Mini TP N°2
2025
Semáforos

Sistemas Operativos y Redes (SOR)

Profesor/es: Lic. Mariano Vargas

Grupo: 19

Alumnos:

- Guillermo Dominguez
- Alan Chagaray
- Braian Palenque

Universidad de General Sarmiento

Sistemas Operativos y Redes

MiniTP 2 – Semaforos

Propósito y sentido de la actividad

En este tp se practican los conceptos de sincronización y exclusión mutua entre threads. Estos conceptos son importantes para evitar errores y problemas que surgen cuando se trabaja con procesos concurrentes y variables compartidas.

Producto final de la actividad

Al finalizar esta actividad se tendra:

- Un ejemplo de un programa concurrente dividido en threads que realiza una tarea compleja.
- Sabremos compilar y ejecutar nuestro programa dentro del sistema GNU/Linux.

Evaluación y entrega:

Para acreditar esta actividad se solicita:

- El código fuente de su implementación en formato de texto plano.
- Un informe en pdf del trabajo realizado punto por punto, dificultades encontradas y soluciones propuestas.

Esta actividad es grupal, obligatoria y con nota.

Fecha de entrega: 17 de abril del 2025.

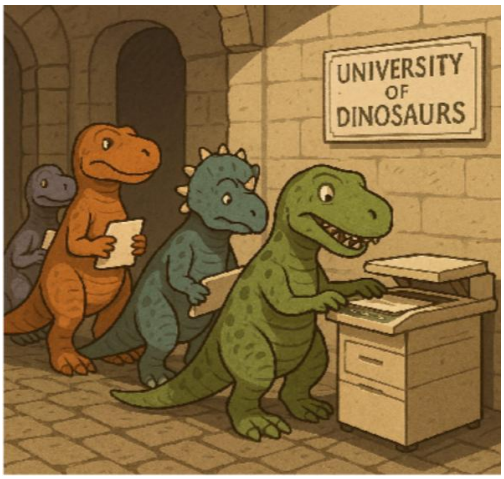
Espacio de entrega: Por Moodle.

Software necesario para el TP

Para realizar esta actividad es necesario tener instalado un sistema GNU/Linux sobre una computadora con procesador de al menos dos núcleos.

Enunciado:

Introducción: En un planeta lejano llamado Tierra, hace aproximadamente 67 millones de años, en un país conocido como Pangea, existía una prestigiosa institución llamada la Universidad de los Dinosaurios. Allí, jóvenes dinosaurios de diversas especies estudiaban carreras muy avanzadas para su época. Como en toda universidad, necesitaban sacar fotocopias de apuntes, teorías y ejercicios. Para ello contaban con una única máquina: "**la fotocopiadora rebelde**".
Esta fotocopiadora tenía un comportamiento muy particular, que causaba malestar y situaciones desafiantes entre los alumnos saurios. En este trabajo se propone analizar y programar la lógica de acceso a dicha fotocopiadora, utilizando hilos y semáforos.



La fotocopiadora puede ser utilizada por un solo dinosaurio a la vez.

Los estudiantes pueden tener uno de dos tipos de trabajos para fotocopiar:

- Tipo A: Textos teóricos (rápidos, no recalientan la máquina).
- Tipo B: Prácticas con imágenes (tardan más y recalientan la máquina).

Restricciones a tener en cuenta:

- No puede haber más de un dinosaurio usando la fotocopiadora a la vez.
- Si se hacen dos trabajos B seguidos, la máquina necesita enfriarse durante 5 segundos antes de continuar.
- Si hay muchos trabajos B en espera, se debe forzar que entre un trabajo A antes de permitir otro B, para evitar el sobrecalentamiento.
- Mientras alguien usa la fotocopiadora, ningún otro dinosaurio puede colocar su trabajo en la bandeja.

Primitivas obligatorias: Para facilitar la resolución y definir claramente las zonas críticas del problema, se incluyen las siguientes primitivas que deben ser utilizadas e implementadas:

- usar_fotocopiadora(tipo_trabajo): representa el acceso a la fotocopiadora. El argumento puede ser 'A' o 'B'. Esta función debe controlar el acceso exclusivo.
- esperar_enfriamiento(): debe ejecutarse automáticamente si hubo dos trabajos B consecutivos. Simula el tiempo de enfriamiento.
- colocar_trabajo_en_bandeja(): representa el momento en que un dinosaurio deja su trabajo en la bandeja. No debe poder ser ejecutada por más de uno al mismo tiempo.
- imprimir_estado(): función auxiliar para mostrar por consola el estado de la fotocopiadora, qué dinosaurio está usándola, tipo de trabajo y si hubo enfriamiento.

Estas primitivas ayudan a identificar las zonas críticas del problema y deben formar parte central de la solución.

