

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 9 Programa 4 FCFS continuación

Fecha: 22/10/2023

Índice

<i>Índice</i>	2
<i>Tabla de imágenes</i>	3
<i>Datos personales</i>	4
<i>Datos de la materia</i>	4
<i>Número de actividad</i>	4
<i>Objetivo de la actividad</i>	4
<i>Notas acerca del lenguaje</i>	4
<i>Conclusión</i>	18

Tabla de imágenes

<i>Ilustración 1 Struct</i>	5
<i>Ilustración 2 Para contar los procesos listos y nuevos.</i>	5
<i>Ilustración 3 Nuevos procesos.</i>	6
<i>Ilustración 4 Nuevos procesos 2.</i>	7
<i>Ilustración 5 Mostrar tabla de tiempos.</i>	7
<i>Ilustración 6 Mostrar tabla de tiempos 2.</i>	8
<i>Ilustración 7 Mostrar tabla de tiempos 3.</i>	9
<i>Ilustración 8 Mostrar tabla de tiempos 4.</i>	10
<i>Ilustración 9 ValidaNumerosEnteros</i>	11
<i>Ilustración 10 Gotoxy</i>	11
<i>Ilustración 11 IWillHaveOrder</i>	12
<i>Ilustración 12 ThisIsOrder</i>	13
<i>Ilustración 13 datosLotes</i>	14
<i>Ilustración 14 imprimirdata</i>	15
<i>Ilustración 15 main</i>	15
<i>Ilustración 16 Como se deberia de ver Inicio</i>	16
<i>Ilustración 17 Tabla de los tiempos</i>	16
<i>Ilustración 18 Proceso nuevo</i>	17
<i>Ilustración 19 Tabla de los tiempos 2</i>	17
<i>Ilustración 20 Tabla de los tiempos finalización.</i>	18

Datos personales

Nombre: Monjaraz Briseño Luis Fernando

Código: 218520958

Correo: 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 4. Algoritmo de planificación FCFS (First Come First Server) Continuación.

Objetivo de la actividad

El objetivo de esta actividad es recrear el funcionamiento de un algoritmo de planificación FCFS. Este mismo para el procesamiento se basa en 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, ahora además cuenta con dos nuevas teclas, siendo “N” de nuevo, el cual al ser pulsado generara un nuevo proceso y “B” la cual muestra la tabla de los tiempos. Esta actividad ayudara a comprender el funcionamiento de un FCFS, esto mientras observamos como las diversas opciones que implementamos se llevan a cabo, siendo el proceso nuevo para corroborar que los tiempos se estén asignando correctamente y la tecla b para poder ver en tiempo real los tiempos, esto para corroborar su funcionamiento.

En resumen, esta actividad es sumamente útil para comprender la teoría del FCFS y el manejo de los Tiempos.

Notas acerca del lenguaje

Lenguaje usado: C++

Motivo: Principalmente es debido a que este código es una continuación directa del programa 3, por lo que implementar los cambios era mas sencillo modificar el código anterior que crear uno nuevo, aunque debo de decir que cada día me arrepiento mas de no haberlo realizado en Python. Cabe resaltar que para las impresiones utilice gotoxy, debo de decir que en lo personal el mostrar la tabla de los tiempos se me hizo extremadamente complicado.

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 1581 líneas de código, cuenta con varias líneas de código comentadas esto por que me sirven de guía o eran la primera versión que realice y no me funciono, el motivo por el que no las borro es por que 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 contadorProcesosNuevos = totalProcesses;
318     int contadorProcesosListos = 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         contadorProcesosListos = contadorProcesosListos + 1;
337         contadorProcesosNuevos = contadorProcesosNuevos - 1;
338     }

```

Ilustración 2 Para contar los procesos listos y nuevos.

Algoritmo de planificación FCFS (First Come First Server) Continuación.

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.

```
} else if(key == 'N' || key == 'n') {  
    pulsar = 'n';  
    break;  
}
```

Ilustración 3 Nuevos procesos.

