

# Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales

Sistemas Operativos



Profesora: Becerra Velázquez Violeta del Rocío

Alumno: Monjaraz Briseño Luis Fernando

Código: 218520958

Carrera: Ingeniería en Computación

Sección: D04

Actividad 12 (Programa 6, Algoritmo de planificación RR)

Fecha: 12/11/2023

**Índice**

<b><i>Índice</i></b>	<b>2</b>
<b><i>Tabla de imágenes</i></b>	<b>3</b>
<b><i>Datos personales</i></b>	<b>4</b>
<b><i>Datos de la materia</i></b>	<b>4</b>
<b><i>Número de actividad</i></b>	<b>4</b>
<b><i>Objetivo de la actividad</i></b>	<b>4</b>
<b><i>Notas acerca del lenguaje</i></b>	<b>4</b>
<b><i>Conclusión</i></b>	<b>15</b>

**Tabla de imágenes**

<i>Ilustración 1 Struct</i>	5
<i>Ilustración 2 Para contar los procesos listos y nuevos.</i>	6
<i>Ilustración 3 Quantum</i>	7
<i>Ilustración 4 Solicitud Quantum</i>	7
<i>Ilustración 5 Impresión Quantum</i>	7
<i>Ilustración 6 Condición Quantum</i>	7
<i>Ilustración 7 Reorganización de listos.</i>	8
<i>Ilustración 8 ValidaNumerosEnteros</i>	8
<i>Ilustración 9 gotoxy</i>	8
<i>Ilustración 10 IWillHaveOrder</i>	9
<i>Ilustración 11 ThisIsOrder</i>	10
<i>Ilustración 12 datosLotes</i>	11
<i>Ilustración 13 imprimirdata</i>	12
<i>Ilustración 14 main</i>	12
<i>Ilustración 15 Vista solicitud</i>	13
<i>Ilustración 16 Vista ejecución</i>	13
<i>Ilustración 17 Vista ejecución 2</i>	14
<i>Ilustración 18 Vista tabla de procesos</i>	14

**Datos personales**

Nombre: Monjaraz Briseño Luis Fernando

Código: 218520958

Correo: [luis.monjaraz5209@alumnos.udg.mx](mailto:luis.monjaraz5209@alumnos.udg.mx)

**Datos de la materia**

Materia: Sistemas Operativos

Sección: D04

Horario: Martes, Jueves, Sábado. 11:00 a 12:55

NRC: 204880

Clave: IL366

**Número de actividad**

Programa 6. Algoritmo de planificación RR

**Objetivo de la actividad**

El objetivo de esta actividad es recrear el funcionamiento de un algoritmo de planificación Round Robin (RR). Este siguiendo para su procesamiento el “Diagrama de 5 Estados”, en este caso se manejan 5 procesos en listos y el resto estarán en nuevos hasta que el número de procesos disminuya. Sin embargo, al pasar una cierta cantidad de tiempo impuesta por el usuario pasara a ejecución el proceso siguiente y el proceso que estaba pasara hasta el ultimo lugar de los listos, esto como lo haría un RR. Además, cuenta con todas las teclas y funcionalidades con las que contaban los anteriores códigos de algoritmos de planificación. Esta actividad nos ayudara a comprender el funcionamiento de un RR, esto mientras observamos como es que el quantum afecta a nuestros procesos, siendo esto observable al pulsar la tecla B o simplemente viendo nuestro “carrusel” que no es otra cosa que se estén actualizando los procesos.

En resumen, esta actividad es sumamente útil para comprender la teoría del RR y el como funciona su Quantum al igual de como este afectara a las demás funcionalidades.

**Notas acerca del lenguaje**

Lenguaje usado: C++

Motivo: Principalmente es debido a que es una modificación en la funcionalidad del programa 4 FCFS continuación. Donde ahora se utiliza un Quantum dado por el usuario, por lo que me era mas sencillo reutilizar el código del programa 4 y agregarle el Quantum, en este caso debo de decir que es la actividad mas sencilla que hemos tenido, pues realmente lo que hice fue reutilizar el funcionamiento de interrumpir del programa 2 y adaptarlo al quantum. Por lo que debo de decir que no me arrepiento para nada de mi elección de utilizar este lenguaje.

Estructuras: Utilice un “Struct” llamado “Process” en esta se almacenan los datos que se utilizaran para los procesos, la estructura es una cola estática (limitada a 5), para que siempre

haya solamente 5 procesos en el estado de “Listo”, cabe resaltar que también los demás procesos están en Colas de 5, sin embargo, mediante un while este va mandándolos a la cola 0 que es la que se maneja en todo el código.

