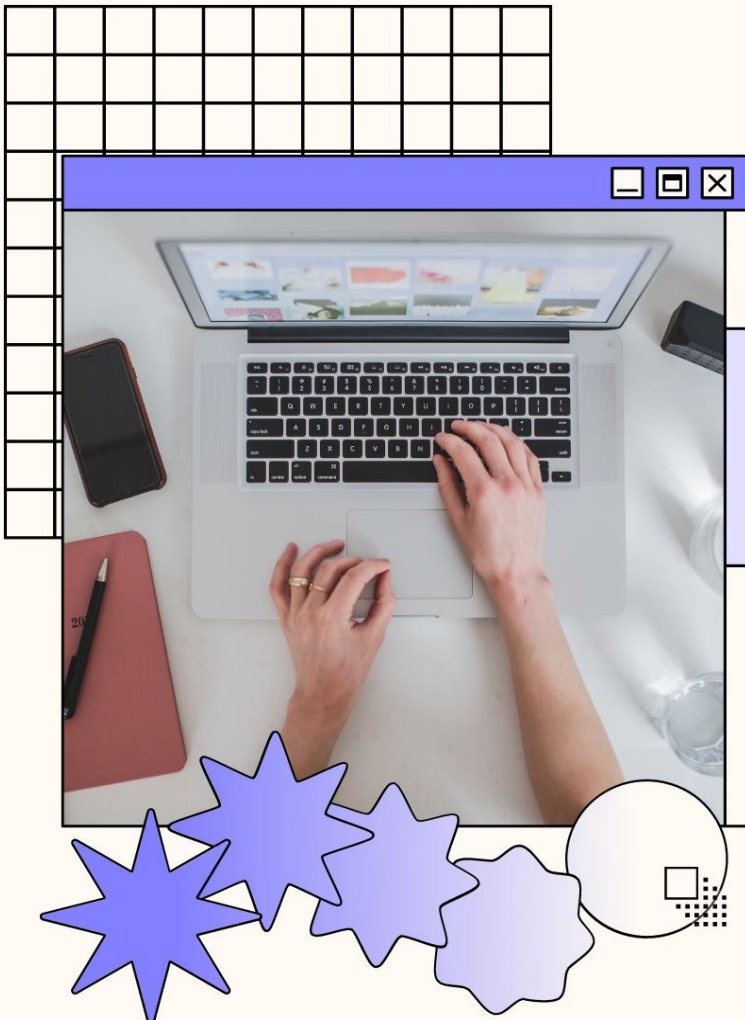


**Centro Universitario de Ciencias
Exactas e Ingenierías**

Universidad de Guadalajara

REPORTE 07

Productor - Consumidor



Alumnos:

Carbajal Armenta Yessenia Paola
Sánchez Lozano Jonathan

Códigos:

220286482
215768126

Profesora:

Becerra Velázquez Violeta del Rocío

Materia:

Seminario de Soluciones de
Problemas de Sistemas Operativos

Departamento:

Ciencias Computacionales

Carrera:

Ingeniería en Computación

NRC:

103844

Sección:

D01

Actividad No. 14

Paginación Simple

- El algoritmo de planificación a implementar es el de RR (Actividad 10).
- Cumplir con todos los requerimientos del programa 5 (actividad 10).
- De manera aleatoria se asignará, el tiempo, operación y tamaño de cada proceso (número aleatorio entero entre 6 y 28).
- El tamaño de la memoria será de 220.
- Dividir la memoria en “Marcos” de igual longitud (44 marcos de 5 cada uno).
- El S.O. ocupará 4 marcos, quedando 40 disponibles para repartir entre los procesos.
- Dividir los procesos en páginas (tamaño 5), es decir dividir los procesos de igual tamaño que los marcos (5).

Objetivo

Para esta práctica se trabajó con el aspecto de la memoria, con la penúltima práctica que hicimos manejamos la memoria con solo un máximo de 5 procesos, considerando los estados de ejecución, listo y bloqueado. La idea general de este programa es implementar la técnica de paginación simple para resolver la gestión de la memoria.