```

1278     else if(pulsar == 'n'){
1279         int currentQueue = 0;
1280         Process newProcess;
1281         newProcess.currentQueue = currentQueue+1;
1282         int operationIndex = rand() % 6;
1283         switch (operationIndex)
1284         {
1285             case 0:
1286                 newProcess.operation = "+";
1287                 break;
1288             case 1:
1289                 newProcess.operation = "-";
1290                 break;
1291             case 2:
1292                 newProcess.operation = "*";
1293                 break;
1294             case 3:
1295                 newProcess.operation = "/";
1296                 break;
1297             case 4:
1298                 newProcess.operation = "residuo";
1299                 break;
1300             case 5:
1301                 newProcess.operation = "porcentaje";
1302                 break;
1303             default:
1304                 break;
1305         }
1306         int operationN1 = rand() % 101;
1307         int operationN2 = rand() % 101;
1308         newProcess.number1 = operationN1;
1309         newProcess.number2 = operationN2;
1310         if(newProcess.operation == "/"){
1311             while(newProcess.number2 == 0){
1312                 int operationN2 = rand() % 101;
1313                 newProcess.number2 = operationN2;
1314             }
1315         }
1316         if(newProcess.operation == "+"){

```

Ilustración 4 Nuevos procesos 2.

```

582     } else if(key == 'B' || key == 'b') {
583         pulsar = 'b';
584         break;

```

Ilustración 5 Mostrar tabla de tiempos.

```
936     else if (pulsar == 'b'){
937
938         system("cls");
939         /*
940         int i = 0;
941         while(!queues[i].empty()){
942             queues[i].pop();
943             // Imprime el tme de queues[i]
944             cout << queues[i].front().tiemposervicio << endl;
945         }
946         system("pause"); */
947         ThisIsOrder();
948         // imprimir datos finales
949         gotoxy(3,1);
950         cout << "Procesos Activos";
951         gotoxy(3,2);
952         cout << "ID"; // ID
953         gotoxy(7,2);
954         cout << "TL"; // tiempo llegada
955         gotoxy(11,2);
956         cout << "TF"; // tiempo finalizacion
957         gotoxy(15,2);
958         cout << "TR"; // tiempo retorno
959         gotoxy(19,2);
960         cout << "TRa"; // tiempo respuesta
961         gotoxy(25,2);
962         cout << "TE"; // tiempo espera
963         gotoxy(31,2);
964         cout << "TS"; // tiempo servicio
965         gotoxy(37,2);
966         cout << "TME"; // tiempo estimado
967         gotoxy(43,2);
968         cout << "Estado"; // Tipo de finalización (Normal o Error)
969         gotoxy(58,2);
970         cout << "Operacion";
971         ffy = 3;
972         // Impresión TDD
973         queue<Process> copiadebloqueados = bloqueados;
974         queue<Process> tddimpresionblanco = unifiedQueue;
```

Ilustración 6 Mostrar tabla de tiempos 2.


```

1006 for (int i = 0; i < maxQueues; i++)
1007 {
1008     queue<Process> salvacion = queues[i];
1009     while(!salvacion.empty()){
1010         Process salvacionproceso = salvacion.front();
1011         salvacion.pop();
1012         gotoxy(3,ffiy);
1013         cout << salvacionproceso.programNumber;
1014         gotoxy(7,ffiy);
1015         cout << salvacionproceso.tiempollegada;
1016         gotoxy(11,ffiy);
1017         cout << "NA";
1018         gotoxy(15,ffiy);
1019         cout << "NA";
1020         gotoxy(19,ffiy);
1021         cout << salvacionproceso.tiemporespuesta;
1022         gotoxy(25,ffiy);
1023         cout << timecontador - salvacionproceso.tiempollegada;
1024         gotoxy(31,ffiy);
1025         cout << salvacionproceso.tiemposervicio; // Al momento
1026         gotoxy(37,ffiy);
1027         cout << salvacionproceso.estimatedTime;
1028         gotoxy(43,ffiy);
1029         cout << "NA";
1030         gotoxy(58,ffiy);
1031         cout << salvacionproceso.number1 << salvacionproceso.operation << salvacionproceso.number2;
1032         ffy = ffy + 1;
1033     }
1034 }
1035 ffy = 18;
1036 queue<Process> p2 = totalProcessesQueue[0];
1037 gotoxy(3,16);
1038 cout << "Procesos Terminados";
1039 gotoxy(3,17);
1040 cout << "ID"; // ID
1041 gotoxy(7,17);
1042 cout << "TL"; // tiempo llegada
1043 gotoxy(11,17);
1044 cout << "TF"; // tiempo finalizacion
1045 gotoxy(15,17);

```

Ilustración 7 Mostrar tabla de tiempos 3.