El código es de 1640 líneas de código, cuenta con varias líneas de código comentadas esto porque me sirven de guía o eran la primera versión que realice y no me funciono, el motivo por el que no las borro es porque me sirven de guía a la hora de ver la previsualización a lado derecho en visual studio.

```
37 struct Process {
38     string operation;
39     int number1;
40     int number2;
41     string result;
42     string result2;
43     int estimatedTime;
44     int programNumber;
45     int currentQueue;
46     int tiempotranscurrido; // no se aplica en todos los casos
47     int tiemporestante; // no se aplica en todos los casos
48
49     int tiempobloqueado;
50     int tiempobloqueado2; // esta siempre sera 8
51     int tiempollegada;
52     int tiempofinalizacion;
53     int tiemporetorno;
54     int tiemporespuesta;
55     int tiempoespera;
56     int tiemposervicio; // este sera igual al tiempotranscurrido o al estimatedTime
57 };
58
```

*Ilustración 1 Struct*

```
315 void imprimirdata() {
316     int procesosTotales = totalProcesses;
317     int contadordeprocesosnuevos = totalProcesses;
318     int contadordeprocesoslistos = 0;
319     int ffy = 7;
320     char pulsar = ' ';
321     int a = 1;
322     gotoxy(80, 13);
323     cout << "I = Interrumpir";
324     gotoxy(80, 15);
325     cout << "E = Error";
326     gotoxy(80, 17);
327     cout << "P = Pausar";
328     gotoxy(80, 19);
329     cout << "C = Continuar";
330     queue<Process> cpc = queues[0];
331     queue<Process> totalProcessesQueueCopia[0];
332     queue<Process> tpqc = totalProcessesQueueCopia[0];
333     while(!cpc.empty()){
334         Process temporal = cpc.front();
335         cpc.pop();
336         contadordeprocesoslistos = contadordeprocesoslistos + 1;
337         contadordeprocesosnuevos = contadordeprocesosnuevos - 1;
338     }
```

*Ilustración 2 Para contar los procesos listos y nuevos.*

#### Funciones:

- ValidaNumerosEnteros: Esta función valida que lo que el usuario ingrese sea un numero entero, esto mediante un char para facilitar su comparación.
- Gotoxy: Esta me ayuda a hacer el aspecto visual del programa.
- IWillHaveOrder: Es la función que grafica las líneas visuales de los datos.
- ThisIsOrder: Es la función que grafica las líneas visuales del programa.
- DatosLotes: Es la función encargada de solicitar y hacer los cálculos con los datos de los procesos, al igual que validarlos.
- ImprimirDatos: Es la función encargada de imprimir toda la información, por lo que también es la que imprime el tiempo.
- Main: Es la función principal que llama a las demás y realiza limpieza en la pantalla.

Para el Quantum se creó una variable global llamada Quantum y a esta se le creo una copia, esto para poder restablecer su valor en el momento que le corresponda. A continuación, los principales cambios. Cabe resaltar que por como esta estructurado mi código tengo un valor llamado “pulsar” el cual se utilizo para poder pasar a la impresión, sin embargo, no tiene que ver con que directamente se pulse una tecla, ya que esta variable cambia cuando se presiona una tecla o se cumple la condición del quantum.

```

37  int quantum;
38  int quantumcopia;

```

Ilustración 3 Quantum

```

207  cout << "Ingrese el quantum: ";
208  while(!ValidaNumerosEnteros(quantumc)) {
209      gotoxy(1,4);
210      cout << " ";
211      gotoxy(1,4);
212      cout << "Ingrese el quantum de nuevo: ";
213      cin >> quantumc;
214  }
215  quantum = atoi(quantumc);
216  quantumcopia = quantum;

```

Ilustración 4 Solicitud Quantum

```

478  gotoxy(95,9);
479  cout << "Tiempo Quantum";
480  gotoxy(95,11);
481  cout << " ";
482  gotoxy(95,11);
483  cout << quantum;
484  gotoxy(17,17);
485  cout << "Qm";
486  gotoxy(22,17);
487  cout << " ";
488  gotoxy(22,17);
489  cout << quantum;

```

Ilustración 5 Impresión Quantum

```

510  quantum = quantum - 1;
511  /* Sleep(1000); // Modi temporizador */
512  if(quantum == 0){
513      pulsar = 'q';
514      break;
515  }

```

Ilustración 6 Condición Quantum

```
894     else if (pulsar == 'q'){
895         // copia de queues[i]
896         queue<Process> tempQueue = queues[i]; // Copia temporal de la cola
897         queues[i].pop();
898         // Volver a formar
899         queues[i].push(tempQueue.front());
900         quantum = quantumcopia;
901     }
```

Ilustración 7 Reorganización de listos.

