "fifteen", "sixteen", "seventeen", "eigteen", "ninteen", "twenty", "twenty one", "twenty two", "twenty three", "twenty four", "twenty five" };

return arr[number];

}

//writing into file

public void writeIntoFile(string name)

{

FileStream file = new FileStream("file.txt", FileMode.Append, FileAccess.Write);//file named "file.txt"

StreamWriter filewriter = new StreamWriter(file);

filewriter.WriteLine(name); //writing the name passed to the function in "file.txt"

filewriter.Close();

file.Close();

}

//reading all contents of the file

public void readFile()

{

FileStream fs = new FileStream("file.txt", FileMode.Open, FileAccess.Read);

StreamReader sr = new StreamReader(fs);

Console.WriteLine("Here is the content of the file:");

string str = sr.ReadLine();

while (str != null)

{

Console.WriteLine(str);

str = sr.ReadLine();

}

Console.ReadLine();

sr.Close();

fs.Close();

}

//function 1 to write from 1 to 5

public void function1()

{

/\*

The lock keyword ensures that one thread does not enter a critical section of code while another thread is in the critical section.

\* If another thread tries to enter a locked code, it will wait, block, until the object is released.

\* refernce : http://msdn.microsoft.com/en-us/library/c5kehkcz.aspx

\*/

lock ("file.txt")//object used here is the File.txt,any object can be used for this purpose

{

string threadname = "thread-one";

for (int i = 0; i < 5; i++)

{

string number = arr(i);//obtainig number from function arr

//writing into file the number and the thread name

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

//reflection of what is being added to the file

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

Thread.Sleep(500);//Thread.sleep main use here is for the user to see clearly the iteration by decreasing the speed of the iterations

}

}

}

//function 2 to write from 1 to 10

public void function2()

{

lock ("file.txt")

{

string threadname = "thread-two";

for (int i = 0; i < 10; i++)

{

string number = arr(i);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

Thread.Sleep(450);

}

}

}

//function 3 to write from 1 to 15

public void function3()

{

lock ("file.txt")

{

string threadname = "thread-three";

for (int i = 0; i < 15; i++)

{

string number = arr(i);

writeIntoFile(number + "----------------------------------------------------------" + threadname);

Console.WriteLine(number + "----------------------------------------------------------" + threadname);

Thread.Sleep(400);

}

}

}

//function 4 to write from 1 to 20

public void function4()

{

lock ("file.txt")

{

string threadname = "thread-four";

for (int i = 0; i < 20; i++)

{

string number = arr(i);

writeIntoFile(number + "----------------------------------------------------------" + threadname);

Console.WriteLine(number + "----------------------------------------------------------" + threadname);

Thread.Sleep(350);

}

}

}

//function 5 to write from 1 to 25

public void function5()

{

lock ("file.txt")

{

string threadname = "thread-five";

for (int i = 0; i < 25; i++)

{

string number = arr(i);

writeIntoFile(number + "-------------------------------------------------------" + threadname);

Console.WriteLine(number + "-------------------------------------------------------" + threadname);

Thread.Sleep(300);//decreasing the milliseconds as function 5 containing has the maximum number of all functions

}

}

}

}

}

Main method :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.IO;

namespace assignment\_one

{

class Program

{

static void Main(string[] args)

{

int choice;

Console.WriteLine("you will see what will happen in the file :");//reflection of what is done in the file

Threads T = new Threads(); //T object of class Threads

/\*

intalizing the functions as Threads

\*/

Thread t1 = new Thread(T.function1);

Thread t2 = new Thread(T.function2);

Thread t3 = new Thread(T.function3);

Thread t4 = new Thread(T.function4);

Thread t5 = new Thread(T.function5);

/\*

starting the threads

\*/

t1.Start();

t2.Start();

t3.Start();

t4.Start();

t5.Start();

Thread.Sleep(28500);//28500 is the sum of all iteration .