```
1087     },
1088     ffy = 3;
1089     gotoxy(90,1);
1090     cout << "Procesos Bloqueados";
1091     gotoxy(90,2);
1092     cout << "ID";
1093     gotoxy(94,2);
1094     cout << "TB";
1095     gotoxy(98,2);
1096     cout << "TE";
1097     gotoxy(102,2);
1098     cout << "TRa";
1099     gotoxy(106,2);
1100     cout << "TME";
1101     while(!copiadebloqueados.empty()){
1102         Process bbb = copiadebloqueados.front();
1103         copiadebloqueados.pop();
1104         gotoxy(90,ffiy);
1105         cout << bbb.programNumber;
1106         gotoxy(94,ffiy);
1107         cout << bbb.tiempobloqueado;
1108         gotoxy(98,ffiy);
1109         cout << bbb.tiempoespera;
1110         gotoxy(102,ffiy);
1111         cout << bbb.tiemporespuesta;
1112         gotoxy(106,ffiy);
1113         cout << bbb.estimatedTime;
1114         ffy = ffy + 1;
1115     }
1116     ffy = 7;
1117 }
```

Ilustración 8 Mostrar tabla de tiempos 4.

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.

Como se puede apreciar la mayoría de los if cambian un valor de “pulsar” este es para que al momento de la impresión se seleccione la forma correcta dependiendo de la opción.

Funciones:

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

Ilustración 9 ValidaNumerosEnteros

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

Ilustración 10 Gotoxy

```
94  ✓ void IWillHaveOrder(){
95      int x = 1, y = 1;
96      gotoxy(0,0);
97      printf("%c", 201); // 
98      gotoxy(132,0);
99      printf("%c", 187); // 
100     gotoxy(0,31);
101     printf("%c", 200); // 
102     gotoxy(132,31);
103     printf("%c", 188); // 
104  ✓   while (y<=30)
105     {
106         gotoxy(0,y);
107         printf("%c", 186); // 
108         gotoxy(132,y);
109         printf("%c", 186); // 
110         y++;
111     }
112  ✓   while (x<=131)
113     {
114         gotoxy(x,0);
115         printf("%c", 205); // 
116         gotoxy(x,31);
117         printf("%c", 205); // 
118         x++;
119     }
```

Ilustración 11 IWillHaveOrder

```

161
162  void ThisIsOrder(){
163      int x = 1, y = 1;
164      gotoxy(0,0);
165      printf("%c", 201); // 
166      gotoxy(132,0);
167      printf("%c", 187); // 
168      gotoxy(0,31);
169      printf("%c", 200); // 
170      gotoxy(132,31);
171      printf("%c", 188); // 
172  while (y<=30)
173  {
174      gotoxy(0,y);
175      printf("%c", 186); // 
176      gotoxy(132,y);
177      printf("%c", 186); // 
178      y++;
179  }
180  while (x<=131)
181  {
182      gotoxy(x,0);
183      printf("%c", 205); // 
184      gotoxy(x,31);
185      printf("%c", 205); // 
186      x++;
187  }
188  }

```

Ilustración 12 ThisIsOrder

