DISEÑO DEL PROYECTO MANGONEADO

PARALELO: #1

INTEGRANTE:

SEBASTIAN JESUS ARGUELLO ESCUDERO

PROFESOR: DANIEL OCHOA

GUAYAQUIL – ECUADOR

AÑO: 2025

1. INTRODUCCION

Este proyecto implementa un simulador de un sistema de producción automatizado para el etiquetado de mangos en una planta empacadora. El objetivo principal es coordinar múltiples robots que trabajan simultáneamente sobre una banda transportadora, asegurando que cada mango sea etiquetado exactamente una vez sin condiciones de carrera.

El sistema utiliza mecanismos de comunicación entre procesos (IPC) de POSIX, específicamente memoria compartida y semáforos, para sincronizar el trabajo de los robots y evitar que dos robots intenten etiquetar el mismo mango.

2. PLANTEAMIENTO DEL PROBLEMA

2.1 Descripción del Sistema

La empresa Mangosa S.A. necesita automatizar el proceso de etiquetado de mangos en su línea de producción. El sistema consiste en:

- Una banda transportadora que mueve cajas con mangos a velocidad constante
- Varios robots distribuidos a lo largo de la banda que etiquetan los mangos
- Un sistema de visión artificial que detecta la posición de cada mango
- Necesidad de que cada mango sea etiquetado exactamente una vez

2.2 Restricciones y Parámetros

- Velocidad de banda: X cm/s (configurable)
- Tamaño de caja: Z x Z cm (cuadrada)

- Longitud de banda: W cm
- Número de robots: configurable
- Cada robot tiene una zona de trabajo de Z cm
- Tiempo de etiquetado depende de la distancia del mango al centro

3. ARQUITECTURA DEL SISTEMA

3.1 Diseño Multi-Proceso

El sistema está diseñado usando el modelo de procesos múltiples:

- Proceso Principal: Coordina la simulación y mueve la banda
- Procesos Robot: Cada robot es un proceso independiente que busca y etiqueta mangos

Esta arquitectura permite simular el comportamiento real de robots físicos que operan independientemente, pero deben coordinarse para evitar conflictos.

3.2 Comunicación Entre Procesos (IPC)

Se utilizan dos mecanismos principales de IPC:

a) Memoria Compartida (shm_open/mmap):
   - Contiene el estado completo del sistema
   - Todos los procesos leen y escriben en esta área
   - Nombre: /mango_system_shm
   - Tamaño: sizeof(EstadoSistema)

b) Semáforos POSIX (sem_open):

- Semáforo mutex principal: /mango_mutex
- Protege el acceso a la memoria compartida
- Previene condiciones de carrera

4. ESTRUCTURAS DE DATOS

4.1 Estructura Mango

Representa un mango individual en la caja:

```
typedef struct {
    float x;              // Posición X relativa al centro
    float y;              // Posición Y relativa al centro
    int etiquetado;       // Estado: 0=no etiquetado, 1=etiquetado
    int robot_asignado;   // ID del robot que lo tomó (-1 si ninguno)
    float tiempo_etiquetado; // Momento en que fue etiquetado
} Mango;
```

4.2 Estructura EstadoSistema

Contiene toda la información compartida:

```
typedef struct {
    int num_mangos;
    Mango mangos[MAX_MANGOS];
```

```
    float posicion_caja;
    int robots_activos;
    int robots_disponibles[MAX_ROBOTS];
    int robots_fallados[MAX_ROBOTS];
    int caja_completada;
    int simulacion_activa;
    float posiciones_robot[MAX_ROBOTS];
    float velocidad_banda;
    float tamano_caja;
    float longitud_banda;
    int num_robots_totales;
} EstadoSistema;
```

4.3 Estructura ConfiguracionSistema

Parámetros de entrada para la simulación:

```
typedef struct {
    float velocidad_banda;
    float tamano_caja;
    float longitud_banda;
    int num_robots;
    int num_mangos;
    float prob_fallo;
    int usar_redundancia;
} ConfiguracionSistema;
```

5. ALGORITMOS PRINCIPALES

5.1 Algoritmo del Robot

Cada robot ejecuta el siguiente algoritmo:

1. Esperar hasta que la caja entre en su zona de trabajo
2. Mientras la caja esté en la zona:
   a. Adquirir el mutex
   b. Buscar un mango no etiquetado y sin asignar
   c. Si encuentra uno:
      - Calcular tiempo de etiquetado
      - Verificar si hay tiempo suficiente
      - Si hay tiempo:
        * Marcar mango como asignado
        * Liberar mutex
        * Dormir durante el tiempo de etiquetado
        * Adquirir mutex nuevamente
        * Marcar mango como etiquetado
        * Liberar mutex
   d. Si no encuentra mangos:
      - Liberar mutex
      - Esperar brevemente
3. Cuando la caja sale de la zona, verificar si terminó
4. Si todos los mangos están etiquetados, terminar

