Departamento de Arquitectura

Sistemas Operativos
Práctico 5

Curso 2025

Objetivos

- Usar Ada en la solución de problemas de sincronización.
- Familiarizarse con los problemas de deadlock en la ejecución de procesos concurrentes.

Duración

- 1 semana.

Ejercicio 1 (medio) Sea una maquina de tejer, constituida por las siguientes unidades:

- **Tres expendedoras de hilo:** cuando hay hilo disponible, lo ofrecen a la unidad tejedora aceptando el encuentro quiero_hilo. El procedimiento hay_hilo indica la presencia o ausencia de hilo. Si falta hilo, se invoca a la función cambiar_rollo, que retorna cuando el operador coloque un rollo.
- **Tejedora:** pide el hilo necesario a las unidades expendedoras invocando al encuentro quiero_hilo y teje, invocando a la rutina tejer, una vez que ha obtenido el hilo de las tres unidades expendedoras.
- **Control:** verifica la producción e invoca la alarma si transcurre más de un minuto sin que la tarea tejedora esté tejiendo.

Implementar la máquina descrita utilizando Ada, de forma de que cada unidad sea una tarea. Debe proponerse una interfaz para la comunicación entre las tareas control y tejedora.

Ejercicio 2 (medio) Sea un consultorio médico atiendido por un único doctor que posee una sala de espera capaz de acomodar como máximo a 10 pacientes. Para la atención de los pacientes en la sala de espera, el doctor atiende de a una persona a la vez, dandole prioridad a los niños.

Cuando el doctor termina de atender a un paciente, deja que entre el próximo, y si no hay ninguno, se dedica a leer durante cinco minutos antes de volver a comprobar si hay alguien para atender. En caso de no haber pacientes nuevamente, repite este comportamiento.

Tenga en cuenta que en esta realidad, el paciente debe solicitar permiso al portero para entrar a la sala y al doctor para entrar al consultorio.

Se dispone de las siguientes funciones:

- **atender()**
  Invocada por el doctor para atender un paciente.
- **leer_diario()**
  Invocada por el doctor para leer el diario.

Se pide: Representar mediante tareas de ADA los pacientes, el doctor y la sala de espera.

Ejercicio 3 (openfing) Se desea modelar en ADA una heladería. En la heladería hay dos vendedores y cinco empleados que arman los helados. Los cucuruchos más grandes solo pueden ser armados por los empleados más experientes por lo que, dependiendo del helado solicitado por el cliente, no necesariamente podrá ser atendido por cualquier empleado.

Cuando llega un cliente le hace el pedido al vendedor y este consulta **simultáneamente** a los cinco empleados para ver si alguno está libre y es capaz de armar el helado solicitado (asumiremos que se

prepara un helado por cliente). El primer empleado que conteste afirmativamente será el encargado de armar el helado solicitado y entregárselo al cliente quien esperará hasta que el pedido esté pronto. A los otros empleados disponibles (en caso que los haya) se les indicará que el pedido ya fue atendido por otro de ellos. En caso de que todos estén ocupados se le indica al cliente que no se le puede vender en este momento y se retira.

Cada cierto tiempo llega un supervisor de bromatología el cual verifica que los helados no estén contaminados. Mientras el supervisor trabaja no se podrá armar ningún helado (los empleados pueden terminar de armar los que ya habían empezado antes de que llegue el supervisor). El supervisor tiene prioridad sobre los clientes. Es válido rechazar clientes nuevos mientras el inspector está trabajando.

Se dispone de los siguientes procedimientos auxiliares:

- `elegir_vendedor(): {1,2}` que ejecutado por un cliente le indica el número de vendedor con quien comunicarse.
- `que_helado(): pedido` que ejecutado por un cliente retorna un pedido con el helado que desea el cliente
- `comer_helado(helado)` ejecutado por el cliente para comer el helado
- `puedo_armar_helado(pedido): boolean` ejecutado por el empleado para ver si puede armar el helado
- `armar_helado(pedido): helado` que ejecutado por un empleado arma el helado solicitado
- `verificar_helados()` ejecutado por el inspector para verificar el estado de los helados en el local
- `otras_tareas_inspector()` ejecutado por el inspector cuando no esta inspeccionando

Se pide: Implementar en ADA las tareas vendedor, empleado, cliente e inspector. Se puede usar hasta una tarea auxiliar.