//Main thread here waits for all threads to finish and then print the following statement

Console.WriteLine("reading data from file press 1");

choice = int.Parse(Console.ReadLine());

if (choice == 1)

{

T.readFile();//reading file

}

else

{

Console.WriteLine("thank you for using the FcFS Scheduling method");

}

}

}

}

Code Round Robin (with comments):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.IO;

using System.Diagnostics;

namespace assignment\_one

{

class Program

{

//helping function : array containing from 1->25 for the functions to access it

public static string arr(int number)

{

string[] arr = { "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "tweleve", "therteen", "fourteen", "fifteen", "sixteen", "seventeen", "eigteen", "ninteen", "twenty", "twenty one", "twenty two", "twenty three", "twenty four", "twenty five" };

return arr[number];

}

//writing into file

public static void writeIntoFile(string name)

{

FileStream file = new FileStream("file.txt", FileMode.Append, FileAccess.Write);

StreamWriter filewriter = new StreamWriter(file);

filewriter.WriteLine(name);//writing the name passed to the function in "file.txt"

filewriter.Close();

file.Close();

}

//reading all contents of the file

public static void readFile()

{

FileStream fs = new FileStream("file.txt", FileMode.Open, FileAccess.Read);

StreamReader sr = new StreamReader(fs);

Console.WriteLine("Here is the content of the file:");

string str = sr.ReadLine();

while (str != null)

{

Console.WriteLine(str);

str = sr.ReadLine();

}

Console.ReadLine();

sr.Close();

fs.Close();

}

//function 1 to write from 1 to 5

//count is the number of array that must be printed

//int words = how many words gonna be printed in each slice(word slice)

//each thread has to print 2 words only and then go to another thread

static public void function1(int count,int words)

{

lock ("file.txt")

{

string threadname = "thread-one";

//for loop to print numbers according to the word slice (ex : 2 words then print 1 2/ex : 3 words then print 1 2 3)

for (int i = 0; i < words; i++)

{

if (count < 5)// as this function print from 1 to 5 if more than 5 is the input of count then the function won't respond

{

string number = arr(count);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

count++;//to print the next one

Thread.Sleep(200);

//Thread.sleep main use here is for the user to see clearly the iteration by decreasing the speed of the iterations

}

}

}

}

//function 2 to write from 1 to 10

static public void function2(int count, int words)

{

lock ("file.txt")

{

string threadname = "thread-two";

for (int i = 0; i < words; i++)

{

if (count < 10)

{

string number = arr(count);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

count++;

Thread.Sleep(200);

}

}

}

}

static public void function3(int count,int words)

{

lock ("file.txt")

{

string threadname = "thread-three";

for (int i = 0; i < words; i++)

{

if (count < 15)

{

string number = arr(count);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

count++;

Thread.Sleep(200);

}

}

}

}

static public void function4(int count, int words)

{

lock ("file.txt")

{

string threadname = "thread-four";

for (int i = 0; i < words; i++)

{

if (count < 20)

{

string number = arr(count);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

count++;

Thread.Sleep(200);

}

}

}

}

static public void function5(int count, int words)

{

lock ("file.txt")

{

string threadname = "thread-five";

for (int i = 0; i < words; i++)

{

if (count < 25)

{

string number = arr(count);

writeIntoFile(number + "--------------------------------------------------------------" + threadname);

Console.WriteLine(number + "--------------------------------------------------------------" + threadname);

count++;

Thread.Sleep(200);

}

}

}

}

static void Main(string[] args)

{

int choice;// choice for readfile

int words=2 ;//default value of word slice is 2

int i = 0; // i is declared seperataly as i increases according to how many words are added //dont need for loop

Console.WriteLine("you will see what will happen in the file :");

while(i<25) //i is the number of the array passed to the threads inorder to print it / words is the slice

{

Thread t1 = new Thread(() => function1(i,words));//method : for passing arguments to Threads

t1.Start();

Thread.Sleep(500);//main function waits for 0.5 seconds as threads may interfere with each others

Thread t2 = new Thread(() => function2(i,words));

t2.Start();

Thread.Sleep(500);//0.5 seconds between the start of each function

Thread t3 = new Thread(() => function3(i,words));

t3.Start();

Thread.Sleep(500);

Thread t4 = new Thread(() => function4(i,words));

t4.Start();

Thread.Sleep(500);

Thread t5 = new Thread(() => function5(i,words));

t5.Start();

Thread.Sleep(500);

for (int j = 0; j < words; j++)// to increase i by the slice quantity

{

i++;

}

}

//every time in the loop threads arguments are changed so creating and starting them is obligatory

