Departamento de Arquitectura

Sistemas Operativos
Práctico 5

Curso 2024

Objetivos

- Resolver problemas complejos con el uso de monitores.
- Resolver problemas con el uso de colas de mensajes.

Duración

- 1 semana.

Ejercicio 1 (avanzado) La piscina de un club es compartida entre nadadores amateurs y profesionales.

Por medidas de seguridad, los nadadores amateurs solo pueden nadar utilizando flotadores y, dependiendo del nadador, podrán requerir uno o dos flotadores de los 30 disponibles en el club.

Para asegurar el correcto entrenamiento de los nadadores profesionales, no se permite compartir el uso de la piscina con los amateurs. Además, el club garantiza limitar el acceso a un máximo de 10 nadadores profesionales, lo que permite un nado ágil y sin aglomeraciones.

Como parte del plan de entrenamiento, los nadadores profesionales tienen prioridad en el ingreso sobre los amateurs. Una vez que todos los profesionales hayan realizado su rutina, podrán ingresar los nadadores amateurs, respetando su orden de llegada.

Se pide: Implementar con monitores los procesos `nadador_amateur` y `nadador_profesional` según la realidad planteada. Asumiendo que:

(a) Se dispone de las siguientes funciones:
- `dar_flotador()`: invocada por los nadadores amateurs y retorna hasta un flotador. En caso de que sea invocada una segunda vez por un nadador que requiere un flotador no retornará nada. Una vez invocada queda esperando obtener un flotador.
- `nadar()`: invocada tanto por los nadadores amateurs como los profesionales para nadar.

(b) Se dispone de las siguientes funciones:
- `cuantos_flotadores()`: invocada por los nadadores amateur. Retorna la cantidad de flotadores necesarios.
- `dar_flotador()`: invocada por los nadadores amateurs y retorna un flotador.
- `nadar()`: invocada tanto por los nadadores amateurs como los profesionales para nadar.

Ejercicio 2 (avanzado) Se quiere definir un sistema que permita modelar el ingreso de docentes y estudiantes al local de una muestra de parciales. Por la situación sanitaria no pueden ingresar más de 20 personas al local. Las personas entrarán en orden de llegada al local. Pero:

1. para que ingresen estudiantes tiene que haber al menos un docente dentro del local
2. en caso de tener personas esperando fuera tendrán prioridad para el ingreso a la sala los docentes
3. en caso que un docente se quiera retirar debe considerar que mientas haya estudiantes en el salón debe quedar al menos un docente.

Implemente con monitores la realidad anterior. Dispone de las funciones:

- `Muestro_Parciales()`: Procedimiento de los docentes que modela la tarea de mostrar parciales. Una vez concluida la ejecución de la función el docente se retira del salón.

- `Reviso_Parcial()`: Procedimiento de los estudiantes, para revisar el parcial. Una vez concluida la ejecución de la función el estudiante se retira del salón.

Nota: no es necesario modelar las interacciones docentes-estudiante durante la muestra.

Ejercicio 3 (avanzado) En una planta nuclear trabajan 4 empleados encargados de controlar el núcleo del reactor. La planta cuenta con una sala de control donde hay 4 paneles que son utilizados por estos empleados para realizar los controles necesarios. Cada empleado necesita 2 paneles de forma exclusiva para realizar sus tareas. Los paneles son utilizados por cada empleado de la siguiente forma:

- El empleado 0 utiliza el panel 0 y 1.
- El empleado 1 utiliza el panel 1 y 2.
- El empleado 2 utiliza el panel 2 y 3.
- El empleado 3 utiliza el panel 3 y 0.

En la planta trabaja también un supervisor que cada cierto tiempo ingresa a la sala de control para verificar que los paneles estén funcionando correctamente. El supervisor supervisa de a dos paneles por vez. Mientras el supervisor realiza su tarea, los empleados no podrán usar los paneles que están siendo supervisados. En caso de encontrar alguna anomalía, el supervisor debe reparar los paneles. Para esto, el supervisor debe tener acceso exclusivo a la sala por lo que debe esperar a que todos los empleados terminen de trabajar y dejen la sala. El supervisor tiene prioridad sobre los empleados a la hora de realizar sus tareas.

Los empleados y el supervisor alternan sus tareas en la sala de control con otras tareas de la planta.

Se dispone de los siguientes métodos:

- `void realizar_controles(int p1, int p2)`: Ejecutado por un empleado para controlar el núcleo del reactor utilizando los paneles `p1` y `p2`.
- `int que_verif()`: Ejecutado por el supervisor para obtener un panel para verificar. Garantiza que dos invocaciones sucesivas genera dos números diferentes.
- `bool verif_paneles(int p1, int p2)`: Ejecutado por el supervisor para verificar los paneles `p1` y `p2`. Devuelve `true` si no se encuentran anomalías en los paneles y `false` en caso contrario.
- `void reparar_paneles()`: Ejecutado por el supervisor para reparar los paneles.
- `void otras_tareas()`: Ejecutado por los empleados y el supervisor para realizar otras tareas que no involucren la sala de control.