Ejercicio 4 (avanzado) Un boliche bailable dispone de un baño con capacidad para 4 hombres, y otro con capacidad para 4 para mujeres. Además, cuenta con un limpiador y un reponedor de insumos (papel higiénico, jabón y toallas).
El limpiador descansará 15 minutos entre cada limpieza de un baño, mientras que el reponedor descansará 10 minutos luego de reponer un baño. Tanto el limpiador como el reponedor necesitan materiales de limpieza y de reposición, que se encuentran en una despensa de acceso exclusivo (uno por vez).
Para llevar a cabo su tarea, ni el limpiador ni el reponedor pueden entrar al baño hasta que salgan las personas que se encontraban dentro. Además, las personas no pueden entrar a un baño si se encuentra el limpiador y/o el reponedor en él.
Una vez que el limpiador y/o el reponedor indican su voluntad de entrar al baño, las personas que estén esperando para entrar, deberán abstenerse de hacerlo hasta que el personal termine su tarea. Tenga en cuenta que el limpiador y el reponedor pueden realizar sus tareas en el mismo baño a la vez.

Se desea modelar en ADA las tareas Persona, Limpiador, Reponedor, Despensa y Baño.

Notas:

- No se podrán implementar tareas auxiliares. Considerar que no hay límite para la cantidad de personas dentro del boliche.
- No debe haber más de una entrada para la comunicación entre el limpiador y la despensa. Lo mismo para la comunicación entre el reponedor y la despensa.

Se dispone de las siguientes funciones y procedimientos:

- **que_soy(): {0, 1}**. Debe ser ejecutado por la persona. Devuelve 0 para personas de sexo masculino y 1 para personas de sexo femenino.

- **utilizar_sanitario()**. Debe ser ejecutado por una persona para utilizar el sanitario una vez entrado en el baño.
- **elegir_proximo_baño(): {1..2}**. Devuelve al azar el baño que le toca limpiar o reponer. Debe ser ejecutado por el limpiador y el reponedor.
- **calcular_insumos(): insumos**. Debe ser ejecutado por el limpiador y el reponedor para saber que insumos necesita retirar de la despensa.
- **obtener_insumos(insumos)**. Debe ser ejecutado por la despensa para retirar los insumos necesarios.
- **limpiar(cantidad_de_personas: integer)**. Debe ser ejecutado para limpiar el baño, siendo cantidad_de_personas las que entraron al baño desde la última vez que se limpió, para determinar el grado de suciedad del baño. El estudiante determinará donde ejecutar esta función (baño o limpiador).
- **reponer()**. Debe ser ejecutado para reponer los insumos del baño. El estudiante determinará donde ejecutar esta función (baño o reponedor).
- **descansar(minutos: integer)**. Ejecutado por los empleados para descansar cierta cantidad de minutos.

Ejercicio 5 (avanzado) El super carrito de panchos "El Panchero" tiene 3 vendedores y vende panchos a los que se les puede agregar un aderezo de entre 9 disponibles.
Los clientes que ingresan se forman en una única fila por orden de llegada hasta que un vendedor esté disponible y atienda al primer cliente de la fila. Dado que es un emprendimiento muy popular, la cantidad máxima de clientes no es conocida.
El carrito tiene un recipiente por cada aderezo. Los vendedores no pueden extraer aderezo a la vez del mismo recipiente.

Se pide: Modelar en Ada las tareas Cliente y Vendedor.

Se dispone de los siguientes procedimientos:

- **que_aderezo_quiero(out:aderezo)**
  Llamada por los clientes y que devuelve el aderezo que va a pedir.
- **armar_pancho(in:aderezo)**
  Llamada por los vendedores para poner el aderezo en el pancho.
- **entregar_o_recibir_pancho()**
  Deberá ser ejecutado por el cliente o el vendedor para que el cliente reciba el pancho (solamente debe ser llamada por una de las dos tareas).
- **comer_pancho()**
  Llamada por los clientes para comer el pancho que le sirvieron.

Nota: Se pueden utilizar tareas auxiliares.