//reading from the file

Console.WriteLine("reading data from file press 1");

choice =int.Parse(Console.ReadLine()) ;

if (choice == 1)

{

readFile();

}

else

{

Console.WriteLine("thank you for using the Round Robin Scheduling method");

}

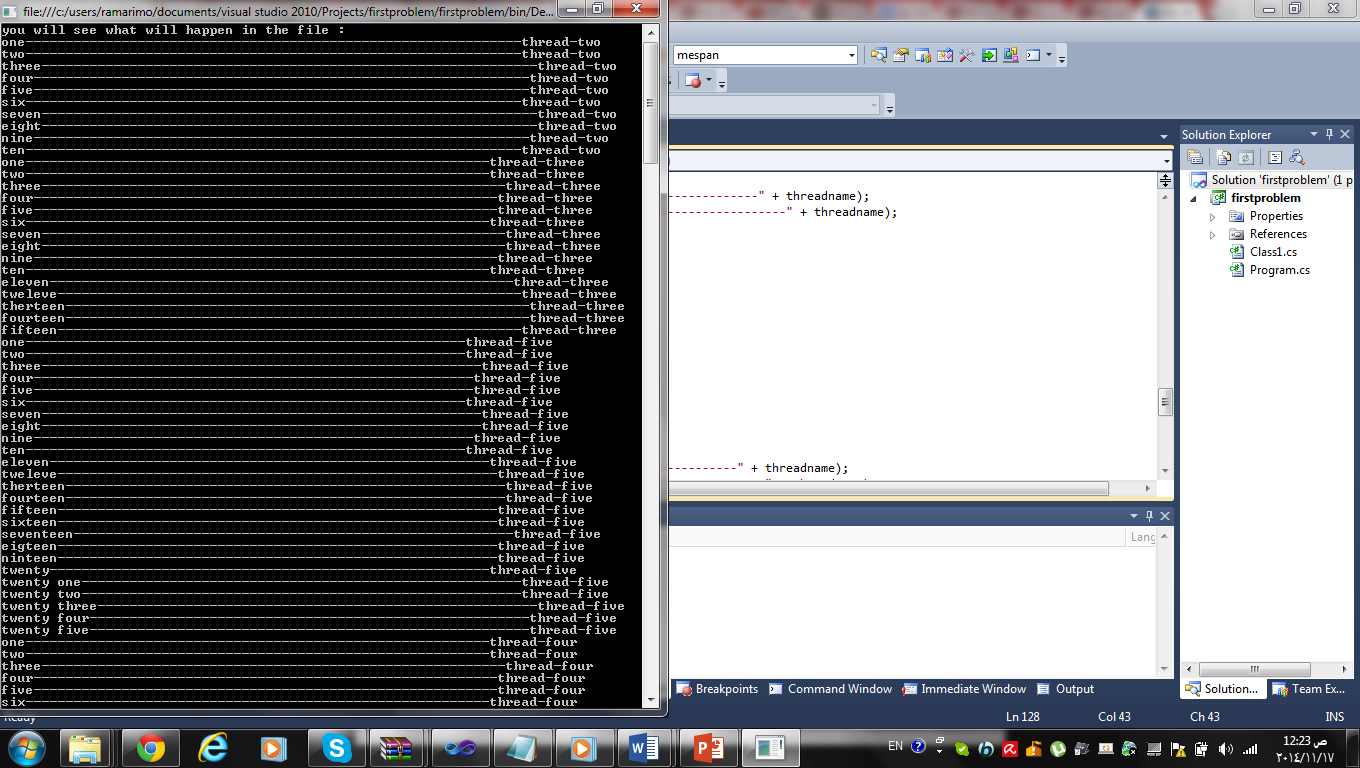
}

}

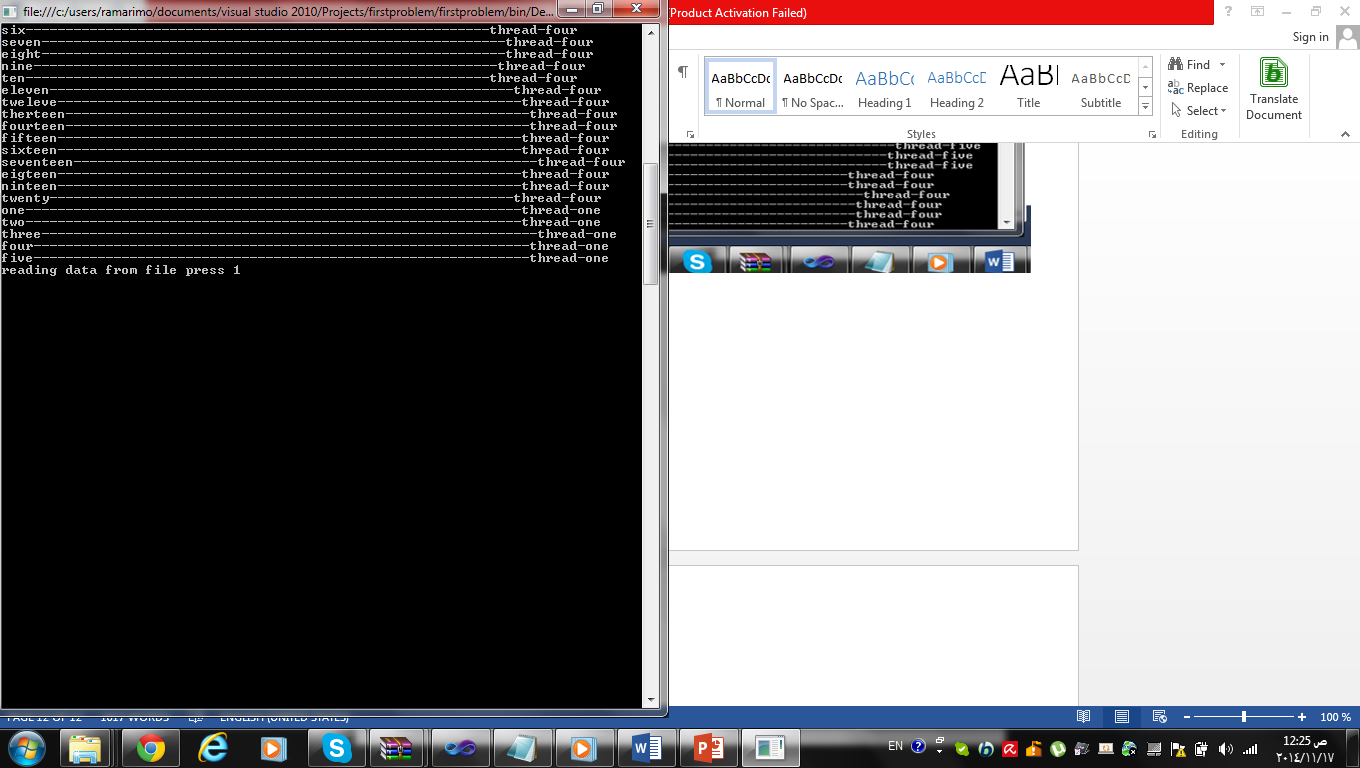
}

FCFS output: Console is reflecting of what is happening in the file

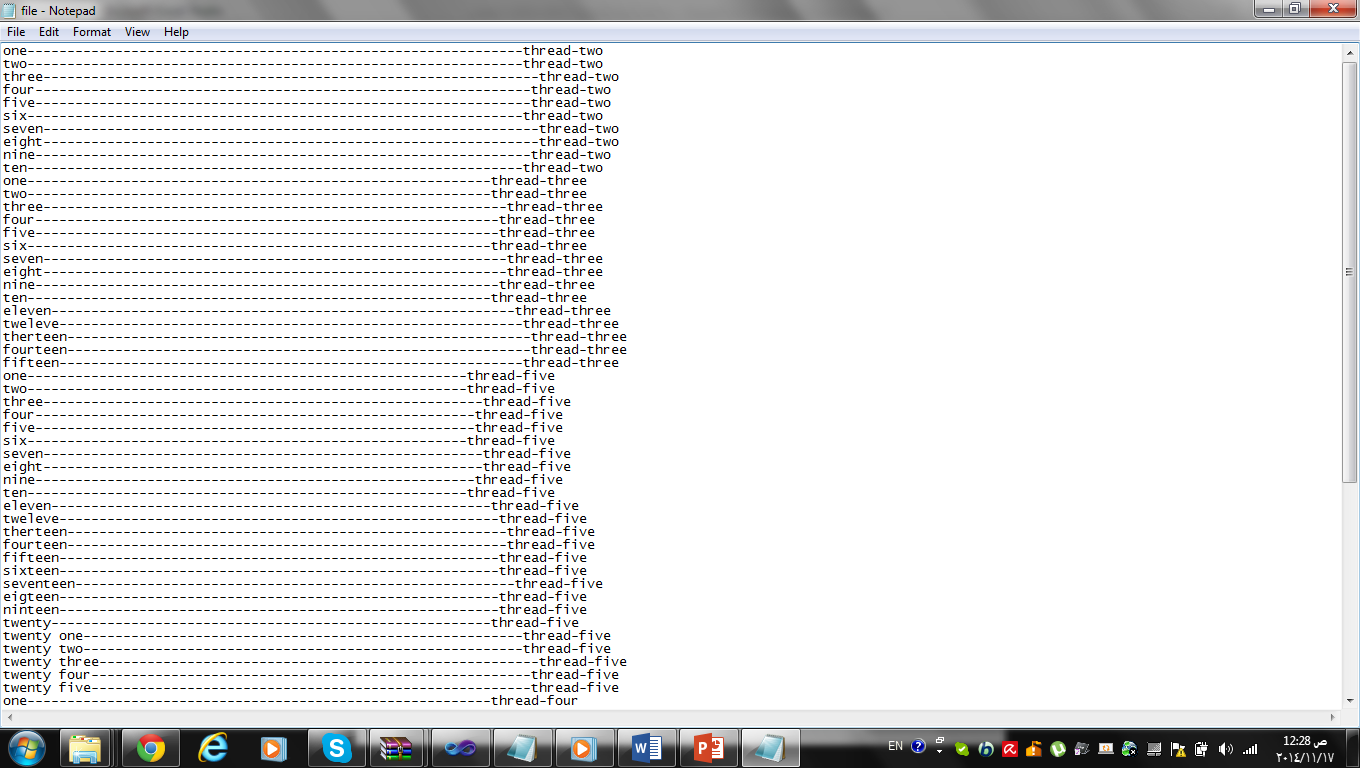
Output (1):