Importante: En cuanto habrá el programa (importante que sea desde el .exe) dele al botón de “Maximizar pantalla”, o sea, el cuadradito que está en medio de minimizar y cerrar, esto porque se emplea Gotoxy y este ocasiona problemas si la pantalla no es lo suficientemente grande, de igual forma el Gotoxy está adaptado a mi pantalla (14 pulgadas) por lo que, si su pantalla es menor no se verá bien, si esta es mayor si se verá bien. De todas formas, implemente unas líneas de código que vuelven la pestaña más grande de lo normal. Nota: La cantidad de librerías es porque son de “colección” de librerías, por lo que no se usaron todas.

Funciones:

```
62 bool ValidaNumerosEnteros(char *dato){
63     bool ban = true;
64     int i = 0;
65     if (*dato == '-' || *dato == '+') {
66         i++;
67     }
68     while (*(dato + i) != '\0') {
69         if (*(dato + i) < '0' || *(dato + i) > '9') {
70             ban = false;
71             break;
72         }
73         i++;
74     }
75     return ban;
76 }
```

Ilustración 8 ValidaNumerosEnteros

```
78 void gotoxy(int x,int y){
79     HANDLE hcon;
80     hcon = GetStdHandle(STD_OUTPUT_HANDLE);
81     COORD dwPos;
82     dwPos.X = x;
83     dwPos.Y= y;
84     SetConsoleCursorPosition(hcon,dwPos);
85 }
86
```

Ilustración 9 gotoxy



```

96 void IWillHaveOrder(){
97     int x = 1, y = 1;
98     gotoxy(0,0);
99     printf("%c", 201); // 
100    gotoxy(132,0);
101    printf("%c", 187); // 
102    gotoxy(0,31);
103    printf("%c", 200); // 
104    gotoxy(132,31);
105    printf("%c", 188); // 
106    while (y<=30)
107    {
108        gotoxy(0,y);
109        printf("%c", 186); // 
110        gotoxy(132,y);
111        printf("%c", 186); // 
112        y++;
113    }
114    while (x<=131)
115    {
116        gotoxy(x,0);
117        printf("%c", 205); // 
118        gotoxy(x,31);
119        printf("%c", 205); // 
120        x++;
121    }
122    y = 1;
123    int y2 = 3;
124    gotoxy(16,2);
125    printf("%c", 203); // 
126    gotoxy(16,31);
127    printf("%c", 202); // 
128    gotoxy(33,0);

```

Ilustración 10 IWillHaveOrder

```
164 void ThisIsOrder(){
165     int x = 1, y = 1;
166     gotoxy(0,0);
167     printf("%c", 201); // 
168     gotoxy(132,0);
169     printf("%c", 187); // 
170     gotoxy(0,31);
171     printf("%c", 200); // 
172     gotoxy(132,31);
173     printf("%c", 188); // 
174     while (y<=30)
175     {
176         gotoxy(0,y);
177         printf("%c", 186); // 
178         gotoxy(132,y);
179         printf("%c", 186); // 
180         y++;
181     }
182     while (x<=131)
183     {
184         gotoxy(x,0);
185         printf("%c", 205); // 
186         gotoxy(x,31);
187         printf("%c", 205); // 
188         x++;
189     }
190 }
```

*Ilustración 11 ThisIsOrder*

```

192 void datosLotes(){
193     char totalProcessesc[100];
194     char quantumc[100];
195     gotoxy(1,1);
196     cout << "Ingrese el numero de procesos: ";
197     cin >> totalProcessesc;
198     while(!ValidaNumerosEnteros(totalProcessesc)){
199         gotoxy(1,1);
200         cout << " ";
201         gotoxy(1,1);
202         cout << "Ingrese el numero de procesos de nuevo: ";
203         cin >> totalProcessesc;
204     }
205     totalProcesses = atoi(totalProcessesc);
206     gotoxy(1,4);
207     cout << "Ingrese el quantum: ";
208     while(!ValidaNumerosEnteros(quantumc)){
209         gotoxy(1,4);
210         cout << " ";
211         gotoxy(1,4);
212         cout << "Ingrese el quantum de nuevo: ";
213         cin >> quantumc;
214     }
215     quantum = atoi(quantumc);
216     quantumcopia = quantum;
217     int currentQueue = 0;
218     for (int i = 1; i <= totalProcesses; ++i) {
219         Process newProcess;
220         newProcess.currentQueue = currentQueue+1;
221         int operationIndex = rand() % 6;
222         switch (operationIndex)
223         {
224             case 0:
225                 newProcess.operation = "+";

