

# Actividad: Cálculo distribuido de PI

Instituto Politécnico Nacional  
Escuela Superior de Computo  
Desarrollo de Sistemas Distribuidos  
Adrian González Pardo  
4CV1  
21/01

2 de octubre de 2020

## 1. Código fuente:

```
1  /*
2  * Archivo: PI.java
3  * @author Adrian Gonzalez Pardo
4  */
5  import java.net.Socket;
6  import java.net.ServerSocket;
7  import java.io.DataOutputStream;
8  import java.io.DataInputStream;
9  import java.lang.Thread;
10 import java.nio.ByteBuffer;
11
12 class PI{
13
14     static Object lock=new Object();
15     static double pi=0;
16     static class Worker extends Thread{
17         Socket conexion;
18         Worker(Socket conexion){
19             this.conexion=conexion;
20         }
21         public void run(){
22             /* Algoritmo 1 */
23             try{
24                 DataOutputStream salida=new DataOutputStream(conexion.getOutputStream());
25                 ;
26                 DataInputStream entrada=new DataInputStream(conexion.getInputStream());
27                 double x;
28                 x=entrada.readDouble();
29                 synchronized(lock){
30                     pi+=x;
31                 }
32                 salida.close();
33                 entrada.close();
34                 conexion.close();
35             }catch (Exception e) {
36                 System.err.println(e.getMessage());
37             }
38         }
39     };
40     public static void main(String[] args) throws Exception{
41         if (args.length!=1){
42             System.err.println("Uso:");
43             System.err.println("java PI <nodo>");
44         }
45     }
46 }
```

```

43     System.exit(0);
44 }
45 int nodo=Integer.valueOf(args[0]);
46 if(nodo==0){
47     /* Algoritmo 2 */
48     ServerSocket servidor=new ServerSocket(50000);
49     Worker w[]=new Worker[3];
50     int i=0;
51     while(i<3){
52         Socket conexion=servidor.accept();
53         w[i]=new Worker(conexion);
54         w[i].start();
55         i++;
56     }
57     double suma=0;
58     i=0;
59     while(i<10000000){
60         suma+=4.0/(8*i+1);
61         i++;
62     }
63     synchronized(lock){
64         pi+=suma;
65     }
66     i=0;
67     while(i<3){
68         w[i].join();
69         i++;
70     }
71     System.out.println("Valor de la variable pi: "+pi);
72 }else{
73     /* Algoritmo 3 */
74     Socket conexion=null;
75     while(true){
76         try{
77             conexion=new Socket("localhost",50000);
78             break;
79         }catch (Exception e){
80             Thread.sleep(100);
81         }
82     }
83     DataOutputStream salida=new DataOutputStream(conexion.getOutputStream());
84     DataInputStream entrada=new DataInputStream(conexion.getInputStream());
85     double suma=0;
86     int i=0;
87     while(i<10000000){
88         suma+=4.0/(8*i+(2*(nodo-1)+3));
89         i++;
90     }
91     suma=(nodo%2==0)?(suma):(-suma);
92     salida.writeDouble(suma);
93     salida.close();
94     entrada.close();
95     conexion.close();
96 }
97 }
98 }

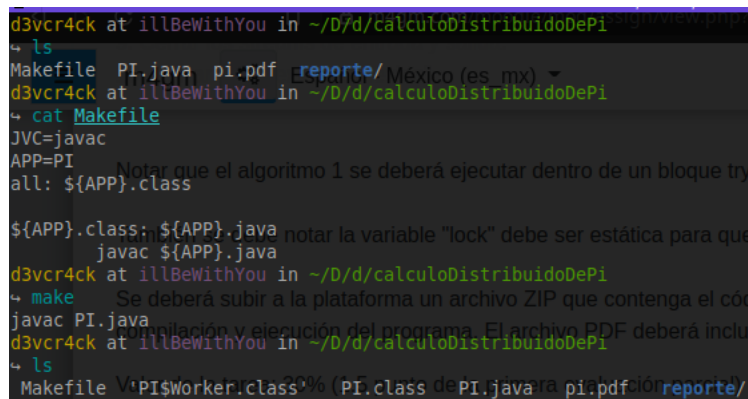
```

```

1 # Archivo Makefile
2 # @author Adrian Gonzalez Pardo
3 JVC=javac
4 APP=PI
5 all: ${APP}.class
6
7 ${APP}.class: ${APP}.java
8     javac ${APP}.java