```
190 void datosLotes(){
191     char totalProcessesc[100];
192     gotoxy(1,1);
193     cout << "Ingrese el numero de procesos: ";
194     cin >> totalProcessesc;
195     while(!ValidaNumerosEnteros(totalProcessesc)){
196         gotoxy(1,1);
197         cout << "
198         gotoxy(1,1);
199         cout << "Ingrese el numero de procesos de nuevo: ";
200         cin >> totalProcessesc;
201     }
202     totalProcesses = atoi(totalProcessesc);
203
204     int currentQueue = 0;
205     for (int i = 1; i <= totalProcesses; i++){
206         Process new int Process::currentQueue
207         newProcess.currentQueue = currentQueue+1;
208         int operationIndex = rand() % 6;
209         switch (operationIndex)
210         {
211         case 0:
212             newProcess.operation = "+";
213             break;
214         case 1:
215             newProcess.operation = "-";
216             break;
217         case 2:
218             newProcess.operation = "*";
219             break;
220         case 3:
221             newProcess.operation = "/";
222             break;
223         case 4:
```

Ilustración 13 datosLotes

```

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     }
339     queue<Process> totalProcessesQueue[1]; // En esta cola se almacenaran todos los datos finalizados
340     queue<Process> totalProcessesQueueVacía[1];
341     queue<Process> tpqv = totalProcessesQueueVacía[1];
342     for (int i = 0; i < maxQueues; i++) {
343         gotoxy(1,1);
344         cout << "Procesos Nuevos: " << contadordeprocesosnuevos; // Modi
345         int acomodainterumpir = 11;
346         int ay = 7;
347         int y = 7;
348         if (!queues[i].empty()) {
349             gotoxy(3,3);
350             cout << "P Listos #" << contadordeprocesoslistos << endl; // Modi
351             queue<Process> tempQueue = queues[i]; // Copia temporal de la cola
352             queue<Process> bloqueados;

```

Ilustración 14 imprimirdata

