

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 3

Fecha: 31/08/2023

Índice

Índice	2
Tabla de imágenes	3
Datos personales	4
Datos de la materia	4
Número de actividad	4
Objetivo de la actividad	4
Notas acerca del lenguaje	4
Conclusión	10

Tabla de imágenes

<i>Ilustración 1 Estructura</i>	5
<i>Ilustración 2 ValidaNumerosEnteros</i>	5
<i>Ilustración 3 gotoxy</i>	6
<i>Ilustración 4 IWillHaveOrder</i>	6
<i>Ilustración 5 ThisIsOrder</i>	7
<i>Ilustración 6 datoslotes</i>	7
<i>Ilustración 7 imprimirdata</i>	8
<i>Ilustración 8 main</i>	8
<i>Ilustración 9 cola/fila</i>	9
<i>Ilustración 10 Resultado 2</i>	10
<i>Ilustración 11 Resultado</i>	10

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 1 Simular procesamiento por lotes.

Objetivo de la actividad

El objetivo de la actividad es la recrear el funcionamiento de un procesamiento por lotes, el cual consta de ejecutar un lote el cual contiene una cantidad de procesos, dichos procesos solo se podrán realizar en orden, o sea, el primero termina primero y el ultimo termina último, al limitar la ejecución de los procesos a un orden específico, se refleja cómo funcionaban los sistemas en esa época y cómo la automatización de tareas en lotes permitía mantener un flujo constante de trabajo sin intervención manual constante. En esta actividad se realizan 5 procesos por lote, a este le asignamos una cantidad de tiempo estimado, esto nos ayuda a comprender el proceso en el que el sistema asigna los recursos para el proceso, al dividirlo en lotes también observamos el como se maneja la transición de cada lote (como se dice en la teoría del procesamiento por lotes).

En resumen, esta actividad es sumamente útil para comprender la teoría del procesamiento por lotes al igual que darnos un vistazo al pasado de las computadoras y ponernos en los pies de aquellos que trabajaron en dichas computadoras, al igual que entender el funcionamiento interno de un sistema.

Notas acerca del lenguaje

Lenguaje usado: C++

Motivo: Realmente iba a utilizar Python para esta actividad, pues en lo personal considero que para manejar información es mucho mas sencillo que C++, sin embargo, utilice estructuras de datos las cuales solamente e practicado en C++, al igual de que es el lenguaje que mas usamos en la carrera así que me sirve para seguir practicándolo (esto no significa que no esté interesado en Python, pues posiblemente pase este código a Python en un futuro para practicarlo). Otro motivo es que ya había realizado un trabajo muy similar a este, pero con el sistema de un Banco por lo que lo utilice de base para esta actividad.

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 existan mas “Lotes” que procesos hice que la cantidad de colas sea equivalente a la cantidad de procesos, por lo que siempre se podrán almacenar todos los procesos.

```

33
34 struct Process {
35     string name;
36     string operation;
37     int number1;
38     int number2;
39     int result;
40     int estimatedTime;
41     int programNumber;
42     int currentQueue;
43     int timeelapsed;
44 };

```

Ilustración 1 Estructura

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.

```

47 bool ValidaNumerosEnteros(char *dato){
48     bool ban = true;
49     int i = 0;
50     if (*dato == '-' || *dato == '+') {
51         i++;
52     }
53     while (*(dato + i) != '\0') {
54         if (*(dato + i) < '0' || *(dato + i) > '9') {
55             ban = false;
56             break;
57         }
58         i++;
59     }
60     return ban;
61 }
62

```

Ilustración 2 ValidaNumerosEnteros

```
78 void IWillHaveOrder(){
79     int x = 1, y = 1;
80     gotoxy(0,0);
81     printf("%c", 201); //┐
82     gotoxy(132,0);
83     printf("%c", 187); //┘
84     gotoxy(0,31);
85     printf("%c", 200); //└
86     gotoxy(132,31);
87     printf("%c", 188); //┌
88     while (y<=30)
89     {
90         gotoxy(0,y);
91         printf("%c", 186); //||
92         gotoxy(132,y);
93         printf("%c", 186); //||
94         y++;
95     }
96     while (x<=131)
97     {
98         gotoxy(x,0);
99         printf("%c", 205); //═
100        gotoxy(x,31);
101        printf("%c", 205); //═
102        x++;
103    }
104    y = 1;
105    int y2 = 3;
```