RESOLUCION:

La solución implementada utiliza 3 semáforos y un contador para sincronizar el correcto acceso a recursos compartidos y la prioridad entre los tipos de trabajo, simulando el comportamiento de una fotocopiadora en un entorno con múltiples usuarios concurrentes. En nuestro programa los múltiples usuarios que intentan utilizar la fotocopiadora son 5 procesos representados por “Dinosaurio”.

Para evitar el “sobrecalentamiento” de la fotocopiadora generado al imprimir dos tareas de tipo B (con imágenes) implementamos el método esperar_enfriamiento(), el cual emula una espera de 5 segundos. En el caso que se impriman de forma consecutiva más de dos tareas de tipo B aseguramos que el próximo “dino” va a ser expropiativo para tipo A mediante el uso del semáforo prioridad_TipoA_sem.

A continuación, listo los semáforos utilizados y su finalidad:

- **fotocopiadora_sem:**
  Función: Controla el acceso exclusivo a la fotocopiadora.
  Inicialización: Se inicializa con el valor 1, lo que actúa como un mutex. Esto asegura que solo un dinosaurio puede usar la fotocopiadora a la vez.
  Uso: sem_wait se utiliza antes de que un dinosaurio use la fotocopiadora, y sem_post se utiliza después de que termina, liberando el acceso para el siguiente.
- **bandeja_sem:**
  Función: Controla el acceso exclusivo a la "bandeja" donde los dinosaurios colocan sus trabajos antes de usar la fotocopiadora.
  Inicialización: También se inicializa con el valor 1, actuando como un mutex.
  Uso: Asegura que solo un dinosaurio coloque o retire un trabajo de la bandeja a la vez, evitando condiciones de carrera al acceder a este recurso compartido.
- **prioridad_TipoA_sem**:
  Función: Controla la prioridad de los trabajos de tipo A sobre los de tipo B.
  Inicialización: Se inicializa con el valor 0. Esto significa que inicialmente está bloqueado.
  Uso: Al estar activo indica que se debe avanzar “obligatoriamente” con un trabajo de tipo A.

Por otra parte, disponemos de la variable ***trabajos_b_seguidos*** utilizada para contabilizar los trabajos de tipo B realizados en forma consecutiva. En el caso que un trabajo de tipo A sea realizado, se reinicia este contador.

Dificultades encontradas y soluciones propuestas:

Las dificultades principales que encontramos fueron:

- Comprender el enunciado y definir en consecuencia los semaforos a utilizar.
- Inanición al dar prioridad a un trabajo de tipo A

Resolvimos el problema del deadlock mediante un ciclo de 5 intentos, en los cuales realizamos un sleep() de un segundo al “dino” actual, para intentar que avance una tarea de tipo A, y, en caso que no se logre avanzar, desbloque fuerce el avance de una tarea de tipo B..

Para resolver ambas dificultades nos apoyamos mucho en el uso de la IA como tutor.

Ejecucion:

```
guille@gdz:~/SOR/2025/MiniTP2 - Semaforos$ ./fotocopiadoraRebelde
Dinosaurio 1 coloco el trabajo de tipo B en la bandeja
------------------------------
Dinosaurio usando: 1
Tipo de trabajo: B
Enfriar al finalizar: No
------------------------------
Dinosaurio 2 coloco el trabajo de tipo B en la bandeja
Dinosaurio 3 coloco el trabajo de tipo B en la bandeja
Dinosaurio 4 coloco el trabajo de tipo A en la bandeja
Dinosaurio 1 terminó de usar la fotocopiadora
------------------------------
Dinosaurio usando: 2
Tipo de trabajo: B
Enfriar al finalizar: Sí
------------------------------
Dinosaurio 5 coloco el trabajo de tipo B en la bandeja
La fotocopiadora necesita enfriarse...
La fotocopiadora está lista para usar
Dinosaurio 2 terminó de usar la fotocopiadora
------------------------------
Dinosaurio usando: 3
Tipo de trabajo: B
Enfriar al finalizar: No
------------------------------
Tiempo de espera agotado. Continuando con trabajos tipo B.
Dinosaurio 3 terminó de usar la fotocopiadora
------------------------------
Dinosaurio usando: 4
Tipo de trabajo: A
Enfriar al finalizar: No
------------------------------
Dinosaurio 4 terminó de usar la fotocopiadora
------------------------------
Dinosaurio usando: 5
Tipo de trabajo: B
Enfriar al finalizar: No
------------------------------
Dinosaurio 5 terminó de usar la fotocopiadora
Todos los dinosaurios han terminado sus trabajos.
Total de trabajos realizados: 5
Fin del programa.
```

Análisis de la ejecución:

Implementacion en C:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

// Semaforos
sem_t fotocopiadora_sem; // Controla el acceso exclusivo a la fotocopiadora
sem_t bandeja_sem; // Controla el acceso exclusivo a la bandeja
sem_t prioridad_TipoA_sem; // Controla la prioridad A sobre B

// Variables Globales
int trabajos_b_seguidos = 0;
int total_trabajos = 0;

// PRIMITIVAS (DEFINIDAS EN EL ENUNCIADO)
void esperar_enfriamiento() {
    printf("La fotocopiadora necesita enfriarse...\n");
    sleep(5);
    printf("La fotocopiadora está lista para usar\n");
}

void imprimir_estado(int id_dinosaurio, char tipo_trabajo) {
    printf("------------------------------\n");
    printf("Dinosaurio usando: %d\n", id_dinosaurio);
    printf("Tipo de trabajo: %c\n", tipo_trabajo);
    printf("Enfriar al finalizar: %s\n", trabajos_b_seguidos == 2 ? "Sí" : "No");
    printf("------------------------------\n");
}

// Simula el uso de la fotocopiadora
void usar_fotocopiadora(char tipo_trabajo, int id_dinosaurio) {
    sem_wait(&fotocopiadora_sem);

    if (tipo_trabajo == 'A') {
        trabajos_b_seguidos = 0; // Reinicia el contador de trabajos B consecutivos
        total_trabajos++;
        imprimir_estado(id_dinosaurio, tipo_trabajo);
        sleep(1); // Simula el tiempo de fotocopiado para A
        sem_post(&prioridad_TipoA_sem); // Libera el semáforo de prioridad para permitir más trabajos
    } else {
        trabajos_b_seguidos++;
        total_trabajos++;

        imprimir_estado(id_dinosaurio, tipo_trabajo);
        sleep(3); // Simula el tiempo de fotocopiado para B

     // Si hay 2 trabajos B seguidos, esperar enfriamiento}
        if(trabajos_b_seguidos == 2)
            esperar_enfriamiento();

     // Si hay más de 2 trabajos B seguidos, esperar prioridad para A
        if (trabajos_b_seguidos > 2) {
            int intentos = 5;
            while (intentos > 0) {
                if (sem_trywait(&prioridad_TipoA_sem) == 0) {
                    break; // Se desbloqueó la prioridad porque llegó un trabajo de tipo A
                }
                sleep(2); // Esperar 1 segundo antes de intentar de nuevo
                intentos--;
            }

            if (intentos == 0) {
                // Si se agotaron los intentos, desbloquear manualmente la prioridad
                printf("Tiempo de espera agotado. Continuando con trabajos tipo B.\n");
            }
        }
    }

    printf("Dinosaurio %d terminó de usar la fotocopiadora\n", id_dinosaurio);
    sem_post(&fotocopiadora_sem);
}

void colocar_trabajo_en_bandeja(char tipo_trabajo, int id_dinosaurio) {
    sem_wait(&bandeja_sem);
    printf("Dinosaurio %d coloco el trabajo de tipo %c en la bandeja\n", id_dinosaurio, tipo_trabajo);
    sleep(1);
    sem_post(&bandeja_sem);
}

// Definir el tipo de trabajo (A o B) aleatoriamente
char definir_tipo_trabajo() {
    if (rand() % 100 < 50) {
        return 'A'; //  Textos teóricos (rápidos, no recalientan la máquina).
    } else {
        return 'B'; // Prácticas con imágenes (tardan más y recalientan la máquina).
    }
}

// Funcion thread del dinosaurio
void *dinosaurio(void *arg) {
    int id_dinosaurio = *(int *)arg;
    char tipo_trabajo = definir_tipo_trabajo();

    colocar_trabajo_en_bandeja(tipo_trabajo, id_dinosaurio);
    usar_fotocopiadora(tipo_trabajo, id_dinosaurio);
    pthread_exit(NULL);
}

int main() {
    // Inicializar semáforos
    sem_init(&fotocopiadora_sem, 0, 1);
    sem_init(&bandeja_sem, 0, 1);
    sem_init(&prioridad_TipoA_sem, 0, 0);

    // Crear threads para los dinosaurios
    pthread_t threads[5];
    int ids[5] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        pthread_create(&threads[i], NULL, dinosaurio, &ids[i]);
    }

    // Esperar a que los threads terminen
    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
    }

    // Destruir semáforos
    sem_destroy(&fotocopiadora_sem);
    sem_destroy(&bandeja_sem);
    sem_destroy(&prioridad_TipoA_sem);

    printf("Todos los dinosaurios han terminado sus trabajos.\n");
    printf("Total de trabajos realizados: %d\n", total_trabajos);
    printf("Fin del programa.\n");

    return 0;
}
```