```

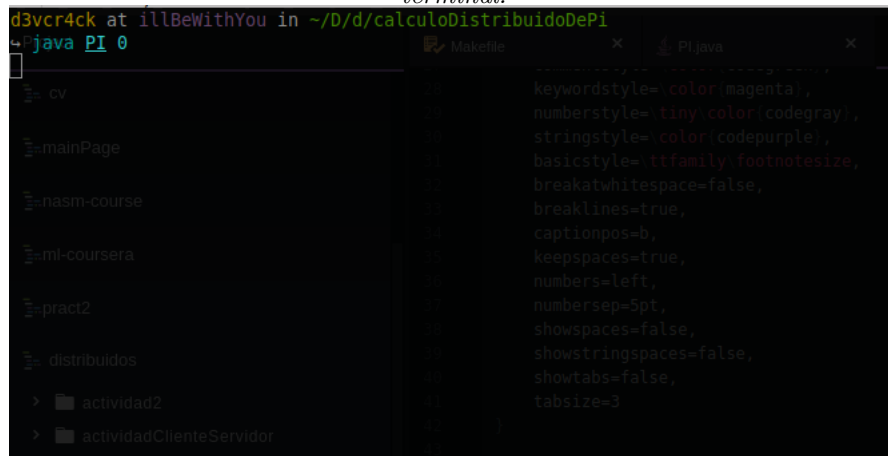
## 2. Capturas y descripción del programa



```
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ ls
Makefile  PI.java  pi.pdf  reporte/ México (es_mx)
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ cat Makefile
JVC=javac
APP=PI
all: ${APP}.class

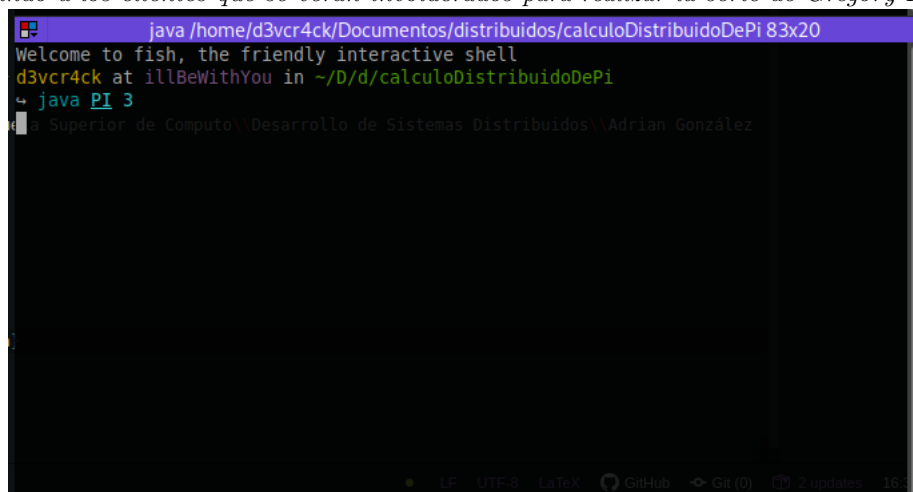
${APP}.class: ${APP}.java
javac ${APP}.java
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ make
javac PI.java
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ ls
Makefile  VPI$Worker.class  20% (PI.class de PI.java para pi.pdf)  reporte/
```

En esta captura podemos ver la compilación rápida del archivo `PI.java` gracias al archivo y las tareas que realiza el archivo `Makefile` y la sencillez de simplemente escribir `make` en la terminal.



