PROGRAMACIÓN DE SISTEMAS CONCURRENTES Y DISTRIBUIDOS

Dorian Boleslaw Wozniak 817570
Jaime Velasco Gimeno 816818

Práctica 4

DISEÑO:

El diseño de esta práctica en comparación a la anterior es la sustitución de los semáforos por un monitor para la gestión de la concurrencia del problema.

El monitor ControlCabinas encapsula las variables compartidas entre los procesos del problema, junto a operaciones para interactuar con dichas variables. Se compone de un entero contador dentro, que cuenta los clientes dentro de las cabinas; el vector de booleanos cabina, que indica qué cabinas están ocupadas; y el booleano limpiando, que indica si se va a proceder a limpiar las cabinas o no. Además, implementa las siguientes operaciones: entraUsuario, que una vez se cumplan las condiciones para entrar ocupa una cabina; saleUsuario, que libera la cabina que ocupaba; entraLimpieza, que avisa que va a limpiar y espera hasta que el local esté vacío; y saleLimpieza, avisando que ha dejado de limpiar.

Adicionalmente incluye dos variables condición: vacío, para los bloqueos del proceso limpieza; y acceder, para los procesos cliente.

El proceso cliente se bloqueará cuando el local esté lleno o se esté limpiando el local a la hora de evaluar, representado por la acción waitC(acceder) en entrarUsuario, y se intentará desbloquear uno si sale un cliente y no se está limpiando en saleUsuario (signalC(acceder)), o si se ha terminado la limpieza se desbloquearán todos y reevaluarán las condiciones de entrada en saleLimpieza(signalC_all(acceder)).

El proceso limpieza se bloqueará cuando el local no esté vacío al final de entraLimpieza (waitC(vacio)), y se desbloqueará una vez el último cliente salga en saleUsuario(signalC(vacio)).

Los procesos cliente y limpieza realizarán las acciones de sección crítica a través de las operaciones del monitor, y las esperas para simular ocupar la cabina o limpiar las cabinas.\

PSEUDOCÓDIGO:

```
--- CONSTANTES ---

const int N_USER = 20;             - Número de usuarios
const int N_TIMES_USER = 30;       - Número de iteraciones por usuario
const int N_TIMES_CLEANING = 5;    - Número de iteraciones de limpieza
const int PER_CLEANING = 100;      - Tiempo de espera entre limpiezas
const int N_CAB = 4;               - Número de cabinas

--- CLASE MONITOR ---

Monitor ControlCabinas
    --- VARIABLES PERMANENTES ---
    integer dentro := 0             - Cuenta cuantos clientes hay
    boolean array cabina[N_CAB]     - Cabinas físicamente ocupadas
    boolean limpiando               - True si se está limpiando el locutorio

    condition vacio, acceder        - Colas condicionales para los procesos cliente y limpieza, respectivamente

    --- OPERACIONES ---

    operation entraUsuario(integer i, j, REF integer cab)
        - Mientras no pueda acceder, se quedará en cola

        while not (not limpiando and dentro < N_CAB)
            waitC(acceder)
        end

        - Ocupa una cabina

        boolean encontrado := false

        for k := 1..N_CAB and not encontrado
            if cabina[k] := false
                cabina[k] := true
                cab := k
                dentro := dentro + 1
                encontrado := true
            end
        end
    end

    operation saleUsuario(integer i, j, cab)
        - Libera la cabina

        cabina[cab] = false
        dentro := dentro - 1

        - Si es el último en salir, desbloquea el proceso limpieza

        if dentro = 0
            signalC(vacio)
        end

        - Desbloquea el siguiente proceso cliente si no se está limpiando

        if not limpiando
            signalC(acceder)
        end
    end

    operation entraLimpieza(integer j)
        - Indica que va a limpiar

        limpiando := true

        - Se bloquea mientras el local no esté vacío.

        while(not dentro = 0)
            waitC(vacio)
        end
    end

    operation saleLimpieza(integer j)
        - Indica que ha dejado de limpiar

        limpiando := false

        - Avisa a todos los procesos cliente en cola de reevaluar condición para entrar

        signalC_all(acceder)
    end
end

--- PROCESOS ---

process usuario(i := 1..N_USER)
    integer cab

    for j := 1..N_TIMES_USER
        ControlCabinas.entraUsuario(i, j, cab) - Entra en una cabina
        espera(aleatorio(10, 40))              - Usa la cabina
        ControlCabinas.saleUsuario(i, j, cab)  - Sale de la cabina
        espera(aleatorio(20, 50))              - Espera a entrar de nuevo
    end
end

process cleaning()
    for j := 1..N_TIMES_CLEANING
        espera(PER_CLEANING)                   - Espera
        ControlCabinas.entraLimpieza(j)        - Entra a limpiar
        espera(aleatorio(80, 120))             - Limpia
        ControlCabinas.saleLimpieza(j)         - Sale de limpiar
    end
end
```