Se pide modelar empleados y supervisor utilizando monitores.

Ejercicio 4 (medio) Se desea modelar usando mailboxes la atención de un peaje de n cajas a vehículos. Los autos deberán darle al cajero el número de tarjeta de crédito y este le devolverá un número de ticket para lo cual contará con la función:

```
function pago(nro_de_tarjeta : in integer): integer
```

Los autos deberán elegir la caja con menor cantidad de autos en cola. Solo se podrá implementar una tarea auxiliar (Admin).

El programa principal ejecutará el siguiente código:

```
begin
    cobegin
        Admin
        Auto(nro_tarjeta1)
        ...
        Auto(nro_tarjetaM)
```

```
        Caja(1)
        ...
        Caja(n)
    coend
end
```

Ejercicio 5 (medio - openfing) Se desea modelar utilizando mailboxes el siguiente problema:
El taller mecánico "Boxes" tiene capacidad para 20 vehículos en sus instalaciones. La entrada de los usuarios a las instalaciones debe ser estrictamente en orden de llegada (la solución no debe permitir bajo ningún concepto "colados").
El usuario debe ser atendido por el primer mecánico libre de los 5 con que cuenta el taller. El mecánico, una vez terminado el arreglo, le indicará a la caja el monto a cobrar por el arreglo. El cliente recibirá de la caja el monto a pagar.

Se dispone de las siguientes funciones:

- `arreglar_auto():integer`
  Invocada por un mecánico para arreglar el auto. Retorna el costo del arreglo.
- `pagar_arreglo(Monto)`
  Invocada por el usuario para pagar el monto que le fue indicado por la caja.

Nota:

- Se prohíbe expresamente el busy-waiting.
- Se debe explicitar la semántica de las primitivas de mailbox utilizadas.
- Tener cuidado de que el socio pague el importe correcto del arreglo a la caja.

Ejercicio 6 (medio) Se desea tener un sistema para simplificar el orden de atención en las muestras de exámenes. Cada examen tiene un número que lo identifica y consta de tres ejercicios. Para ver los exámenes los estudiantes entran al salón en orden de llegada. El salón tiene espacio para seis estudiantes. Una vez dentro del salón los ejercicios se muestran en orden, es decir, primero se responden las dudas del ejercicio 1, luego del ejercicio 2 y por último del ejercicio 3. Los docentes que muestran cada ejercicio atenderán a los estudiantes en el orden en que solicitaron ver ese ejercicio. La cantidad de docentes no está determinada.

Si es necesario cambiar la nota del examen el docente deberá modificar la planilla de notas. Esta solo puede ser accedida por un docente a la vez.

Se pide: implementar los docentes y estudiantes usando mailboxes. Debe especificar la semántica de los mailboxes utilizados. Puede asumir que existe al menos un docente para cada ejercicio.

Dispone de los siguientes procedimientos y funciones:

- `bool quiero_ver(ex,ej)` : Función ejecutada por los estudiantes que recibe un examen `ex` y un ejercicio `ej` y retorna si el estudiante quiere que se muestre el ejercicio.
- `bool muestra(ex,ej)` : Función ejecutada por los docentes que muestra el ejercicio `ej` del examen `ex`. Retorna verdadero si es necesario modificar la nota en la planilla.
- `void modificar_planilla()` : Ejecutado por los docentes para cambiar la planilla de notas.
- `int mi_examen()` : Ejecutado por los estudiantes para obtener su número de examen.
- `int mi_ejercicio()` : Ejecutado por los docentes para ver que ejercicio contestan.

Ejercicio 7 (avanzado)

Se desea modelar el funcionamiento de una mesa de votación y su respectivo cuarto secreto para las elecciones nacionales.

Cuando un votante llega a la mesa deberá presentar su credencial la cual será validada por la mesa. Luego de ser autorizado el votante tomará un sobre de la pila y se dirigirá al cuarto secreto. Si el cuarto está ocupado deberá esperar a que se libere.

Una vez en el cuarto secreto, el votante pondrá la hoja de votación en el sobre. Cuando esta tarea finalice retornará a la mesa para depositar el voto. La mesa puede atender más votantes mientras el cuarto secreto esté ocupado. En el momento que el votante salga del cuarto secreto a depositar su voto puede tener que esperar a que la mesa termine la atención de otro votante que se encuentra en la mesa haciendo los trámites previos a la entrada al cuarto secreto, pero tendrá prioridad frente a los demás votantes y delegados esperando que la mesa los atienda.

