SIMULADOR DE GESTIÓN DE MEMORIA - VERSIÓN ANSI C (C90)

⚠️ VERSIÓN PARA PROFESORES ESTRICTOS

Esta versión del simulador cumple **estrictamente** con el estándar **ANSI C (C89/C90)**, diseñada para profesores que exigen cumplimiento total del estándar.

🎯 Estándares Cumplidos

☑ **ANSI C (C89/C90)** - Estándar ISO/IEC 9899:1990

☑ **Todas las variables declaradas al inicio de cada función**

☑ **Comentarios estilo `/* */`** (no se usa `//`)

☑ **Sin declaraciones en bucles `for`**

☑ **Sin extensiones de GNU**

☑ **Compila con `-ansi -pedantic` sin errores ni advertencias**

📋 Archivos Incluidos

1. **`gestion_memoria_ansi.c`** - Código fuente compatible con ANSI C
2. **`compilar_ansi.sh`** - Script de compilación estricta
3. **`CAMBIOS_ANSI_C.md`** - Documentación detallada de cambios
4. **`README_ANSI.md`** - Este archivo

🔧 Compilación

Opción 1: Usar el script (recomendado)

```bash
chmod +x compilar_ansi.sh
./compilar_ansi.sh
```

Opción 2: Compilación manual

```bash
gcc -ansi -pedantic -Wall -Wextra -o gestion_memoria_ansi gestion_memoria_ansi.c
```

Explicación de las banderas:

- **`-ansi`**: Activa modo ANSI C (equivalente a `-std=c89`)
- **`-pedantic`**: Rechaza extensiones no estándar
- **`-Wall`**: Todas las advertencias comunes
- **`-Wextra`**: Advertencias adicionales

🚀 Ejecución

```bash
./gestion_memoria_ansi
```

Configuración sugerida para pruebas:

```
Tamaño total de memoria: 1000 KB
Tamaño de partición: 100 KB
→ Esto creará 10 particiones de 100 KB cada una
```

📝 Diferencias con la Versión Moderna

Aspecto	Versión Moderna (C99)	Versión ANSI C (C90)
Comentarios	`// comentario`	`/* comentario */`
Variables	En cualquier lugar	Al inicio de funciones
Bucle for	`for(int i=0; ...)`	`int i; for(i=0; ...)`

| **Declaraciones** | Mezcladas con código | Todas al inicio |

🎓 Principales Cambios Realizados

1. Variables al Inicio

ANTES (C99):

```c
void funcion(void) {
    printf("Hola\n");
    int x = 5;  /* ERROR en ANSI C */
    for (int i = 0; i < 10; i++) { /* ERROR en ANSI C */
        /* código */
    }
}
```

DESPUÉS (ANSI C):

```c
void funcion(void) {
    int x;
    int i;

    printf("Hola\n");
    x = 5;
    for (i = 0; i < 10; i++) {
        /* código */
    }
}
```

2. Comentarios

ANTES (C99):

```c
// Este es un comentario de línea
```

int x = 5; // Comentario al final
```

**DESPUÉS (ANSI C):**
```c
/* Este es un comentario tradicional */
int x = 5;  /* Comentario al final */
```

3. Cast Explícito
ANTES:
```c
srand(time(NULL));
```

DESPUÉS:
```c
srand((unsigned int)time(NULL));
```

🎯 Funcionalidades del Programa

Menú Principal:
1. **Crear Proceso** - Genera proceso con tamaño aleatorio
2. **Cerrar Proceso** - Libera partición y elimina proceso
3. **Ver Tabla de Procesos** - Muestra procesos activos
4. **Ver Tabla de Particiones** - Muestra estado y fragmentación
5. **Ver Memoria Completa** - Visualización del vector de memoria
6. **Ver Todas las Tablas** - Muestra todo el estado del sistema
0. **Salir** - Libera recursos y finaliza

Características:
- ☑ Algoritmo **Primer Ajuste (First Fit)**
- ☑ Cálculo de **fragmentación interna** en porcentaje
```

- ☑ Lista enlazada para gestión de procesos
- ☑ Tabla de particiones fija
- ☑ Visualización del vector de memoria

🍼 **Ejemplo de Ejecución**

```
=== INICIALIZACIÓN DEL SISTEMA DE MEMORIA ===
Ingrese el tamaño total de memoria (KB): 1000
Ingrese el tamaño de cada partición (KB): 100

>>> Se crearán 10 particiones de 100 KB cada una.
>>> Memoria inicializada exitosamente.

╔═══════════════════ MENÚ PRINCIPAL ═══════════════════╗
║  1. Crear Proceso                                    ║
║  2. Cerrar Proceso                                   ║
║  3. Ver Tabla de Procesos                            ║
║  4. Ver Tabla de Particiones                         ║
║  5. Ver Memoria Completa                             ║
║  6. Ver Todas las Tablas                             ║
║  0. Salir                                            ║
╚══════════════════════════════════════════════════════╝
Seleccione una opción: 1

--- CREANDO NUEVO PROCESO ---
Proceso ID: 1
Tamaño requerido: 75 KB
>>> PROCESO CREADO EXITOSAMENTE <<<
   Asignado a la partición: 0
   Dirección de inicio: 0
   Fragmentación interna: 25.00%
```

🔍 Verificación de Cumplimiento

Prueba de Compilación Estricta:

```bash
gcc -ansi -pedantic -Wall -Wextra -o test gestion_memoria_ansi.c
```

Resultado Esperado:

```
✓ 0 errores
✓ 0 advertencias
✓ Compilación exitosa
```

📖 Conceptos de Sistemas Operativos

Gestión de Memoria Particionada Estática Fija:

- Memoria dividida en **particiones de tamaño fijo**
- Cada partición contiene **un solo proceso**
- **No hay redimensionamiento** durante la ejecución

Algoritmo Primer Ajuste:

1. Recorrer tabla de particiones desde el inicio
2. Asignar proceso a la **primera partición libre** que quepa
3. Si no hay espacio, rechazar el proceso

Fragmentación Interna:

```
Fragmentación = ((Tamaño_Partición - Tamaño_Proceso) / Tamaño_Partición) × 100%
```

🎯 Por Qué Este Código es Excelente

Para tu Profesor:

☑ Cumple 100% con ANSI C (C90)

☑ Demuestra conocimiento de estándares históricos

☑ Código limpio y bien estructurado

☑ Todas las mejores prácticas aplicadas

Para ti:

☑ Aprenderás la diferencia entre C89/C90 y versiones modernas

☑ Entenderás por qué algunos profesores son estrictos

☑ Código portable a cualquier compilador

☑ Base sólida para sistemas embebidos

🛠️ Requisitos del Sistema

- **Sistema Operativo**: Linux (cualquier distribución)
- **Compilador**: GCC 4.0 o superior (o cualquier compilador ANSI C)
- **Librerías**: Estándar ANSI C (`stdio.h`, `stdlib.h`, `time.h`)

❓ Preguntas Frecuentes

¿Por qué ANSI C si es tan antiguo?

- Máxima portabilidad
- Compatible con sistemas embebidos
- Estándar en educación de fundamentos
- Muchos compiladores legacy solo soportan C90

¿Compila en compiladores modernos?

Sí, **100% compatible**. ANSI C es un subconjunto de C99/C11/C17.

¿Funciona igual que la versión moderna?

Sí, la **funcionalidad es idéntica**. Solo cambia la sintaxis y organización del código.

📜 Licencia

Código libre para uso académico y educativo.

✨ Notas Finales

Este código ha sido cuidadosamente adaptado para cumplir con los estándares más estrictos de ANSI C. Está listo para ser entregado a profesores que exigen cumplimiento total del estándar C89/C90.

¡Buena suerte con tu entrega! 🚀

Versión: ANSI C (C89/C90)

Fecha: 2026

Estándares: ISO/IEC 9899:1990

Compatibilidad: ☑ Compiladores antiguos y modernos