Ejercicio 6 (avanzado)

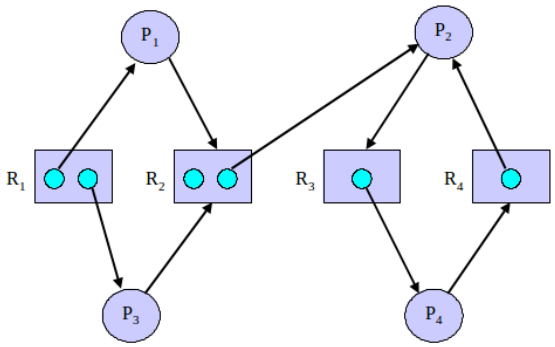
Se desea modelar un sistema de ordeño automático seis puestos para ordeñar seis vacas en forma simultánea. Los animales acceden al sistema de a uno, por orden de llegada. Se cuenta con dos empleados de mantenimiento que deben limpiar cada puesto luego de ser usado y antes de que ingrese otra vaca. Si un puesto de ordeño queda libre por más de 5 minutos (porque no llegan animales o porque los empleados de mantenimiento no llegaron a limpiarlo) se debe activar una alarma.

Implementar en ADA los procesos puesto, vaca y empleado. Se pueden usar tareas auxiliares y se dispone de los siguientes funciones:

Se dispone de los siguientes funciones auxiliares:

- `ordeñar(puesto:integer)` → Ejecutada por el puesto cuando la vaca llega para comenzar el ordeño.
- `limpiar(puesto:integer)` → Ejecutada por el empleado para limpiar el puesto.
- `alarma(puesto:integer)` → Ejecutado por el puesto para indicar que ha estado 5 minutos sin usarse.

Ejercicio 7 (básico)



Considere el grafo de recursos reusables de la figura.

(a) ¿Cuáles procesos están bloqueados?

(b) ¿Cuáles procesos están en deadlock?

(c) ¿Este estado, es un estado de deadlock?

Ejercicio 8 (básico) Considere tres procesos P_1, P_2, y P_3 ejecutando concurrentemente con la siguiente secuencia de código:

P_1	**P_2**	**P_3**
M	M	M
P(X)	P(Y)	P(Z)
M	M •	M
P(Z) •	V(Y)	P(X) •
M		M
V(X)		V(Z)
M		M
V(Z)		V(X)

El símbolo • en cada columna indica qué instrucción de cada proceso se está ejecutando actualmente. X, Y, Z son semáforos y fueron inicializados en uno.

(a) Dibuje un grafo de recurso reusable describiendo esta situación, donde cada semáforo está representado como un recurso, y P y V representan pedidos y liberaciones de recursos.

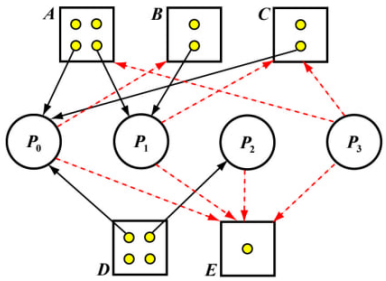
(b) Reduzca el grafo hasta donde sea posible, muestre si representa un estado de deadlock.

(c) Si Ud. pudiera aumentar la cantidad de unidades de cualquiera de los tres recursos, ¿cuál aumentaría? ¿Resolvería el deadlock?

Ejercicio 9 (medio) En el contexto de la programación concurrente:

(a) Explique si la existencia de un ciclo en el grafo de asignación-recursos implica que un sistema entrará en estado de deadlock.

(b) Determine si el sistema dado en la figura genera un deadlock o no. Las flechas negras indican asignaciones de recursos y las flechas rojas punteadas indican requerimientos de recursos. Justifique su respuesta.



Ejercicio 10 (medio) Considere un sistema con 5 procesos ejecutando y con 4 tipos de recursos. La siguiente tabla indica cuántos recursos tiene asignado cada proceso y cuál es la cantidad máxima que se necesita. Además se indica en la última columna la cantidad disponible de cada recurso.

	Asignados	Máximo	Disponible
P_0	0012	0012	1520
P_1	1000	1750	
P_2	1345	2356	
P_3	0632	0652	
P_4	0014	0656	

Conteste las siguientes preguntas utilizando el algoritmo del banquero:

(a) ¿El sistema está en un estado seguro?

(b) Si llega el pedido (0, 4, 2, 0) del proceso P_1, ¿puede satisfacerse el pedido de forma inmediata?