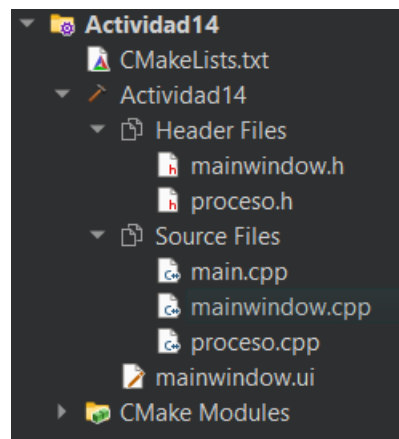
Se deberá mostrar la memoria, con la información del número de marco, la cantidad de espacios que se ocupan dentro del frame, quién ocupa dicho frame y el estado del proceso que esté ocupando el frame. Además, en caso de que se muestre toda la memoria en la pantalla, al teclear la tecla “T”.

Finalmente, se deberán conservar con todos los requisitos del programa 5, el algoritmo de planificación Round Robin.

Desarrollo

Para este programa decidimos volver tanto a C++ como al framework que habíamos estado utilizando hasta el momento, debido a que la dificultad y complejidad de los últimos 2 programas es más elevada que el resto de los programas que habíamos estado realizando, por lo que cambiar a otro lenguaje a estas alturas del semestre sería muy complicado para tener a tiempo el programa, aparte que considerando que es necesario cumplir con los requisitos del programa 5, nos ahorramos un poco de tiempo seguir trabajando sobre de este programa.

Se realizaron los siguientes archivos:



Lo primero que realizamos para esta actividad fue modificar la clase del “Proceso” para que ahora se puedan almacenar y recuperar los valores correspondientes al tamaño del proceso y la cantidad de frames que se van a necesitar en base a su tamaño. El tamaño del proceso es un número aleatorio entre 6 y 28.

```
69 void Proceso::setTamaño(int value)
70 {
71     tamaño = value;
72     frames = (tamaño / PAGES_PER_FRAME) + (tamaño % PAGES_PER_FRAME == 0 ? 0 : 1);
73 }
74
75 int Proceso::getTamaño() const
76 {
77     return tamaño;
78 }
```

Posterior a la modificación de la clase “Proceso”, añadimos a su operador de asignación las correspondientes de los nuevos atributos. Utilizamos un simple struct que almacenará la información de la cantidad de páginas libres dentro de ese frame, el ID del proceso que esté ocupando dicho frame y el estado del proceso; todo esto con la intención de poder mostrar en pantalla de manera correcta lo que está pasando en el programa.

```
24 #define PAGES 5
25 #define FRAMES 43
26 #define FREE_PAGE 0
27 #define SO -4

34 struct Frame{
35     int freePages = 5;
36     int usedBy = FREE_PAGE;
37     int state = FREE_PAGE;
38 };
```

Ya con la memoria lista, lo siguiente fue reemplazar todas las partes en las que se necesitaba validar el tamaño máximo de la memoria y adecuarlos a que ahora ya no está sujeto a una cantidad límite, sino que la memoria ahora sí se llene realmente.

Para cumplir con lo anterior creamos una serie de métodos que nos permiten saber si la memoria está llena, la cantidad de espacios que están libres, ingresar la información del proceso en la memoria, o eliminar dicha información. Comenzamos con el primer método llamado: filledMemory, éste nos retorna un true en caso de que la memoria esté llena, este solo nos permite validar los casos en los que todos los frames estén ocupados, por lo que al inicio no resulta tan útil, pero se puede aplicar para validar ciertas cuestiones a la hora de mostrar la información en pantalla o checar si un nuevo elemento puede entrar.

```
572 bool MainWindow::filledMemory()
573 {
574     int i, filled = 0;
575
576     for (i = 0; i < FRAMES - 5; i++){
577         if (memory[i].freePages < PAGES){
578             filled++;
579         }
580     }
581     return filled == FRAMES - 5;
582 }
```

```

584 int MainWindow::availableFrames()
585 {
586     int i, availableFrames;
587
588     for (i = availableFrames = 0; i < FRAMES - 5; i++){
589         if (memory[i].freePages == PAGES){
590             availableFrames++;
591         }
592     }
593     return availableFrames;
594 }
595
596 void MainWindow::fillFrames(int size, int frames, int id, int state)
597 {
598     int i, j;
599
600     for (i = j = 0; i < FRAMES - 5; i++){
601         if (memory[i].freePages == PAGES and j < frames){
602             if (j == frames - 1){
603                 if (size % PAGES != 0){
604                     memory[i].freePages -= size % PAGES;
605                 }
606                 else{
607                     memory[i].freePages = 0;
608                 }
609             }
610             else{
611                 memory[i].freePages = 0;
612             }
613
614             qDebug() << "Free pages = " << memory[i].freePages;
615             memory[i].usedBy = id;
616             memory[i].state = state;
617             j++;
618         }
619     }
620 }