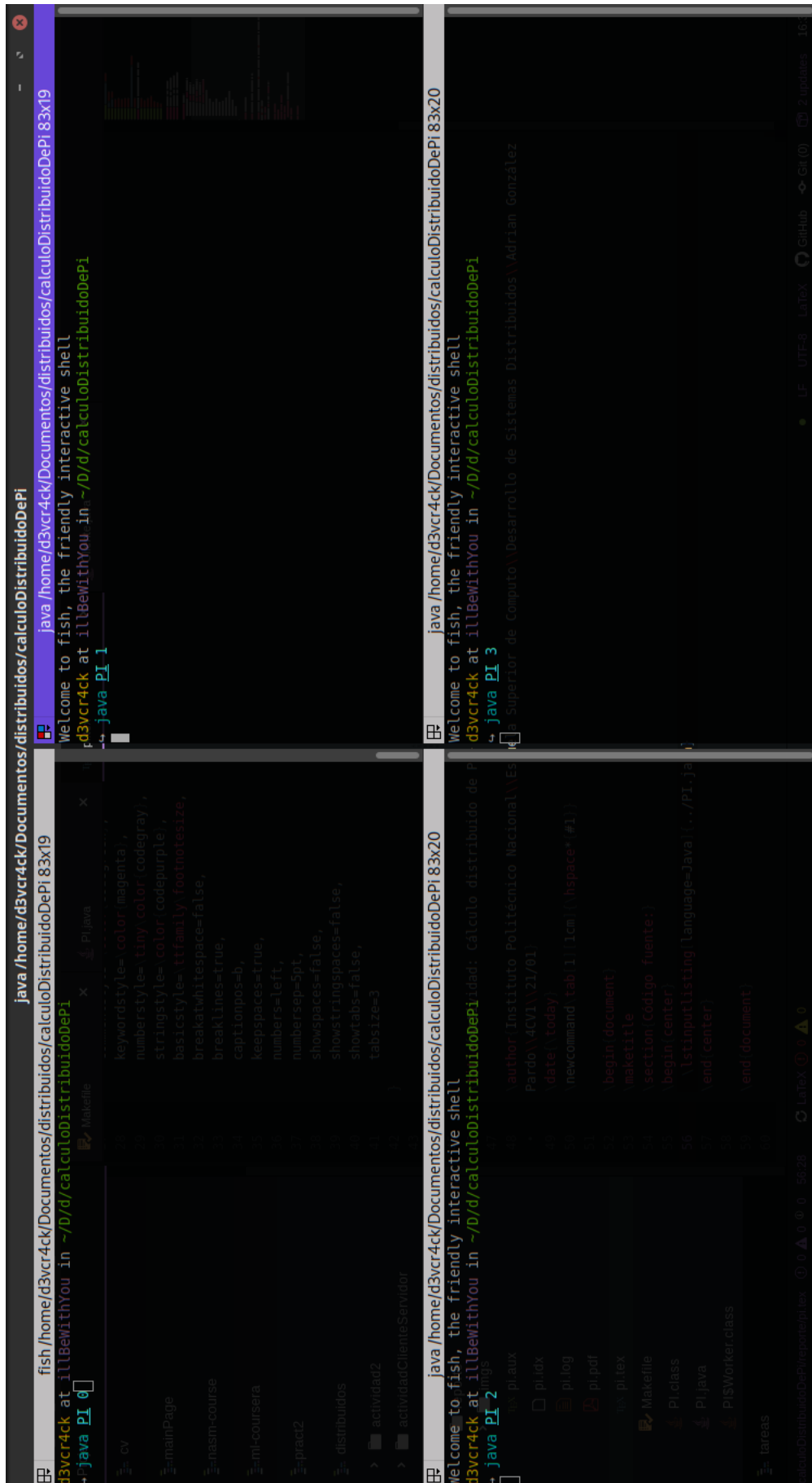
```
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ java PI 0
cv
mainPage
nasm-course
ml-coursera
pract2
distribuidos
  > actividad2
  > actividadClienteServidor
Makefile
PI.java
```

En esta captura podemos ver que cuando se ejecuta el nodo 0 (Servidor) este se queda esperando a los clientes que se verán involucrados para realizar la serie de Gregory-Leibniz.



```
java /home/d3vcr4ck/Documents/distribuidos/calculoDistribuidoDePi 83x20
Welcome to fish, the friendly interactive shell
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi [fish:~]$ java PI 3
Superior de Computo\Desarrollo de Sistemas Distribuidos\Adrian González
```

En esta captura podemos ver que cuando se ejecuta el nodo 3 (Cliente) el cual espera a que este disponible el Servidor (nodo 0), cabe destacar aquí que el server fue parado para mostrar esta funcionalidad.



*En esta captura podemos ver que los nodos 1-3 (Clientes) están esperando la ejecución del nodo 0 (Servidor) para poder realizar la serie del cálculo de  $\pi$*

```
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi
+ java PI 0
Valor de la variable pi: 3.141592628592157
d3vcr4ck at illBeWithYou in ~/D/d/calculoDistribuidoDePi
+ 
```

mainPage	0	numberstyle= tiny,color:codegray,
nasm-course	1	stringstyle= color:codepurple,
ml-coursera	2	basicstyle= ttfamily footnotesize,
pract2	3	breakatwhitespace=false,
distribuidos	4	breaklines=true,
actividad2	5	captionpos=b,
actividadClienteServidor	6	keepspaces=true,
	7	numbers=left,
	8	numbersep=5pt,
	9	showspaces=false,
	10	showstringspaces=false,
	11	showtabs=false,
	12	tabsize=3

*En esta captura podemos ver la ejecución de todos los nodos y de que el nodo 0 muestra el resultado obtenido de la ejecución y unión de los resultados de los nodos 1-3*