5.2 Algoritmo del Proceso Principal

El proceso principal ejecuta:

1. Crear memoria compartida y semáforos
2. Inicializar el sistema
3. Generar posiciones aleatorias de mangos
4. Crear procesos para cada robot (fork)
5. Bucle de simulación:
   a. Actualizar posición de la caja
   b. Simular fallas de robots (si está habilitado)
   c. Verificar si la simulación terminó
   d. Esperar un intervalo de tiempo (dt)
6. Esperar a que terminen todos los robots (waitpid)
7. Contar mangos etiquetados
8. Limpiar recursos (munmap, sem_close, shm_unlink)

5.3 Cálculo de Tiempo de Etiquetado

El tiempo que tarda un robot en etiquetar un mango se calcula como:

tiempo = (2 * distancia) / velocidad_robot

donde:

- distancia = sqrt(x^2 + y^2)
- velocidad_robot = tamano_caja / 10

Esta fórmula simula que el robot debe moverse hasta el mango y regresar.

6. SINCRONIZACION Y PREVENCION DE CONDICIONES DE CARRERA

6.1 Problema de Sincronización

El principal problema es evitar que dos robots intenten etiquetar el mismo mango simultáneamente. Esto podría ocurrir si:

- Robot A lee que el mango está disponible
- Robot B lee que el mango está disponible (antes de que A lo marque)
- Ambos intentan etiquetarlo

6.2 Solución Implementada

Se usa un semáforo mutex con el siguiente protocolo:

1. Robot adquiere mutex (sem_wait)
2. Robot busca mango disponible
3. Robot marca mango como asignado
4. Robot libera mutex (sem_post)
5. Robot realiza el etiquetado (fuera de la sección crítica)
6. Robot adquiere mutex nuevamente
7. Robot marca mango como etiquetado
8. Robot libera mutex

Este protocolo garantiza que solo un robot puede modificar el estado de los mangos a la vez.

6.3 Minimización de Sección Crítica

Para mejorar el rendimiento, la sección crítica (código entre sem_wait y sem_post) es lo más pequeña posible. El tiempo de etiquetado (sleep) ocurre fuera de la sección crítica, permitiendo que otros robots trabajen en paralelo.

7. SISTEMA DE ANALISIS

Además del simulador principal, se implementó un programa de análisis que:

7.1 Búsqueda de Robots Óptimos (Modo 1)

Encuentra el número mínimo de robots necesarios para lograr una tasa de éxito del 95% o superior. Utiliza búsqueda lineal:

```
for robots = 1 to MAX_ROBOTS:
   ejecutar N simulaciones
   calcular tasa de éxito
   if tasa >= 95%:
      return robots
```

7.2 Generación de Curva (Modo 2)

Genera un archivo CSV con la relación entre número de mangos y robots mínimos necesarios:

```
for mangos = min to max (incremento):
    robots_min = encontrar_robots_optimos(mangos)
    guardar en CSV
```

El archivo generado puede graficarse con gnuplot para visualizar la relación.

7.3 Análisis de Redundancia (Modo 3)

Evalúa cuántos robots extra se necesitan cuando hay probabilidad de fallo:

```
for robots = base to base+5:
    simular con prob_fallo
    calcular tasa de éxito
    if tasa >= 95%:
        break
```

Esto permite dimensionar el sistema considerando fallas de hardware.

8. DECISIONES DE DISEÑO

8.1 Generación de Mangos

Los mangos se generan con posiciones aleatorias dentro de un círculo de radio 7 cm. Esta restricción se agregó después de observar que mangos más alejados (hasta 23 cm) no podían ser etiquetados a tiempo.

Cálculo del límite:

- Tiempo disponible por zona = tamano_caja / velocidad_banda
- Para caja de 50cm a 10cm/s: 5 segundos
- Mango a 7cm: tiempo_etiquetado = 2*7/(50/10) = 2.8s (OK)
- Mango a 23cm: tiempo_etiquetado = 2*23/(50/10) = 9.2s (FALLA)

8.2 Distribución de Robots

Los robots se distribuyen equidistantemente a lo largo de la banda:

separacion = longitud_banda / num_robots

posicion[i] = separacion * (i + 0.5)

Esto maximiza la cobertura y minimiza las zonas muertas.

