

HOME

ABOUT

MORE



SISTEMA DE GESTION SIMULADOR ADMINISTRADOR DE TAREAS

ESTRUCTURAS DINÁMICAS LINEALES: LISTA, COLA Y PILA

INTEGRANTES:

- FRIDMAN HARRY PAPEL ACHAHUI
- .Yoshet Pata
- .Anderson Soncco Daza
- .Gilberth Ricardo Rodriguez Ccallo

¿QUÉ PROBLEMA BUSCAMOS RESOLVER?

Simular el funcionamiento de un sistema operativo en cuanto a la creación, ejecución y finalización de procesos, así como gestionar de manera eficiente la memoria asignada a cada uno. Todo esto implementado de forma completamente manual, sin recurrir a librerías de datos estándar, para replicar los aspectos fundamentales de un sistema operativo real.



REQUISITOS FUNCIONALES Y NO FUNCIONALES

Funcionales:

- Crear procesos (ID, nombre, prioridad)
- Planificar ejecución (cola de prioridad)
- Asignar/liberar memoria (pila)

No funcionales:

- Interfaz por consola
- Código modular, comentado
- Sin usar vector, queue, etc.



ESTRUCTURAS DE DATOS PROPUESTAS

El sistema se construyó con tres estructuras dinámicas implementadas desde cero.

- Lista enlazada: se usa para el gestor de procesos, permitiendo insertar, eliminar, buscar y modificar procesos de forma eficiente.
- Cola de prioridad: utilizada en el planificador de CPU, asegura que los procesos con mayor prioridad se ejecuten primero.
- Pila: empleada para la gestión de memoria, facilita la asignación y liberación de bloques siguiendo un orden LIFO.

DESCRIPCIÓN DE ESTRUCTURAS DE DATOS Y OPERACIONES

Lista Enlazada – Gestor de Procesos

- Guarda los procesos activos.
- Permite insertar, eliminar, buscar y modificar procesos.
- Cada nodo contiene ID, nombre, prioridad y puntero al siguiente.

Cola de Prioridad – Planificador de CPU

- Organiza los procesos según prioridad.
- Los procesos más urgentes se ejecutan primero.
- Estructura implementada manualmente con nodos ordenados.

Pila – Gestor de Memoria

- Administra bloques de memoria dinámica.
- Sigue una lógica LIFO (último en entrar, primero en salir).
- Se usa para asignar y liberar memoria por proceso.

OPERACIONES EN LA LISTA ENLAZADA

`insertarProceso(nombre, prioridad)`

Crea e inserta un nuevo proceso al final de la lista.

`eliminarProceso(id)`

Elimina un proceso dado su ID, ajustando punteros.

`buscarPorId(id)`

Recorre la lista y devuelve el proceso si existe.

`modificarPrioridad(id, nuevaPrioridad)`

Cambia la prioridad de un proceso existente.

`mostrarProcesos()`

Muestra todos los procesos activos con sus datos.

OPERACIONES DEL PLANIFICADOR Y LA MEMORIA

Cola de Prioridad [CPU]

`encolarProceso(proceso)`

Inserta el proceso en la posición correcta según prioridad.

`desencolarProceso()`

Ejecuta el proceso con mayor prioridad.

`mostrarCola()`

Lista los procesos en cola.



Pila de Memoria

`asignarMemoria(tamaño, id_proceso)`

Asigna memoria si hay suficiente disponible.

`liberarMemoria()`

Libera el último bloque de memoria asignado.

`mostrarEstadoMemoria()`

Muestra la memoria total, disponible y bloques activos.

CODIGO

```
1 #include <iostream>
2 #include <string>
3 #include <windows.h>
4 using namespace std;
5
6 // Prototipos de funciones generales
7 void limpiarConsola();
8 int validarPrioridad();
9 int validarMemoria();
10 string obtenerNombrePrioridad(int prioridad);
11 void pausar();
12
13 // Estructuras de datos
14 struct Proceso {
15     int id;
16     string nombre;
17     int prioridad; // 0=Baja, 1=Media, 2=Alta
18     int memoria;
19 };
20
21 struct NodoLista {
22     Proceso proceso;
23     NodoLista* siguiente;
24 };
25
26 struct GestorProcesos {
27     NodoLista* cabeza;
28     int contador_id;
29 };
```

```
29    };
30
31     struct NodoCola {
32         Proceso proceso;
33         NodoCola* siguiente;
34     };
35
36     struct PlanificadorCPU {
37         NodoCola* frente;
38     };
39
40     struct NodoPila {
41         int tamano;
42         int id_proceso;
43         NodoPila* siguiente;
44     };
45
46     struct GestorMemoria {
47         NodoPila* tope;
48         int memoria_total;
49         int memoria_disponible;
50     };
51
```

```
// Prototipos de funciones del Gestor de Procesos
void inicializarGestor(GestorProcesos* gp);
void insertarProceso(GestorProcesos* gp, string nombre, int prioridad, int memoria);
bool eliminarProceso(GestorProcesos* gp, int id);
Proceso* buscarPorId(GestorProcesos* gp, int id);
Proceso* buscarPorNombre(GestorProcesos* gp, string nombre);
bool modificarPrioridad(GestorProcesos* gp, int id, int nuevaPrioridad);
void mostrarProcesos(GestorProcesos* gp);

// Prototipos de funciones del Planificador de CPU
void inicializarPlanificador(PlanificadorCPU* pc);
void encolarProceso(PlanificadorCPU* pc, Proceso p);
Proceso desencolarProceso(PlanificadorCPU* pc);
void mostrarCola(PlanificadorCPU* pc);
bool estaVacia(PlanificadorCPU* pc);

// Prototipos de funciones del Gestor de Memoria
void inicializarGestorMemoria(GestorMemoria* gm, int total);
bool asignarMemoria(GestorMemoria* gm, int tamano, int id_proceso);
bool liberarMemoria(GestorMemoria* gm);
void mostrarEstadoMemoria(GestorMemoria* gm);
```

EJECUCION DEL PROGRAMA

[HOME](#)[ABOUT](#)[MORE](#)

```
Administrator de Procesos X + ▾  
--- Sistema de Gestión de Procesos ---  
1. Gestor de Procesos  
2. Planificador de CPU  
3. Gestor de Memoria  
4. Salir  
Seleccione una opción: |
```

```
--- Gestor de Procesos ---  
1. Insertar nuevo proceso  
2. Eliminar proceso  
3. Buscar proceso por ID  
4. Buscar proceso por nombre  
5. Modificar prioridad  
6. Mostrar todos los procesos  
7. Regresar  
Seleccione una opción: |
```

```
--- Gestor de Procesos ---  
1. Insertar nuevo proceso  
2. Eliminar proceso  
3. Buscar proceso por ID  
4. Buscar proceso por nombre  
5. Modificar prioridad  
6. Mostrar todos los procesos  
7. Regresar  
Seleccione una opción: 1  
Nombre del proceso: Google  
Prioridad (0=Baja, 1=Media, 2=Alta): 2  
Memoria requerida (MB): 900  
  
Proceso agregado exitosamente.  
Presione Enter para continuar...
```

```
--- Planificador de CPU ---  
1. Encolar proceso  
2. Desencolar y ejecutar proceso  
3. Mostrar cola de procesos  
4. Regresar  
Seleccione una opción: |
```

```
--- Cola de Procesos (Prioridad) ---  
ID: 1, Nombre: Google, Prioridad: Alta, Memoria: 900MB  
-----  
  
Presione Enter para continuar...
```

EJECUCION DEL PROGRAMA

HOME

ABOUT

MORE

```
---- Gestor de Memoria ----  
1. Asignar memoria a proceso  
2. Liberar memoria  
3. Mostrar estado de memoria  
4. Regresar  
Seleccione una opción: 1  
ID del proceso: 1  
Memoria a asignar (MB): 900
```

Memoria asignada exitosamente.
Presione Enter para continuar...

```
---- Estado de Memoria ----  
Memoria total: 16384MB  
Memoria disponible: 15484MB  
Memoria en uso: 900MB  
Bloques asignados:  
Proceso ID: 1, Bloque: 900MB
```

Presione Enter para continuar...

HOME

ABOUT

MORE

THANK YOU

www.reallygreatsite.com