Cada cierto tiempo, los delegados de los diferentes partidos llegan a la mesa de votación y solicitan acceder al cuarto secreto para reponer listas. Un delegado partidario debe presentar a la mesa su identificación de delegado. Los delegados harán cola para ser atendidos por la mesa junto a los votantes esperando para presentar su credencial. Ambos serán atendidos por orden de llegada.

En ningún momento podrán haber colados en las filas de la mesa de votación y del cuarto secreto. Se puede asumir que nunca se acaban las listas en el cuarto secreto. Si el documento que presenta no es válido la persona se retira inmediatamente.

Se pide: modelar utilizando mailboxes, los procesos Mesa, Votante y Delegado.

Se dispone de los siguientes procedimientos auxiliares:

- `obtener_documento():Documento` Ejecutado por los votantes y delegados para obtener su documento de identificación
- `validar_documento(Documento):boolean` Ejecutado por la mesa para validar la credencial del votante o la identificación del delegado
- `tomar_sobre():Sobre` Ejecutado por los votantes para obtener un sobre
- `poner_lista(Sobre):Sobre` Ejecutado por los votantes para poner una lista en el sobre
- `votar(Sobre)` Ejecutado por los votantes para poner el sobre en la urna
- `recargar()` Ejecutado por los delegados para recargar las listas de votación

Ejercicio 8 (avanzado) Con motivo de las fiestas, Papá Noel y los duendes se disponen a entregar regalos a los niños. Los niños eligen el regalo que quieren y mandan una carta al polo norte. En el polo norte trabajan 6 duendes. La carta se asigna a un duende que este esperando para que esté confeccione el regalo. Si no hay duendes esperando cuando se recibe la carta, se descarta la misma y el niño se queda sin regalo.

Los regalos se van agregando a una bolsa. La bolsa solo puede ser accedida por un duende a la vez. Cuando la misma alcanza 100 regalos, se debe despertar a Papá Noel para que reparta los mismos. Dependiendo de que tan cansado esté, Papá Noel puede solicitar la ayuda de uno de los duendes. Mientras Papá Noel reparte los regalos, los duendes pueden seguir trabajando.

Se pide implementar, usando mailboxes, los procesos `Niño`, `PapaNoel` y `Duende`. Se pueden usar procesos auxiliares.

Dispone de los siguientes procedimientos auxiliares:

- `escribir_carta(): Carta`. Ejecutado por un niño para escribir la carta a enviar al servidor.
- `confeccionar_regalo(c: Carta): Regalo`. Ejecutado por un duende para crear un regalo.
- `repartir_regalos(regalos: array[100] of Regalo)`. Ejecutado por Papá Noel para repartir los regalos. Recibe un arreglo con 100 regalos.

- `estoy_cansado(): Boolean`. Ejecutada por Papá Noel. Retorna True si necesita la ayuda de un duende.
- `ayudar()`: Ejecutada por un duende para ayudar a Papá Noel. Puede asumir que esta función termina al mismo tiempo que `repartir_regalos`.
- `recibir_regalo()`. Ejecutada por el Niño. Esta función se bloquea y termina cuando Papá Noel le entrega el regalo.

Ejercicio 9 (avanzado) Se desea modelar una ferretería que cuenta con: 6 empleados que atienden a los clientes que llegan, un depósito y una única caja con un POS operado por el mismo empleado que atiende al cliente (solo se puede cobrar a un cliente por vez). Los empleados reciben los pedidos de los clientes, los retiran del depósito y luego los cobran en la caja. En la ferretería puede haber a lo sumo 5 clientes. Cuando está llena los clientes que llegan esperan afuera.

En caso de que el producto a retirar sea muy pesado, el empleado necesitará la ayuda de otro empleado para que lo ayude a cargarlo. Para esto recurrirá a uno de los empleados que se encuentre libre. En caso de no haber ninguno esperará hasta que se libere alguno. Los empleados que terminan de cobrar deben darle prioridad a ayudar a los que ya tomaron pedidos en lugar de tomar ellos pedidos nuevos.

Se pide implementar, usando mailboxes, los procesos `empleado` y `cliente`. No se pueden usar procesos auxiliares.

Dispone de los siguientes procedimientos auxiliares:

- `obtener_pedido(): Pedido`. Ejecutado por los clientes para hacer un pedido.
- `es_pesado(p: Pedido): boolean`. Ejecutado por los empleados para saber si precisan ayuda.
- `obtener_artículo(p: Pedido)`. Ejecutado por los empleados para sacar los artículos pedidos del depósito. Si se necesita ayuda no se podrá ejecutar hasta conseguir un ayudante.
- `ayudar()`. Ejecutado por los empleados para ayudar a otro. Esta función termina junto con `obtener_artículo` cuando se termina el trabajo.
- `pagar()`. Ejecutado por los clientes para pagar el pedido.