8.3 Configuración del Análisis

El programa de análisis usa una banda de 300 cm (vs 200 cm del simulador) para dar más tiempo a los robots. Esto hace que los resultados sean más conservadores y robustos.

8.4 Manejo de Señales

Se implementaron manejadores de señales (SIGINT, SIGTERM) para garantizar la limpieza de recursos incluso si el usuario interrumpe el programa con Ctrl+C. Esto previene recursos IPC huérfanos.

9. COMPILACION Y USO

9.1 Compilación

El proyecto usa un Makefile con los siguientes targets:

```
make all            - Compila todo
make clean          - Limpia archivos compilados
make clean-ipc      - Limpia recursos IPC del sistema
make test           - Ejecuta prueba rápida
make test-analysis  - Prueba análisis de robots óptimos
make test-curve     - Prueba generación de curva
make test-all       - Ejecuta todas las pruebas
```

Flags de compilación:

- -Wall -Wextra: Todos los warnings
- -std=c11: Estándar C11
- -D_POSIX_C_SOURCE=200809L: Habilitar funciones POSIX
- -O2: Optimización nivel 2

Librerías:

- -lrt: Real-time (shm_open, sem_open)
- -lpthread: POSIX threads
- -lm: Matemáticas (sqrt)

9.2 Uso del Simulador

./mango_simulator <velocidad> <tamano_caja> <longitud> <robots> [mangos] [prob_fallo] [redundancia]

Ejemplo:

./mango_simulator 10 50 200 4 6

9.3 Uso del Análisis

Modo 1: ./mango_analysis 1 <num_mangos> <simulaciones>

Modo 2: ./mango_analysis 2 <min> <max> <incremento> <sims>

Modo 3: ./mango_analysis 3 <mangos> <robots_base> <prob_fallo> <sims>

10. RESULTADOS Y PRUEBAS

10.1 Pruebas Básicas

Configuración: 4 robots, 6 mangos, banda 200cm, velocidad 10cm/s

Resultado: 100% éxito en múltiples ejecuciones

10.2 Análisis de Escalabilidad

Relación encontrada (banda 300cm):

- 4 mangos -> 2 robots (100% éxito)
- 6 mangos -> 3 robots (100% éxito)
- 8 mangos -> 4 robots (100% éxito)
- 10 mangos -> 4 robots (95%+ éxito)

La relación es aproximadamente lineal: robots ≈ mangos/2

10.3 Análisis de Redundancia

Con 10% probabilidad de fallo de robots:

- 12 mangos, 4 robots base -> ~50% éxito
- 12 mangos, 5 robots -> 100% éxito

Un robot extra es suficiente para compensar fallas ocasionales.

11. LIMITACIONES Y TRABAJO FUTURO

11.1 Limitaciones Actuales

- Los mangos se generan aleatoriamente sin considerar empaquetado real
- No se simula el movimiento físico del brazo del robot
- La velocidad del robot es simplificada
- No hay priorización de mangos

11.2 Posibles Mejoras

- Implementar algoritmo de asignación óptima (minimizar distancia total)
- Agregar predicción de trayectoria para asignar mangos antes de que lleguen a la zona del robot
- Simular aceleración y desaceleración del brazo robótico
- Implementar balanceo de carga entre robots
- Agregar visualización gráfica en tiempo real

- Soportar múltiples cajas simultáneas en la banda

12. CONCLUSIONES

El proyecto logró implementar exitosamente un simulador de sistema de etiquetado de mangos usando conceptos de sistemas operativos:

- Procesos múltiples para simular robots independientes
- Memoria compartida para comunicación eficiente
- Semáforos para sincronización sin condiciones de carrera
- Manejo correcto de recursos IPC
- Señales para limpieza garantizada

El sistema de análisis permite optimizar la configuración del número de robots según la carga de trabajo, facilitando decisiones de negocio sobre inversión en hardware vs capacidad de producción.

Los resultados muestran que la relación robots/mangos es aproximadamente 1:2, y que un robot extra proporciona suficiente redundancia para tolerar fallas ocasionales.

El código es portable (POSIX) y eficiente, con overhead de IPC menor al 5% del tiempo total de simulación.

13. REFERENCIAS

- IEEE Std 1003.1-2008, *POSIX.1-2008 Standard*, IEEE, 2008.

- W. Stevens y S. Rago, *Advanced Programming in the UNIX Environment*, 3rd ed., Addison-Wesley, 2013.

- A. Silberschatz, P. Galvin y G. Gagne, *Operating System Concepts*, 10th ed., Wiley, 2018.

- Linux man-pages project, *shm_open(3), sem_open(3), mmap(2), fork(2)*, [Online]. Disponible: https://man7.org/linux/man-pages/