```

Ilustración 12 datosLotes

## Algoritmo de planificación RR

```
328 void imprimirdata() {
329     int procesosTotales = totalProcesses;
330     int contadordeprocesosnuevos = totalProcesses;
331     int contadordeprocesoslistos = 0;
332     int ffy = 7;
333     char pulsar = ' ';
334     int a = 1;
335     gotoxy(80, 13);
336     cout << "I = Interrumpir";
337     gotoxy(80, 15);
338     cout << "E = Error";
339     gotoxy(80, 17);
340     cout << "P = Pausar";
341     gotoxy(80, 19);
342     cout << "C = Continuar";
343     queue<Process> cpc = queues[0];
344     queue<Process> totalProcessesQueueCopia[0];
345     queue<Process> tpqc = totalProcessesQueueCopia[0];
346     while(!cpc.empty()){
347         Process temporal = cpc.front();
348         cpc.pop();
349         contadordeprocesoslistos = contadordeprocesoslistos + 1;
350         contadordeprocesosnuevos = contadordeprocesosnuevos - 1;
351     }
352     queue<Process> totalProcessesQueue[1]; // En esta cola se almacenaran todos los datos finalizados
353     queue<Process> totalProcessesQueueVacía[1];
354     queue<Process> tpqv = totalProcessesQueueVacía[1];
355     for (int i = 0; i < maxQueues; i++) {
356         gotoxy(1,1);
357         cout << "Procesos Nuevos: " << contadordeprocesosnuevos; // Modi
358         int acomodainterumpir = 11;
359         int ay = 7;
360         int y = 7;
361         if (!queues[i].empty()) {
362             gotoxy(3,3);
```

Ilustración 13 imprimirdata

```
1624 int main() {
1625     HWND consoleWindow = GetConsoleWindow();
1626     RECT desktop;
1627     GetWindowRect(GetDesktopWindow(), &desktop);
1628     MoveWindow(consoleWindow, desktop.left, desktop.top, desktop.right, desktop.bottom, TRUE);
1629     system("pause");
1630     ThisIsOrder();
1631     datoslotes();
1632     system("cls");
1633     IWillHaveOrder();
1634     imprimirdata();
1635     gotoxy(80,30);
1636     system("pause");
1637     system("cls");
1638     return 0;
1639 }
1640
```

Ilustración 14 main

Como se debería de ver:



Ilustración 15 Vista solicitud

Procesos Nuevos: 1														
P Listos #5			Ejec											
ID	TME	TT	Nn											
2	10	0	ID	1						Procesos restantes				
3	6	0	Ope	porcentaje						6				
4	16	0	TME	12						Tiempo total    Tiempo Quantum				
5	8	0	TT	2						2                    2				
Bloqueados			TR	10						I = Interrumpir				
ID	TB		Qm	2_						E = Error				
										P = Pausar				
										C = Continuar				

Ilustración 16 Vista ejecución

## Algoritmo de planificación RR

Procesos Nuevos: 1					Terminados			Procesos restantes	
P Listos #5	Ejec				ID Ope	Res	Tanda Guía	Tiempo total	Tiempo Quantum
ID TME TT	Nn								
5 8 0	ID 4								6
1 12 4	Ope -								
2 10 4									
3 6 4	TME 16								14 2
	TT 2								
Bloqueados	TR 14								I = Interrumpir
ID TB	Qm 2								E = Error
									P = Pausar
									C = Continuar

Ilustración 17 Vista ejecución 2

Procesos Activos										Procesos Bloqueados						
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	TT	ID	TL	TB	TE	TRa	TME	TT
1	0	NA	NA	1	22	6	12	NA	6							
2	0	NA	NA	4	22	4	10	NA	4							
3	0	NA	NA	8	22	4	6	NA	4							
4	0	NA	NA	12	22	4	16	NA	4							
5	0	NA	NA	16	22	4	8	NA	4							
6	0	NA	NA	0	22	0	16	NA	0							

  

Procesos Terminados																
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	Resultado	TT						

Ilustración 18 Vista tabla de procesos

Enlace de descarga (contenido):

<https://drive.google.com/drive/folders/1Sa592otNQ8zw4UFRBbO1ooMvObuPVm6G?usp=sharing>

### ***Conclusión***

En conclusión, esta actividad nos ayuda a reafirmar los conocimientos sobre el algoritmo RR, en lo personal esta actividad a sido la mas sencilla hasta el momento, pues me tomo tan solo 30 minutos en realizar, puede ser que esto sea debido a que desde que nos la menciono estuve 1 semana pensando el cómo la realizaría. Ahora sobre la actividad, la utilización del quantum me forzó a reestructurar ligeramente el como manejaba los tiempos, pues antes usaba el sleep al final de la ejecución, sin embargo, tuve que moverla al inició para que se ejecutara de la manera correcta. Es sumamente interesante el ver como reacciona el programa con la implementación del Quantum, pues algunos procesos terminan mucho antes que los demás, al igual del como le afectan las demás teclas. En resumen, esta actividad se me hizo bastante útil para complementar el tema de los algoritmos de planificación especialmente el RR, pues de eso se trataba este programa.