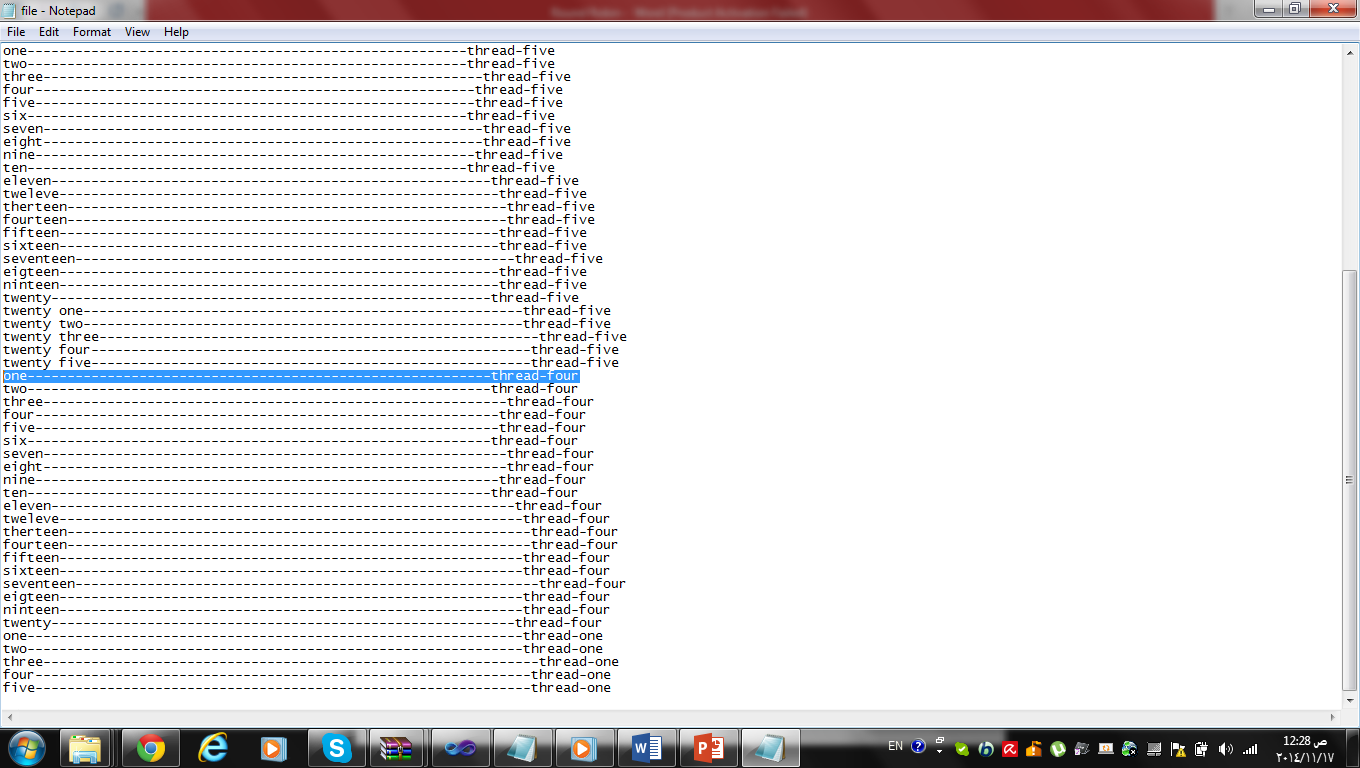


Output (2):