```

Por consiguiente, se programó la nueva tecla que nos permite ver la tabla de memoria donde observamos lo que es la paginación simple, la cual manda a llamar a un método que muestra esa ventana con su respectiva tabla utilizando los datos necesarios que se deben de imprimir en ella, tales como el marco, el espacio, el id y el estado de cada uno de los procesos.

```

761 else if (event->key() == Qt::Key_T and state == RUNNING){
762     qDebug() << "Tabla de memoria";
763     state = PAUSED;
764     ui->stackedWidget->setCurrentIndex(SHOW_MEMORY_TABLE);
765     setFixedSize(500,720);
766     showProcessesBcp();
767     this->setFocus();
768     pause.exec();
769     this->setFocus();
770 }

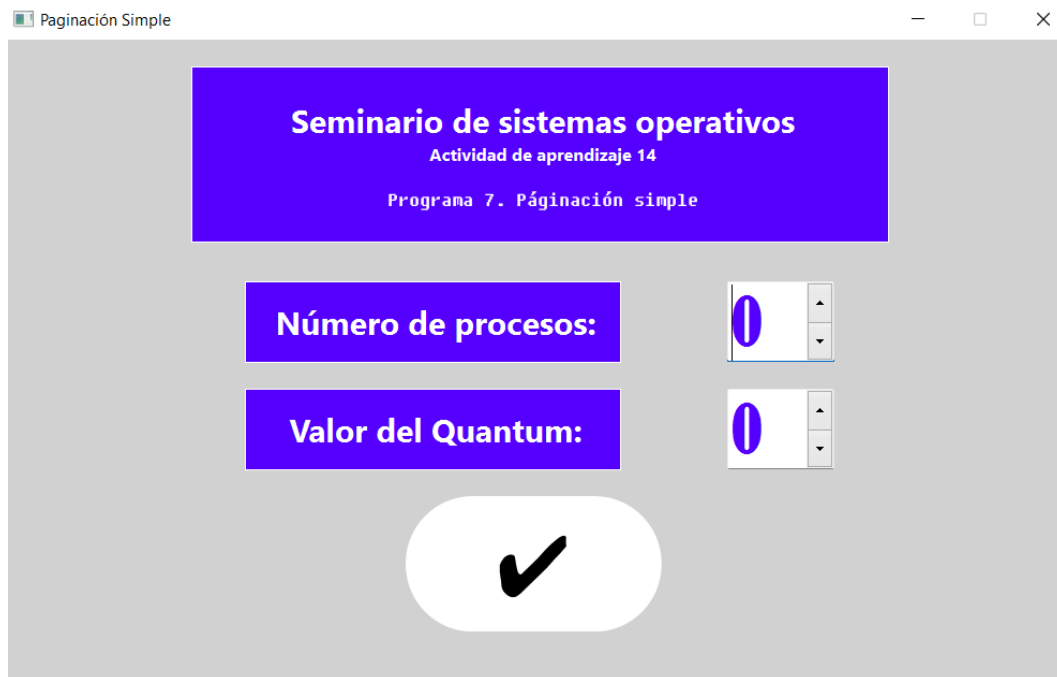
```

```

622 void MainWindow::showMemoryTable()
623 {
624     int i, rows = 0;
625     std::string state = "";
626
627     for (i = 0; i < FRAMES - 3; i++, rows++){
628         ui->memoryTB->setItem(rows,0,new QTableWidgetItem(QString::number(i)));
629         ui->memoryTB->setItem(rows,1,new QTableWidgetItem(QString::number(PAGES - memory[i].freePages) + "/5"));
630         ui->memoryTB->setItem(rows,2,new QTableWidgetItem(QString::number(memory[i].usedBy)));
631         switch (memory[i].state){
632             case States::Listo:state = "Listo"; break;
633             case States::Ejecutandose:state = "Ejecución"; break;
634             case States::Bloqueado:state = "Bloqueado"; break;
635             default:
636                 state = "Libre";
637         }
638         ui->memoryTB->setItem(rows,3,new QTableWidgetItem(QString(state.c_str())));
639     }
640
641     for (i = 40; i <= FRAMES; i++){
642         ui->memoryTB->setItem(i,0,new QTableWidgetItem(QString::number(i)));
643         ui->memoryTB->setItem(i,1,new QTableWidgetItem(QString("5/5")));
644         ui->memoryTB->setItem(i,2,new QTableWidgetItem(QString("S0")));
645         ui->memoryTB->setItem(i,3,new QTableWidgetItem(QString("Ocupado ")));
646     }
647 }