```

1566 int main() {
1567     HWND consoleWindow = GetConsoleWindow();
1568     RECT desktop;
1569     GetWindowRect(GetDesktopWindow(), &desktop);
1570     MoveWindow(consoleWindow, desktop.left, desktop.top, desktop.right, desktop.bottom, TRUE);
1571     system("pause");
1572     ThisIsOrder();
1573     datosLotes();
1574     system("cls");
1575     IWillHaveOrder();
1576     imprimirdata();
1577     gotoxy(80,30);
1578     system("pause");
1579     system("cls");
1580     return 0;
1581 }
1582

```

Ilustración 15 main

Como se debería de ver:

Algoritmo de planificación FCFS (First Come First Server) Continuación.

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

Ilustración 16 Como se debería de ver Inicio

Procesos Activos										Procesos Bloqueados				
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	Operacion	ID	TB	TE	TRa	TME
2	0	NA	NA	12	13	1	10	NA	69porcentaje65					
3	0	NA	NA	0	13	0	6	NA	22residuo49					
Procesos Terminados														
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	Operacion	Resultado				
1	0	12	12	1	12	12	12	Normal	85porcentaje72	61				

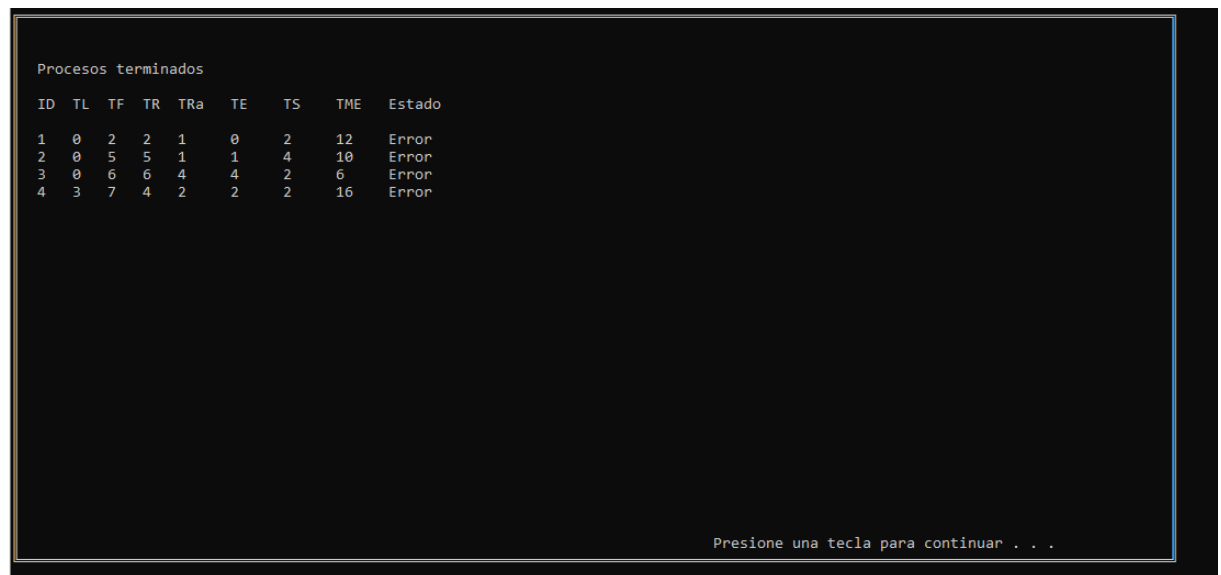
Ilustración 17 Tabla de los tiempos

Procesos Nuevos: 0			Terminados			Procesos restantes	
P Listos #3	Ejec		ID	Ope	Res	Tanda	Procesos restantes
ID TME TT	Nn		ID	Ope	Res	Guia	
3 6 0	ID 2						2
4 16 0	Ope porcentaje						Tiempo total
	TME 10						17
	TT 5						
Bloqueados	TR 5						
ID TB							

Ilustración 18 Proceso nuevo

Procesos Activos										Procesos Bloqueados				
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	Operacion	ID	TB	TE	TRa	TME
4	16	NA	NA	8	10	2	16	NA	61-63	3	7	22	22	6
Procesos Terminados										Procesos Bloqueados				
ID	TL	TF	TR	TRa	TE	TS	TME	Estado	Operacion	ID	TB	TE	TRa	TME
1	0	12	12	1	8	12	12	Normal	85porcentaje72	61				
2	0	22	22	12	8	10	10	Normal	69porcentaje65	44				

Ilustración 19 Tabla de los tiempos 2



Procesos terminados								
ID	TL	TF	TR	TRa	TE	TS	TME	Estado
1	0	2	2	1	0	2	12	Error
2	0	5	5	1	1	4	10	Error
3	0	6	6	4	4	2	6	Error
4	3	7	4	2	2	2	16	Error

Presione una tecla para continuar . . .

Ilustración 20 Tabla de los tiempos finalización.

Enlace de descarga (contenido):

<https://drive.google.com/drive/folders/1UnEIMU4-pDTsPGZ3HhORHTchSUHAyU5f?usp=sharing>

Conclusión

En conclusión, esta actividad nos ayuda a reafirmar los conocimientos de los algoritmos FCFS, esta actividad debo de decir que ha sido de las más difíciles que he realizado por la tabla de los tiempos, pues esta por como esa diseñado mi programa fue un dolor de cabeza muy grande, llego el punto en el que le tuve que pedir ayuda a un amigo el cual me dio alguna que otra idea para solucionarlo, lo cual desencadeno otros errores que si se pudieron corregir. Sobre la actividad el echo de poder visualizar los tiempos al pulsar la tecla b es realmente útil para observar el como trabajan los tiempos y las duraciones de los procesos en ejecución, por que a diferencia de la anterior actividad este si lo muestra en cualquier momento y no solamente al final. En resumen, esta actividad es bastante útil como complemento de la anterior. Sin embargo, debo decir que se me hace bastante repetitiva la actividad, pues prácticamente todos los programas hasta el momento son el mismo en esencia.