Ilustración 4 IWillHaveOrder

```
63 void gotoxy(int x,int y){
64     HANDLE hcon;
65     hcon = GetStdHandle(STD_OUTPUT_HANDLE);
66     COORD dwPos;
67     dwPos.X = x;
68     dwPos.Y= y;
69     SetConsoleCursorPosition(hcon,dwPos);
70 }
71
```

Ilustración 3 gotoxy

```

174 void datosLotes(){
175     char totalProcessesc[100];
176     gotoxy(1,1);
177     cout << "Ingrese el numero de procesos: ";
178     cin >> totalProcessesc;
179     while(!ValidaNumerosEnteros(totalProcessesc)){
180         gotoxy(1,1);
181         cout << " ";
182         gotoxy(1,1);
183         cout << "Ingrese el numero de procesos de nuevo: ";
184         cin >> totalProcessesc;
185     }
186     totalProcesses = atoi(totalProcessesc);
187     totallotes = totalProcesses % 5;
188     if (totallotes == 0) {
189         totallotes = 0;
190     } else {
191         totallotes = totalProcesses + (5 - totallotes);
192     }
193     totallotesr = totallotes / 5;
194     totallotesrr = totallotes - (totallotesr*4);
195
196     int currentQueue = 0;
197     for (int i = 1; i <= totalProcesses; ++i) {
198         Process newProcess;
199         system("cls");
200         ThisIsOrder();
201         gotoxy(1,1);
202         cout << "Proceso #" << i << ":" << endl;
203         gotoxy(1,3);
204         cout << "Ingrese el nombre del proceso: ";
205         cin >> newProcess.name;
206         newProcess.currentQueue = currentQueue+1;
207         do{
208             gotoxy(1,5);

```

Ilustración 6 datoslotes

```

146 void ThisIsOrder(){
147     int x = 1, y = 1;
148     gotoxy(0,0);
149     printf("%c", 201); //┐
150     gotoxy(132,0);
151     printf("%c", 187); //┑
152     gotoxy(0,31);
153     printf("%c", 200); //└
154     gotoxy(132,31);
155     printf("%c", 188); //┘
156     while (y<=30)
157     {
158         gotoxy(0,y);
159         printf("%c", 186); //||
160         gotoxy(132,y);
161         printf("%c", 186); //||
162         y++;
163     }
164     while (x<=131)
165     {
166         gotoxy(x,0);
167         printf("%c", 205); //=-
168         gotoxy(x,31);
169         printf("%c", 205); //=-
170         x++;
171     }
172 }
173

```

Ilustración 5 ThisIsOrder

```
437 int main() {
438     HWND consoleWindow = GetConsoleWindow();
439     RECT desktop;
440     GetWindowRect(GetDesktopWindow(), &desktop);
441     MoveWindow(consoleWindow, desktop.left, desktop.top, desktop.right, desktop.bottom, TRUE);
442     system("pause");
443     ThisIsOrder();
444     datosLotes();
445     system("cls");
446     IWillHaveOrder();
447     imprimirdata();
448     gotoxy(70,30);
449     system("pause");
450     system("cls");
451     return 0;
452 }
453
```

Ilustración 8 main

```
292 void imprimirdata() {
293     int procesosTotales = totalProcesses;
294     int lotesf = totallotesrr;
295     int ffy = 7;
296     for (int i = 0; i < maxQueues; i++) {
297         gotoxy(1,1);
298         cout << "Lotes restantes: " << lotesf;
299         int ay = 7;
300         int y = 7;
301         if (!queues[i].empty()) {
302             gotoxy(3,3);
303             cout << "Lote #" << i + 1 << endl;
304             queue<Process> tempQueue = queues[i]; // Copia temporal de la cola
305             while (!tempQueue.empty()) {
306                 // fila 1
307                 Process tprocess = tempQueue.front();
308                 tempQueue.pop();
309                 gotoxy(3,5);
310                 cout << "ID";
311                 gotoxy(3,y);
312                 cout << tprocess.programNumber;
313                 gotoxy(6,5);
314                 cout << "TME";
315                 gotoxy(6,y);
316                 cout << tprocess.estimatedTime;
317                 y = y + 2;
318             }
319             while(!queues[i].empty()){
320                 // fila 2
321                 Process process = queues[i].front();
322                 gotoxy(12,3);
323                 cout << "Ejec";
324                 gotoxy(12,5);
325                 cout << "Nn "; // nombre
326                 gotoxy(12,7);