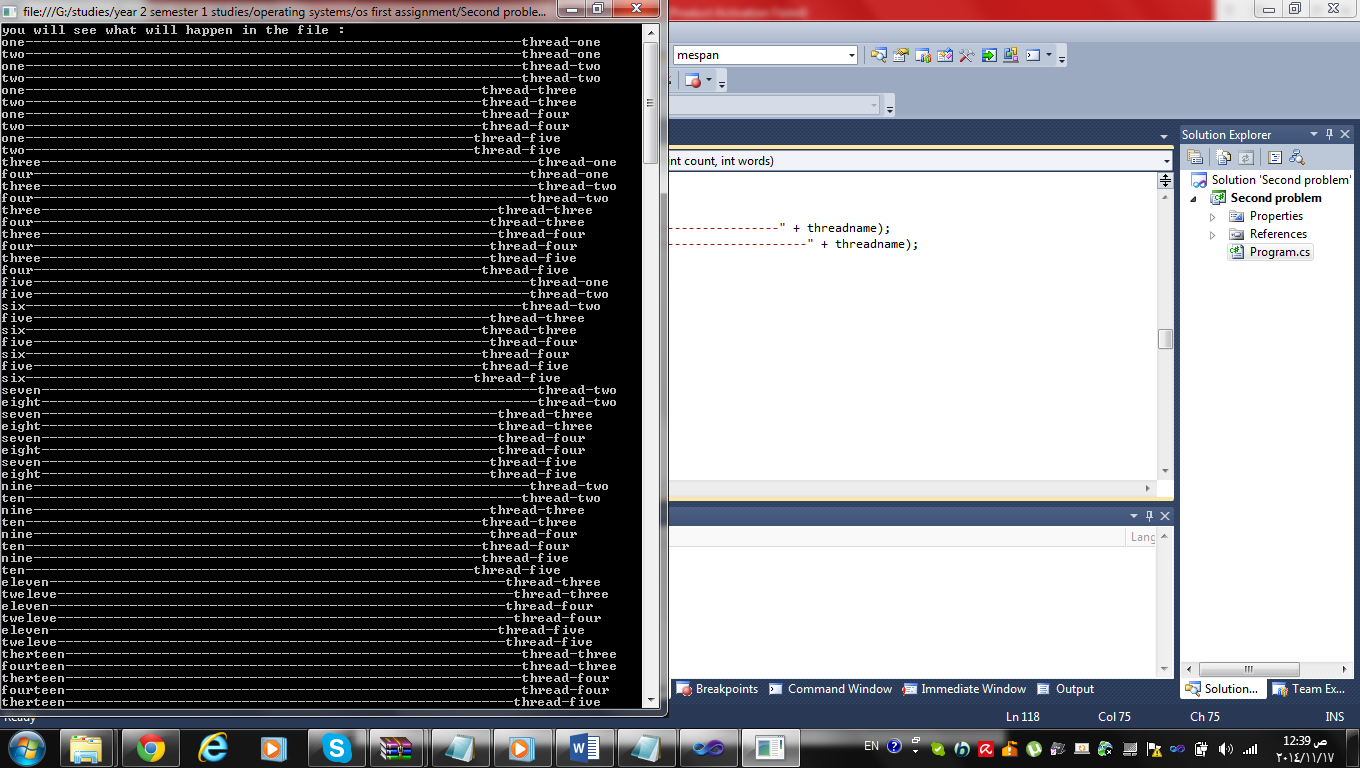
File.txt:



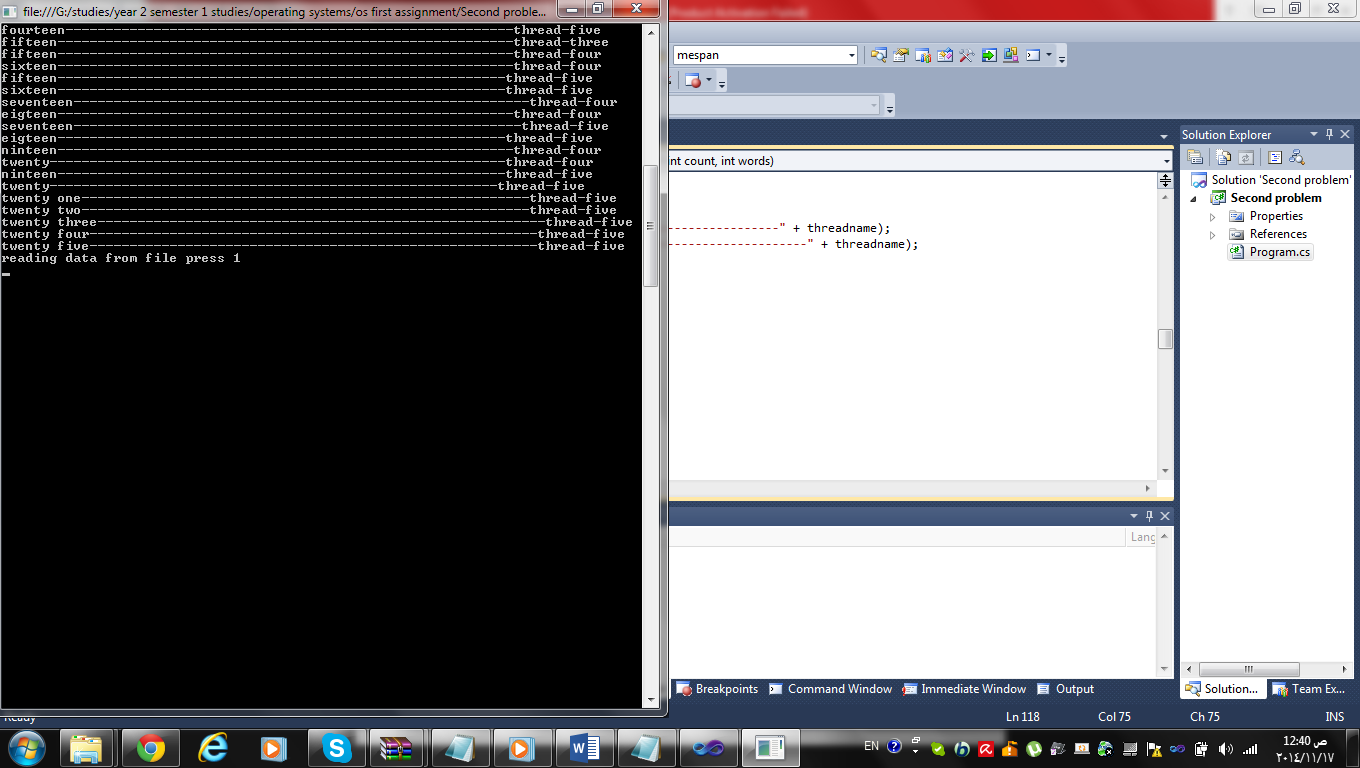


Round Robin output: Console is reflecting of what is happening in the file

Output (1):



Output (2):



File.Txt:

