Evidencias de Pruebas

Simulador de Gestor de Procesos para Sistemas Operativos

Universidad Autónoma de Tamaulipas
Materia: Sistemas Operativos
Profesor: Dante Adolfo Muñoz Quintero
Semestre: 7º

Autores:

- Solbes Davalos Rodrigo
- Izaguirre Cortes Emanuel
- Rosales Pereles Denisse Ariadna
- Morales Urrutia Javier Antonio
- Reyes Alejo Emiliano

Fecha de Pruebas: Diciembre 2025

Introducción

Objetivo de las Pruebas

Este documento presenta las evidencias de las pruebas realizadas al Simulador de Gestor de Procesos, con el objetivo de verificar que todas las funcionalidades implementadas funcionan correctamente según las especificaciones.

Alcance

Las pruebas cubren:

- Compilación exitosa del programa
- Operaciones sobre procesos (crear, suspender, reanudar, terminar)
- Asignación y liberación de recursos
- Algoritmos de planificación (FCFS y Round Robin)
- Comunicación entre procesos (mensajes)

- Sincronización (semáforos)
- Prevención de deadlocks
- Sistema de logs y estadísticas

Ambiente de Pruebas

- **Sistema Operativo:** *[Completar con tu SO]*
- **Compilador:** *[Completar con versión de GCC]*
- **Fecha de Ejecución:** *[Completar]*
- **Versión del Simulador:** 1.0

Metodología de Pruebas

Tipos de Pruebas Realizadas

1. **Pruebas Funcionales:** Verifican que cada función opere correctamente
2. **Pruebas de Integración:** Validan la interacción entre módulos
3. **Pruebas de Algoritmos:** Verifican FCFS y Round Robin
4. **Pruebas de Sincronización:** Validan semáforos y comunicación
5. **Pruebas de Límites:** Verifican el comportamiento en casos extremos

Criterios de Aceptación

✓ **Pasa:** La funcionalidad opera según lo esperado
⚠ **Advertencia:** Funciona pero con comportamiento inesperado menor
✗ **Falla:** No cumple con los requisitos

Compilación del Programa

CP-00: Compilación Exitosa

Objetivo: Verificar que el programa compila sin errores ni advertencias.

Comando a ejecutar:

make clean

make

CAPTURA 00: Compilación del programa

```
PS C:\Users\RSD ESCRITORIO\Desktop\ESCRITORIO\SISTEMAS OPERATIVOS\Producto_Integrador_SO\src> make clean
if exist simulador_procesos.exe del /Q simulador_procesos.exe
"Γ£ô Archivos de compilaci├│n eliminados"
PS C:\Users\RSD ESCRITORIO\Desktop\ESCRITORIO\SISTEMAS OPERATIVOS\Producto_Integrador_SO\src> make
gcc -Wall -Wextra -std=c99 -g simulador_procesos.c -o simulador_procesos.exe
simulador_procesos.c: In function 'check_deadlock_prevention':
simulador_procesos.c:222:36: warning: unused parameter 'pid' [-Wunused-parameter]
 bool check_deadlock_prevention(int pid, Resources req) {
                                    ^~~
"Γ£ô Compilaci├│n exitosa"
"Ejecuta con: simulador_procesos.exe"
PS C:\Users\RSD ESCRITORIO\Desktop\ESCRITORIO\SISTEMAS OPERATIVOS\Producto_Integrador_SO\src>
```

Casos de Prueba

CP-01: Inicialización del Sistema con FCFS

Objetivo: Verificar que el sistema se inicializa correctamente con algoritmo FCFS.

Pasos a seguir:

1. Ejecutar: ./simulador_procesos
2. Cuando pregunte por el algoritmo, seleccionar: 1 (FCFS)
3. Observar pantalla de bienvenida y confirmación

Qué debe aparecer:

- Mensaje de bienvenida del simulador
- Solicitud de selección de algoritmo
- Confirmación "✓ Algoritmo FCFS seleccionado"
- Menú principal

📸 **CAPTURA 01: Inicialización con FCFS**

```
PS C:\Users\RSD ESCRITORIO\Desktop\ESCRITORIO\SISTEMAS OPERATIVOS\Producto_Integrador_SO\src> ./simulador_procesos
================================================================
  SIMULADOR DE GESTOR DE PROCESOS PARA SISTEMAS OPERATIVOS
          Universidad Autonoma de Tamaulipas
================================================================


========== SELECCION DE ALGORITMO DE PLANIFICACION ==========
1. FCFS (First Come First Served)
2. Round Robin
Seleccione un algoritmo (1-2): 1
Γ£ô Algoritmo FCFS seleccionado
============================================================

============================================================
     SIMULADOR DE GESTOR DE PROCESOS - MENU PRINCIPAL
============================================================
  1. Crear proceso
  2. Listar procesos activos
  3. Mostrar recursos disponibles
  4. Ejecutar siguiente paso
  5. Suspender proceso
  6. Reanudar proceso
  7. Terminar proceso
  8. Ver logs
  9. Ver estadisticas
 10. Enviar mensaje
 11. Recibir mensajes
 12. Crear semaforo
 13. Wait en semaforo
 14. Signal en semaforo
 15. Demo Productor-Consumidor
  0. Salir
============================================================
Seleccione una opcion:
```

CP-02: Inicialización del Sistema con Round Robin

Objetivo: Verificar que el sistema se inicializa correctamente con algoritmo Round Robin.

Pasos a seguir:

1. Ejecutar: ./simulador_procesos
2. Cuando pregunte por el algoritmo, seleccionar: 2 (Round Robin)
3. Cuando pida el quantum, ingresar: 3
4. Observar confirmación

Qué debe aparecer:

- Mensaje de bienvenida
- Confirmación "✓ Algoritmo Round Robin seleccionado (Quantum: 3)"
- Menú principal

📸 **CAPTURA 02: Inicialización con Round Robin**

```
PS C:\Users\RSD ESCRITORIO\Desktop\ESCRITORIO\SISTEMAS OPERATIVOS\Producto_Integrador_SO\src> ./simulador_procesos
==============================================================
  SIMULADOR DE GESTOR DE PROCESOS PARA SISTEMAS OPERATIVOS
           Universidad Autonoma de Tamaulipas
==============================================================


========== SELECCION DE ALGORITMO DE PLANIFICACION ==========
1. FCFS (First Come First Served)
2. Round Robin
Seleccione un algoritmo (1-2): 2
Ingrese el quantum: 3
Γ£ô Algoritmo Round Robin seleccionado (Quantum: 3)
==============================================================

==============================================================
     SIMULADOR DE GESTOR DE PROCESOS - MENU PRINCIPAL
==============================================================
  1. Crear proceso
  2. Listar procesos activos
  3. Mostrar recursos disponibles
  4. Ejecutar siguiente paso
  5. Suspender proceso
  6. Reanudar proceso
  7. Terminar proceso
  8. Ver logs
  9. Ver estadisticas
 10. Enviar mensaje
 11. Recibir mensajes
 12. Crear semaforo
 13. Wait en semaforo
 14. Signal en semaforo
 15. Demo Productor-Consumidor
  0. Salir
==============================================================
Seleccione una opcion: |
```

CP-03: Creación de Procesos

Objetivo: Crear múltiples procesos con diferentes parámetros.

Pasos a seguir:

1. En el menú, seleccionar opción: 1 (Crear proceso)
2. Crear **Proceso 1:**
    - Tiempo de ejecución: 10
    - Prioridad: 5
    - Bloques de memoria: 2
3. Seleccionar opción 1 nuevamente para crear **Proceso 2:**
    - Tiempo de ejecución: 5
    - Prioridad: 3
    - Bloques de memoria: 1

4. Seleccionar opción 1 nuevamente para crear **Proceso 3:**
    - Tiempo de ejecución: 8
    - Prioridad: 7
    - Bloques de memoria: 1

Qué debe aparecer:

- Tres mensajes "✓ Proceso creado exitosamente" con PIDs 1, 2 y 3

📸 **CAPTURA 03: Creación de tres procesos**

```
Seleccione una opcion: 1

--- Crear Proceso ---
Tiempo de ejecucion (burst time): 10
Prioridad (mayor numero = mayor prioridad): 5
Bloques de memoria necesarios (1 bloque = 1024MB, max 4 bloques): 2
Γ£ô Proceso creado exitosamente (PID: 1)
```

```
Seleccione una opcion: 1

--- Crear Proceso ---
Tiempo de ejecucion (burst time): 5
Prioridad (mayor numero = mayor prioridad): 3
Bloques de memoria necesarios (1 bloque = 1024MB, max 4 bloques): 1
Γ£ô Proceso creado exitosamente (PID: 2)
```

```
Seleccione una opcion: 1

--- Crear Proceso ---
Tiempo de ejecucion (burst time): 8
Prioridad (mayor numero = mayor prioridad): 7
Bloques de memoria necesarios (1 bloque = 1024MB, max 4 bloques): 1
Γ£ô Proceso creado exitosamente (PID: 3)
```

CP-04: Listar Procesos Activos

Objetivo: Verificar que los procesos se muestran correctamente en la lista.

Pasos a seguir:

1. Después de crear los 3 procesos del CP-03

2. Seleccionar opción: 2 (Listar procesos activos)

Qué debe aparecer:

- Tabla con 3 procesos
- Columnas: PID, Estado, Prioridad, Burst, Restante, Memoria, Espera
- Los 3 procesos deben estar en estado "LISTO"

📸 **CAPTURA 04: Lista de procesos**

```
Seleccione una opcion: 2

========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------------
1     LISTO      5        10         10       0         MB 0
2     LISTO      3        5          5        0         MB 0
3     LISTO      7        8          8        0         MB 0
========================================
```

CP-05: Mostrar Recursos del Sistema

Objetivo: Verificar el estado inicial de los recursos del sistema.

Pasos a seguir:

1. Después de crear procesos (pero sin ejecutar pasos)
2. Seleccionar opción: 3 (Mostrar recursos disponibles)

Qué debe aparecer:

- CPU: Total 1, Disponible 1, En uso 0
- Memoria: Total 4 bloques (4096 MB), Disponible 4, En uso 0

📸 **CAPTURA 05: Recursos del sistema**

```
Seleccione una opcion: 3

========== RECURSOS DEL SISTEMA ==========
CPU:
  Total: 1
  Disponible: 1
  En uso: 0

Memoria:
  Total: 4 bloques (4096 MB)
  Disponible: 4 bloques (4096 MB)
  En uso: 0 bloques (0 MB)
==========================================
```

CP-06: Ejecución Paso a Paso (FCFS)

Objetivo: Ejecutar procesos usando el algoritmo FCFS.

Pasos a seguir:

1. Con los 3 procesos creados (CP-03) y algoritmo FCFS
2. Seleccionar opción: 4 (Ejecutar siguiente paso) **5 veces seguidas**
3. Observar cómo ejecuta el Proceso 1

Qué debe aparecer:

- Tiempo incrementándose: 1, 2, 3, 4, 5
- Mensaje "CPU: Ejecutando PID 1" en cada paso
- CPU nunca debe estar inactiva mientras hay procesos

📸 **CAPTURA 06: Ejecución FCFS - 5 pasos**

```
Seleccione una opcion: 4

=== Tiempo: 1 ===
CPU: Ejecutando PID 1
```

```
Seleccione una opcion: 4

=== Tiempo: 2 ===
CPU: Ejecutando PID 1
```

```
Seleccione una opcion: 4

=== Tiempo: 3 ===
CPU: Ejecutando PID 1
```

```
Seleccione una opcion: 4

=== Tiempo: 4 ===
CPU: Ejecutando PID 1
```

```
Seleccione una opcion: 4

=== Tiempo: 5 ===
CPU: Ejecutando PID 1
```

CP-07: Recursos Durante Ejecución

Objetivo: Verificar que los recursos se asignan correctamente durante la ejecución.

Pasos a seguir:

1. Después de ejecutar algunos pasos (CP-06)
2. Seleccionar opción: 3 (Mostrar recursos disponibles)

Qué debe aparecer:

- CPU: En uso 1, Disponible 0
- Memoria: En uso 2 bloques (2048 MB del Proceso 1), Disponible 2

📸 **CAPTURA 07: Recursos durante ejecución**

```
Seleccione una opcion: 3

========== RECURSOS DEL SISTEMA ==========
CPU:
  Total: 1
  Disponible: 0
  En uso: 1

Memoria:
  Total: 4 bloques (4096 MB)
  Disponible: 2 bloques (2048 MB)
  En uso: 2 bloques (2048 MB)
==========================================
```

CP-08: Proceso Completado

Objetivo: Verificar que un proceso termina correctamente.

Pasos a seguir:

1. Continuar ejecutando pasos (opción 4) hasta que el Proceso 1 termine
2. Ejecutar 10 pasos en total desde el inicio
3. Luego seleccionar opción: 2 (Listar procesos)

Qué debe aparecer:

- Proceso 1 en estado "TERMINADO"
- Proceso 1 con remaining_time = 0
- Proceso 1 con memoria = 0MB (recursos liberados)
- Proceso 2 debe estar ejecutando o terminado

📸 **CAPTURA 08: Proceso terminado**

```
Seleccione una opcion: 2

========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------
1     TERMINADO  5        10         0        0         MB 1
2     EJECUTANDO 3        5          5        1024      MB 11
3     LISTO      7        8          8        0         MB 11
========================================
```

CP-09: Round Robin - Cambio de Contexto

Objetivo: Verificar que Round Robin cambia entre procesos según el quantum.

Preparación:

1. **REINICIAR EL SIMULADOR**
2. Seleccionar Round Robin con quantum = 3
3. Crear 3 procesos:
    - P1: Burst=8, Prioridad=5, Memoria=1
    - P2: Burst=4, Prioridad=3, Memoria=1
    - P3: Burst=6, Prioridad=7, Memoria=1

Pasos a seguir:

1. Ejecutar opción 4 (Ejecutar siguiente paso) **10 veces**
2. Observar cómo la CPU alterna entre procesos cada 3 unidades

Qué debe aparecer:

- Tiempo 1-3: CPU ejecutando PID 1

- Tiempo 4-6: CPU ejecutando PID 2
- Tiempo 7-9: CPU ejecutando PID 3
- Tiempo 10: CPU ejecutando PID 1 nuevamente

📸 **CAPTURA 09: Round Robin - Cambios de contexto**

```
=== Tiempo: 1 ===
CPU: Ejecutando PID 1
```

```
=== Tiempo: 2 ===
CPU: Ejecutando PID 1
```

```
=== Tiempo: 3 ===
CPU: Ejecutando PID 1
```

```
=== Tiempo: 4 ===
CPU: Ejecutando PID 2
```

```
=== Tiempo: 5 ===
CPU: Ejecutando PID 2
```

```
=== Tiempo: 6 ===
CPU: Ejecutando PID 2
```

etc

CP-10: Ver Logs del Sistema

Objetivo: Verificar que el sistema registra eventos correctamente.

Pasos a seguir:

1. Después de ejecutar varios pasos
2. Seleccionar opción: 8 (Ver logs)

Qué debe aparecer:

- Tabla con columnas: Tiempo, PID, Evento
- Logs de creación de procesos
- Logs de asignación de recursos
- Logs de ejecución ("PID X ejecutando")
- Logs de cambios de estado (si RR: "quantum agotado")

📸 **CAPTURA 10: Logs del sistema**

```
========== LOGS DEL SISTEMA ==========
Tiempo   PID    Evento
--------------------------------------------------------------------------------
11       1      Recursos asignados a PID 1 (CPU: 1, Memoria: 1 bloques = 1024MB)
11       1      Proceso PID 1 ahora en ejecucion
12       1      PID 1 ejecutando (Tiempo restante: 1)
13       1      PID 1 ejecutando (Tiempo restante: 0)
13       1      Recursos liberados de PID 1 (CPU: 1, Memoria: 1 bloques = 1024MB)
13       1      Proceso PID 1 terminado (Causa: Normal, Turnaround: 13, Espera: 7)
13       3      Recursos asignados a PID 3 (CPU: 1, Memoria: 1 bloques = 1024MB)
13       3      Proceso PID 3 ahora en ejecucion
14       3      PID 3 ejecutando (Tiempo restante: 5)
15       3      PID 3 ejecutando (Tiempo restante: 4)
16       3      PID 3 ejecutando (Tiempo restante: 3)
16       3      Recursos liberados de PID 3 (CPU: 1, Memoria: 1 bloques = 1024MB)
16       3      PID 3: quantum agotado, vuelve a READY
16       3      Recursos asignados a PID 3 (CPU: 1, Memoria: 1 bloques = 1024MB)
16       3      Proceso PID 3 ahora en ejecucion
17       3      PID 3 ejecutando (Tiempo restante: 2)
18       3      PID 3 ejecutando (Tiempo restante: 1)
19       3      PID 3 ejecutando (Tiempo restante: 0)
19       3      Recursos liberados de PID 3 (CPU: 1, Memoria: 1 bloques = 1024MB)
19       3      Proceso PID 3 terminado (Causa: Normal, Turnaround: 19, Espera: 14)
======================================
```

CP-11: Estadísticas del Sistema

Objetivo: Verificar el cálculo de estadísticas de rendimiento.

Pasos a seguir:

1. Después de que algunos procesos hayan terminado
2. Seleccionar opción: 9 (Ver estadísticas)

Qué debe aparecer:

- Tiempo total de simulación
- Procesos completados (número)
- Tiempo promedio de espera
- Tiempo promedio de retorno
- Utilización de CPU (porcentaje)
- Throughput
- Algoritmo de planificación usado

📸 **CAPTURA 11: Estadísticas**

```
Seleccione una opcion: 9

========== ESTADISTICAS DEL SISTEMA ==========
Tiempo total de simulacion: 19 unidades
Procesos completados: 3
Tiempo promedio de espera: 9.67 unidades
Tiempo promedio de retorno: 14.33 unidades
Utilizacion de CPU: 94.74%
Throughput: 0.1579 procesos/unidad de tiempo

Algoritmo de planificacion: Round Robin (Quantum: 3)
==============================================
```

CP-12: Suspender Proceso

Objetivo: Verificar la suspensión de un proceso en ejecución.

Preparación:

1. **REINICIAR EL SIMULADOR** con FCFS
2. Crear un proceso: Burst=20, Prioridad=5, Memoria=1
3. Ejecutar 5 pasos (opción 4)

Pasos a seguir:

1. Seleccionar opción: 5 (Suspender proceso)
2. Ingresar PID: 1
3. Luego seleccionar opción: 2 (Listar procesos)

Qué debe aparecer:

- Mensaje "✓ Proceso PID 1 suspendido"
- En la lista, Proceso 1 en estado "ESPERANDO"
- CPU debe quedar inactiva

📸 **CAPTURA 12: Suspender proceso**

```
Seleccione una opcion: 5

--- Suspender Proceso ---
PID del proceso a suspender: 1
Γ£ô Proceso PID 1 suspendido
```

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------------
1     ESPERANDO  5        20         17       1024       MB 1
========================================
```

CP-13: Reanudar Proceso

Objetivo: Verificar la reanudación de un proceso suspendido.

Pasos a seguir:

1. Después de suspender el proceso (CP-12)
2. Seleccionar opción: 6 (Reanudar proceso)
3. Ingresar PID: 1
4. Luego seleccionar opción: 2 (Listar procesos)

Qué debe aparecer:

- Mensaje "✓ Proceso PID 1 reanudado"
- En la lista, Proceso 1 en estado "LISTO"
- Al ejecutar siguiente paso, debe volver a ejecutar

📸 **CAPTURA 13: Reanudar proceso**

```
Seleccione una opcion: 6

--- Reanudar Proceso ---
PID del proceso a reanudar: 1
Γ£ô Proceso PID 1 reanudado
```

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------------
1     LISTO      5        20         17       1024       MB 1
========================================
```

CP-14: Terminar Proceso Forzadamente

Objetivo: Verificar la terminación manual de un proceso.

Pasos a seguir:

1. Con un proceso ejecutando
2. Seleccionar opción: 7 (Terminar proceso)
3. Ingresar PID del proceso a terminar
4. Seleccionar opción: 3 (Mostrar recursos)

Qué debe aparecer:

- Mensaje "✓ Proceso PID X terminado"
- Recursos liberados (CPU y memoria disponibles incrementan)

📸 **CAPTURA 14: Terminar proceso forzadamente**

```
Seleccione una opcion: 7

--- Terminar Proceso ---
PID del proceso a terminar: 1
Γ£ô Proceso PID 1 terminado
```

```
========== RECURSOS DEL SISTEMA ==========
CPU:
  Total: 1
  Disponible: 1
  En uso: 0

Memoria:
  Total: 4 bloques (4096 MB)
  Disponible: 4 bloques (4096 MB)
  En uso: 0 bloques (0 MB)
==========================================
```

CP-15: Enviar Mensaje Entre Procesos

Objetivo: Verificar la comunicación entre procesos mediante mensajes.

Preparación:

1. **REINICIAR EL SIMULADOR**
2. Crear dos procesos (PID 1 y PID 2)

Pasos a seguir:

1. Seleccionar opción: 10 (Enviar mensaje)
2. Ingresar:
    - PID emisor: 1
    - PID receptor: 2
    - Contenido: Hola desde proceso 1

Qué debe aparecer:

- Mensaje "✓ Mensaje enviado"

📸 **CAPTURA 15: Enviar mensaje**

```
Seleccione una opcion: 10

--- Enviar Mensaje ---
PID emisor: 1
PID receptor: 2
Contenido del mensaje: hola desde proceso 1
Γ£ô Mensaje enviado
```

CP-16: Recibir Mensajes

Objetivo: Verificar la recepción de mensajes por un proceso.

Pasos a seguir:

1. Después de enviar el mensaje (CP-15)
2. Seleccionar opción: 11 (Recibir mensajes)
3. Ingresar PID receptor: 2

Qué debe aparecer:

- Lista de mensajes para PID 2

- Mensaje "De PID 1: Hola desde proceso 1"

📸 **CAPTURA 16: Recibir mensajes**

```
Seleccione una opcion: 11

--- Recibir Mensajes ---
PID receptor: 2

=== Mensajes para PID 2 ===
De PID 1: hola desde proceso 1
============================
```

CP-17: Crear Semáforo

Objetivo: Verificar la creación de un semáforo.

Preparación:

1. **REINICIAR EL SIMULADOR**

Pasos a seguir:

1. Seleccionar opción: 12 (Crear semáforo)
2. Ingresar valor inicial: 1

Qué debe aparecer:

- Mensaje "✓ Semáforo creado con ID: 0"

📸 **CAPTURA 17: Crear semáforo**

```
Seleccione una opcion: 12

--- Crear Semaforo ---
Valor inicial: 1
Γ£ô Semaforo creado con ID: 0
```

CP-18: Operaciones Wait y Signal en Semáforo

Objetivo: Verificar las operaciones básicas de semáforos.

Preparación:

1. Tener un semáforo creado (ID: 0) con valor inicial 1
2. Crear dos procesos (PID 1 y PID 2)

Pasos a seguir:

1. Seleccionar opción: 13 (Wait en semáforo)
    - ID del semáforo: 0
    - PID del proceso: 1
2. Seleccionar opción: 13 nuevamente (Wait)
    - ID del semáforo: 0
    - PID del proceso: 2
3. Seleccionar opción: 2 (Listar procesos) - PID 2 debe estar ESPERANDO
4. Seleccionar opción: 14 (Signal en semáforo)
    - ID del semáforo: 0
5. Seleccionar opción: 2 nuevamente - PID 2 debe estar LISTO

Qué debe aparecer:

- Proceso 2 se bloquea después del segundo wait
- Proceso 2 se desbloquea después del signal

📸 **CAPTURA 18: Wait y Signal en semáforo**

```
Seleccione una opcion: 13

--- Wait en Semaforo ---
ID del semaforo: 0
PID del proceso: 1
Γ£ô Wait ejecutado
```

```
Seleccione una opcion: 13

--- Wait en Semaforo ---
ID del semaforo: 0
PID del proceso: 2
Γ£ô Wait ejecutado
```

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------
1     LISTO      10       5          5        0         MB 0
2     ESPERANDO  3        4          4        0         MB 0
========================================
```

```
Seleccione una opcion: 14

--- Signal en Semaforo ---
ID del semaforo: 0
Γ£ô Signal ejecutado
```

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
-----------------------------------------------------------------
1     LISTO      10       5          5        0         MB 0
2     LISTO      3        4          4        0         MB 0
=======================================
```

CP-19: Demostración Productor-Consumidor

Objetivo: Verificar la configuración automática del problema productor-consumidor.

Preparación:

1. **REINICIAR EL SIMULADOR**

Pasos a seguir:

1. Seleccionar opción: 15 (Demo Productor-Consumidor)
2. Observar la configuración automática
3. Seleccionar opción: 2 (Listar procesos)

Qué debe aparecer:

- Mensaje de configuración con 2 productores y 2 consumidores
- 3 semáforos creados (mutex, empty, full)
- 4 procesos creados (PIDs 1, 2, 3, 4)
- Lista mostrando los 4 procesos en estado LISTO

📸 **CAPTURA 19: Demo Productor-Consumidor**

```
Seleccione una opcion: 15

========== DEMOSTRACION PRODUCTOR-CONSUMIDOR ==========
Configuracion: 2 Productores, 2 Consumidores, Buffer de tamano 5

Semaforos creados:
  - mutex (ID: 0): control de acceso al buffer
  - empty (ID: 1): espacios vacios en buffer
  - full (ID: 2): espacios llenos en buffer
Γ£ô Proceso creado exitosamente (PID: 1)
Productor 1 creado (PID: 1)
Γ£ô Proceso creado exitosamente (PID: 2)
Productor 2 creado (PID: 2)
Γ£ô Proceso creado exitosamente (PID: 3)
Consumidor 1 creado (PID: 3)
Γ£ô Proceso creado exitosamente (PID: 4)
Consumidor 2 creado (PID: 4)

Los procesos productores y consumidores han sido creados.
Use 'Ejecutar siguiente paso' para simular la produccion/consumo.
==========================================================
```

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------------
1     LISTO      5        10         10       0         MB 0
2     LISTO      5        10         10       0         MB 0
3     LISTO      5        10         10       0         MB 0
4     LISTO      5        10         10       0         MB 0
=======================================
```

CP-20: Prevención de Deadlock

Objetivo: Verificar que el sistema previene deadlocks al denegar recursos insuficientes.

Preparación:

1. **REINICIAR EL SIMULADOR** con FCFS
2. Crear Proceso 1: Burst=10, Prioridad=5, Memoria=3 bloques
3. Crear Proceso 2: Burst=8, Prioridad=3, Memoria=2 bloques

Pasos a seguir:

1. Ejecutar 1 paso (opción 4) - Proceso 1 debe obtener recursos

2. Crea P3: burst=5, prioridad=4, memoria=2
3. Intenta crear P4: burst=6, prioridad=2, memoria=2
4. Seleccionar opción: 2 (Listar procesos)
5. Seleccionar opción: 8 (Ver logs)

Qué debe aparecer:

- P1 en EJECUTANDO con 3072MB asignados
- P2, P3, P4 en LISTO con 0MB (insuficiente memoria disponible)
- Recursos: 1 bloque disponible de 4 (cada proceso necesita 2)
- Sistema previene deadlock manteniendo procesos en cola de espera

📸 **CAPTURA 20: Prevención de deadlock**

```
========== LISTA DE PROCESOS ==========
PID   Estado     Prior.   Burst      Restante Memoria    Espera
----------------------------------------------------------------------
1     EJECUTANDO 5        10         10       3072      MB 1
2     LISTO      3        8          8        0         MB 1
3     LISTO      4        5          5        0         MB 0
4     LISTO      2        6          6        0         MB 0
========================================
```

```
========== RECURSOS DEL SISTEMA ==========
CPU:
  Total: 1
  Disponible: 0
  En uso: 1

Memoria:
  Total: 4 bloques (4096 MB)
  Disponible: 1 bloques (1024 MB)
  En uso: 3 bloques (3072 MB)
==========================================
```

```
========== LOGS DEL SISTEMA ==========
Tiempo   PID    Evento
--------------------------------------------------------------------------------
0        -1     Sistema inicializado
0        -1     Algoritmo FCFS seleccionado
0        1      Proceso PID 1 creado (Burst: 10, Prioridad: 5, Memoria: 3 bloques = 3072MB)
0        2      Proceso PID 2 creado (Burst: 8, Prioridad: 3, Memoria: 2 bloques = 2048MB)
1        1      Recursos asignados a PID 1 (CPU: 1, Memoria: 3 bloques = 3072MB)
1        1      Proceso PID 1 ahora en ejecucion
1        3      Proceso PID 3 creado (Burst: 5, Prioridad: 4, Memoria: 2 bloques = 2048MB)
1        4      Proceso PID 4 creado (Burst: 6, Prioridad: 2, Memoria: 2 bloques = 2048MB)
======================================
```

CP-21: Prueba de Límites - Memoria Excedida

Objetivo: Verificar el manejo de error al solicitar más memoria del máximo.

Preparación:

1. **REINICIAR EL SIMULADOR**

Pasos a seguir:

1. Seleccionar opción: 1 (Crear proceso)
2. Ingresar:
    - Tiempo de ejecución: 10
    - Prioridad: 5
    - Bloques de memoria: 5 (excede el máximo de 4)

Qué debe aparecer:

- Mensaje de error: "Error: Memoria solicitada excede el máximo (4 bloques de 1024MB)"

📸 **CAPTURA 21: Error de memoria excedida**

```
Seleccione una opcion: 1

--- Crear Proceso ---
Tiempo de ejecucion (burst time): 10
Prioridad (mayor numero = mayor prioridad): 5
Bloques de memoria necesarios (1 bloque = 1024MB, max 4 bloques): 5
Error: Memoria solicitada excede el maximo (4 bloques de 1024MB)
```

CP-22: Completar Todos los Procesos y Estadísticas Finales

Objetivo: Verificar estadísticas después de completar todos los procesos.

Preparación:

1. **REINICIAR EL SIMULADOR** con FCFS
2. Crear 3 procesos:
    - P1: Burst=5, Prioridad=5, Memoria=1
    - P2: Burst=3, Prioridad=3, Memoria=1
    - P3: Burst=4, Prioridad=7, Memoria=1

Pasos a seguir:

1. Ejecutar pasos (opción 4) hasta que los 3 procesos terminen (12-15 pasos)
2. Seleccionar opción: 9 (Ver estadísticas)
3. Seleccionar opción: 0 (Salir) para ver estadísticas finales

Qué debe aparecer:

- 3 procesos completados
- Estadísticas calculadas (tiempos promedio, utilización CPU, throughput)
- Mensaje de despedida con estadísticas finales

📸 **CAPTURA 22: Estadísticas finales**

```
Seleccione una opcion: 9

========== ESTADISTICAS DEL SISTEMA ==========
Tiempo total de simulacion: 13 unidades
Procesos completados: 3
Tiempo promedio de espera: 5.33 unidades
Tiempo promedio de retorno: 9.33 unidades
Utilizacion de CPU: 92.31%
Throughput: 0.2308 procesos/unidad de tiempo

Algoritmo de planificacion: FCFS (First Come First Served)
==============================================
```

```
¡Gracias por usar el simulador!
Estadisticas finales:

========== ESTADISTICAS DEL SISTEMA ==========
Tiempo total de simulacion: 13 unidades
Procesos completados: 3
Tiempo promedio de espera: 5.33 unidades
Tiempo promedio de retorno: 9.33 unidades
Utilizacion de CPU: 92.31%
Throughput: 0.2308 procesos/unidad de tiempo

Algoritmo de planificacion: FCFS (First Come First Served)
==============================================
```

Validación de Requisitos

Requisitos Funcionales

ID	Requisito	Caso de Prueba	Estado
RF-01	Crear procesos con PCB	CP-03	Pasa
RF-02	Suspender procesos	CP-12	Pasa
RF-03	Reanudar procesos	CP-13	Pasa
RF-04	Terminar procesos	CP-08, CP-14	Pasa
RF-05	Asignar recursos	CP-07	Pasa
RF-06	Liberar recursos	CP-08	Pasa
RF-07	Prevenir deadlocks	CP-20	Pasa
RF-08	Algoritmo FCFS	CP-01, CP-06	Pasa
RF-09	Algoritmo Round Robin	CP-02, CP-09	Pasa
RF-10	Enviar mensajes	CP-15	Pasa
RF-11	Recibir mensajes	CP-16	Pasa
RF-12	Crear semáforos	CP-17	Pasa
RF-13	Operación wait	CP-18	Pasa

RF-14	Operación signal	CP-18	Pasa
RF-15	Demo prod-cons	CP-19	Pasa
RF-16	Mostrar logs	CP-10	Pasa
RF-17	Calcular estadísticas	CP-11, CP-22	Pasa

Requisitos No Funcionales

ID	Requisito	Verificación	Estado
RNF-01	Interfaz CLI funcional	Todas las capturas muestran menú	Pasa
RNF-02	Compilación sin errores	CP-00	Pasa
RNF-03	Manejo de errores	CP-21	Pasa
RNF-04	Portabilidad	Compilado en: Windows 11 con GCC	Pasa

Conclusiones

Resumen de Pruebas Ejecutadas

Total de Casos de Prueba: 22
Casos Exitosos: 22
Casos Fallidos: 0
Porcentaje de Éxito: 100%

Funcionalidades Validadas

- ✓ Gestión de Procesos (crear, suspender, reanudar, terminar)
- ✓ Planificación FCFS
- ✓ Planificación Round Robin
- ✓ Asignación y liberación de recursos
- ✓ Prevención de deadlocks
- ✓ Comunicación por mensajes
- ✓ Sincronización con semáforos

- ✓ Sistema de logs
- ✓ Cálculo de estadísticas

Conclusión Final

El Simulador de Gestor de Procesos cumple exitosamente con todos los requisitos especificados. Las pruebas demuestran que el sistema es robusto, funcional y apropiado para fines educativos. El código es de alta calidad, bien documentado, y proporciona una excelente plataforma para comprender conceptos fundamentales de sistemas operativos.

Universidad Autónoma de Tamaulipas
Sistemas Operativos - 7º Semestre
Profesor: Dante Adolfo Muñoz Quintero