```

Ilustración 7 imprimirdata

Como mencione anteriormente para solucionar este programa lo que hice fue utilizar colas estáticas, estas me servirían para simular el proceso de lotes con exactitud, esto debido a que las colas funcionan por “FIFO”, o sea, el primero en entrar es el primero en salir al igual que en los lotes que se trata de simular, ahora el por qué son estáticas es simplemente para facilitarme el ponerle límite de 5 “procesos” por cola. Primero el usuario tendría que ingresar la cantidad de procesos una vez que el usuario ingresa la cantidad de procesos se le solicitarían los datos que llevarían, o sea, el nombre, la operación, el número, el segundo número, el tiempo estimado, el tiempo transcurrido, etcétera. En cuanto el usuario ingresa la operación esta calcula el resultado, sin embargo en el momento de la impresión para dar la “ilusión” de que se tarda utilice Sleep y gotoxy para que se muestre una vez que terminara el tiempo de ejecución, en general, eso aplico para toda la impresión, pues con la ayuda de Sleep controle los tiempo para que una vez que pasara el tiempo máximo estimado este mostrara los resultados, aunque esto es algo que tendré que cambiar en el próximo programa, pues en el siguiente se manejara con el tiempo en base al tiempo faltante, y con la ayuda de gotoxy le di forma a todo el código, es verdad que esto trae problemas pero nada que no se solucione calculando correctamente las coordenadas al igual que borrar correctamente las mismas. Realmente no tiene mucha ciencia el código, o sea, si tengo muchas declaraciones pero estas son para controlar en su mayoría el tiempo o la cantidad de procesos y lotes, pues para calcular la cantidad de lotes tuve que realizar una operación matemática, esto debido a que técnicamente la cantidad de lotes que puedo ingresar la limite a la cantidad de procesos que ingrese el usuario, si lo sé, puede ser confuso pero básicamente así me aseguro de que siempre existan mas lotes, pues si el usuario ingresa 5 procesos estos solo ocuparían un lote pero realmente se generaron 5, aunque los otros 4 estén

```

73     const int maxProcessesPerQueue = 5;
74     const int maxQueues = maxProcessesPerQueue;
75     queue<Process> queues[maxQueues];
76     unordered_set<int> usedProgramNumbers;
77
78     int main() {

```

Ilustración 9 cola/fila

Modelos de sistemas operativos

en desuso, lo mismo pasaría con 10 procesos, se generarían 10 lotes pero solo 2 estarían en uso, algo rebuscado pero en su momento fue lo que mejor se me ocurrió.

Lotes restantes: 3		Terminados				Procesos restantes
Lote #1	Ejec	ID	Ope	Res	Lote	
ID TME	Nn 4					
	ID 4	1	1+1	2	1	8
	Ope +	2	2+2	4	1	Tiempo total
	TME 4	3	3+3	6	1	6
4 4	TT 0					
5 5	TR 4					

Ilustración 11 Resultado

Lotes restantes: 2		Terminados				Procesos restantes
Lote #2	Ejec	ID	Ope	Res	Lote	
ID TME	Nn 9					
	ID 9	1	1+1	2	1	3
	Ope +	2	2+2	4	1	Tiempo total
	TME 9	3	3+3	6	1	43
9 9	TT 7	4	4+4	8	1	
10 10	TR 2	5	5+5	10	1	
		6	6+6	12	2	
		7	7+7	14	2	
		8	8+8	16	2	

Ilustración 10 Resultado 2

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.

Conclusión

En conclusión, realmente esta actividad fue hasta cierto punto sencilla, pues la verdad en mi caso lo que más se me complicó fue la realización de lo visual, por el uso de gotoxy, soy consciente de que existen mejores métodos, pero como este es el único que me se pues lo realice mediante el. Pero en sí, esta práctica me dio una idea de como funcionaban las computadoras antiguas que funcionaban por el procesamiento por lotes, lo cual es hasta cierto punto

gratificante. Ahora si bien mi código tal vez no sea el mejor, pues realmente lo limite mucho para que funcionara internamente bien, ahora con el próximo programa tendré que reestructurar casi todo el código. Considero que el uso de colas para simular lotes fue el correcto.

Enlace de descarga:

<https://drive.google.com/drive/folders/1TN4srV2rW0VlOr06FzKJoDVGhpPloLig?usp=sharing>