```

Programa en ejecución



Cola de listos

ID	M. Estimac	T. Transcurrido
----	------------	-----------------

Procesos terminados

ID	Operación	Resultado
----	-----------	-----------

Proceso

Procesos nuevos:

Tiempo Quantum:

Tiempo transcurrido:

Simular

Proceso en ejecución

ID	Operación
	T. M. Estimado
	T. Transcurrido
	T. Restante
	Tamaño
	Quantum

Cola de bloqueados

ID	T. T. en bloqueado
----	--------------------

Cola de listos

ID	M. Estimac	T. Transcurrido
1	10	6
6	9	4
4	12	3
7	11	4
8	13	4
11	10	2
12	13	2
3	6	6

Procesos terminados

ID	Operación	Resultado
9	35%-51	ERROR
10	91+2	ERROR

Siguiente: ID 15 Tamaño: 13

Procesos nuevos:

Tiempo Quantum:

Tiempo transcurrido:

Simular

Proceso en ejecución

ID	Operación
5	43-5
	T. M. Estimado
	T. Transcurrido
	T. Restante
	Tamaño
	Quantum

Cola de bloqueados

ID	T. T. en bloqueado
----	--------------------

Datos de los procesos

ID	Estado	T. Bloqueado	Operación	Resultado	T. Llegada	T. Finalización	T. Retorno	T. Espera	T. Servicio	T. Restante	T. Respuesta
1	LISTO	0	-24*-50	NULL	0	NULL	NULL	34	6	4	NULL
2	LISTO	0	-100-15	NULL	0	NULL	NULL	37	3	5	NULL
3	LISTO	0	-40+50	NULL	0	NULL	NULL	34	6	0	NULL
4	LISTO	0	10/-67	NULL	0	NULL	NULL	37	3	9	NULL
5	LISTO	0	43-5	NULL	0	NULL	NULL	36	4	4	NULL
6	LISTO	0	-40*4	NULL	3	NULL	NULL	36	4	5	NULL
7	LISTO	0	4/3	NULL	3	NULL	NULL	36	4	7	NULL
8	LISTO	0	38%-67	NULL	4	NULL	NULL	36	4	9	NULL
9	FINALIZADO	ERROR	35%-51	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR
10	FINALIZADO	ERROR	91+2	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR
11	LISTO	0	-26*33	NULL	18	NULL	NULL	38	2	8	NULL
12	LISTO	0	28%-85	NULL	19	NULL	NULL	38	2	11	NULL
13	EJECUCION	0	-70+1	NULL	20	NULL	NULL	39	1	6	NULL
14	LISTO	0	-69-1	NULL	21	NULL	NULL	40	0	7	NULL
15	NUEVO	NULL	-31*-50	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabla de memoria

Marco	Espacio	Proceso	Estado
0	5/5	1	Listo
1	5/5	1	Listo
2	4/5	1	Listo
3	5/5	2	Listo
4	5/5	2	Listo
5	1/5	2	Listo
6	0/5	0	Libre
7	0/5	0	Libre
8	5/5	4	Listo
9	5/5	4	Listo
10	5/5	4	Listo
11	4/5	4	Listo
12	5/5	5	Listo
13	5/5	5	Listo
14	2/5	5	Listo

Conclusiones


Carbajal Armenta Yessenia Paola:

Este programa ha sido de los más complicados del curso, debido a que por ser a final de semestre tenemos varios proyectos a la vez y resulta complicado administrar bien el tiempo para cada tarea. Además, al contener elementos nuevos resultó más desafiante implementar ese manejo de memoria que se mostró en esta actividad.

Sánchez Lozano Jonathan:

Esta práctica ha sido de las difíciles y más complicadas del curso, ya que debido a todas las modificaciones que tuvimos que realizar para que el programa funcione correctamente nos tardamos un poco más tiempo del esperado, y considerando que ya estamos a semanas de terminar el semestre tenemos más carga de trabajo de todas las materias, como proyectos, exámenes entre otras. Pero al final pudimos concluir con el programa con todas las especificaciones requeridas.

Enlace del código

 https://drive.google.com/drive/folders/1g-0UvV8UReMZnG2Pb4--W_fC2ICEQKM3?usp=sharing

Enlace del vídeo

 https://drive.google.com/file/d/1gccV7IfcA5D1Xw7ffLeqF_X9yzKVQFVf/view?usp=sharing