Problemas de sincronización

Sistemas Operativos

Franco Frizzo

11 de abril de 2017

1. Introducción y repaso

El objetivo de esta clase es aprender a resolver **ejercicios largos** de sincronización, también conocidos como "*cuentitos*". Pondremos el foco en la forma de encarar este tipo de ejercicios, en algunos patrones de sincronización típicos que aparecen en ellos, y en la comparación entre el uso de distintas primitivas de sincronización en su resolución.

Contamos con las siguientes herramientas:

- **Objetos atómicos** (funcionan en modo *usuario*).
  - Operaciones comunes:
    - `atomic T get()`.
    - `atomic void set(T newValue)`.
    - `atomic T compareAndSwap(T testValue, T newValue)`.
  - Booleanos atómicos (`atomic<bool>`). Operaciones:
    - `atomic bool getAndSet(bool newValue)`.
    - `atomic bool testAndSet()`.
  - Enteros atómicos (`atomic<int>`). Operaciones:
    - `atomic int getAndInc()`.
    - `atomic int getAndDec()`.
    - `atomic int getAndAdd(int value)`.
- **Spinlocks** (también funcionan en modo *usuario*).
  - Basados en `testAndSet()`: `TASLock`.
  - Generan **espera activa**.
  - Operaciones:
    - `void lock()`.
    - `void unlock()`.
- **Semáforos** (requieren una llamada al *kernel*).
  - Almacenan un valor entero (capacidad).
  - Operaciones:
    - `void wait()`.
    - `void signal()`.
  - ¿Garantizan orden FIFO? En general, no.

2. Enunciado[1]

La CNRT recibió una denuncia reclamando que muchas líneas de colectivos no recogen a los pasajeros. Ellos sospechan que, al no haber suficientes colectivos, estos se llenan muy rápidamente. Debido a esto, pidieron construir un simulador que comprenda a los actores e interacciones involucradas.

El simulador debe contar con dos tipos de procesos, **Colectivero** y **Pasajero**. Los colectivos realizan un recorrido cíclico, donde cada parada está representada por un número. Hay N paradas y M colectivos.

Cada pasajero comienza esperando en una parada (que recibe por parámetro) detrás de las personas que ya se encontraban en ella (de haberlas). Una vez que el colectivo llega y el pasajero logra subir, le indica

[1] Ejercicio 24 de la práctica 3.

su destino al colectivero, con la función `indicarDestino()`. Esta función devuelve el número de colectivo. Luego espera que el colectivero haga `marcarTarifa()` y, finalmente, el pasajero ejecuta `pagarConSUBE()`.

Después de pagar, el pasajero procede a `viajar()`, y cuando termina, se dispone a bajar del colectivo. Para ello, efectúa `dirigirseAPuertaTrasera()` y una vez que el colectivo se detiene, los pasajeros que estan agrupados en la puerta trasera realizan `bajar()` de a uno por vez, sin importar el orden.

Por su parte, el colectivero recibe como parámetro la capacidad (cantidad máxima de pasajeros) del colectivo, y el identificador del colectivo (entre 0 y $M - 1$). El colectivo comienza su recorrido desde la parada número 0, e inicialmente está vacío.

Al llegar a una parada, el colectivero se detiene con `detener()`. Si hay pasajeros esperando para bajar, este abre su puerta trasera para indicar que ya pueden hacerlo (`abrirPuertaTrasera()`). Mientras esto sucede, abre la puerta delantera (`abrirPuertaDelantera()`) y, si hay pasajeros en la parada, estos comienzan a ascender en orden, siempre y cuando haya capacidad.

Las personas proceden a subir y el colectivero, amablemente, los atiende de a uno marcando en la máquina con `marcarTarifa()`. Ningún pasajero puede `indicarDestino()` antes de que el anterior haya terminado de `pagarConSUBE()`. Si no hay más pasajeros para subir o se llegó al límite de capacidad, el colectivero no duda en `cerrarPuertaDelantera()`, impidiendo que el resto de las personas en la parada ascienda.

Una vez que los pasajeros terminan de ascender, el colectivero espera a que terminen de descender todos los pasajeros que así lo desean, y procede a `cerrarPuertaTrasera()` y `avanzar()` hacia la siguiente parada, donde la dinámica será la misma.

3. Resolución

3.1. Punteo del enunciado

Pasajero

- Voy al final de la fila de la parada.
- Espero hasta poder subir a un colectivo.
- Indico el destino.
- Espero a que me marquen la tarifa.
- Pago con la SUBE.
- Viajo.
- Voy a la puerta trasera.
- Indico que quiero bajar.
- Espero a que me indiquen que baje.
- Bajo del colectivo.
- Si hay alguien más esperando bajar...
  - Le indico que baje.

Colectivero

- Llego a la parada.
- Si hay gente que quiere bajar...
  - Abro la puerta trasera.
  - Le indico a uno de ellos que baje.
- Abro la puerta delantera.
- Mientras haya pasajeros esperando y el colectivo tenga capacidad...
  - Dejo subir a un pasajero.
  - Espero a que me indique su destino.
  - Le marco la tarifa.
  - Espero a que pague.
- Cierro la puerta delantera.
- Si está abierta la puerta trasera...
  - Espero a que todos bajen.
  - Cierro la puerta trasera.
- Continúo el viaje.

3.2. Código

Variables globales

```
Queue<Semaphore> parada[N];
TASLock          lockParada[N];

Semaphore        destino[M] = {0, ..., 0};
Semaphore        tarifa[M]  = {0, ..., 0};
Semaphore        pago[M]    = {0, ..., 0};

atomic<int>      capacidad[M];

Semaphore        puertaTrasera[M]      = {0, ..., 0};
atomic<int>      esperandoParaBajar[M] = {0, ..., 0};
```

Pasajero

```
process Pasajero(paradaInicial) {
  Semaphore yo = 0;

  // Me pongo al final de la fila
  lockParada[paradaInicial].lock();
  parada[paradaInicial].push(yo);
  lockParada[paradaInicial].unlock();

  yo.wait();  // Espero que llegue el colectivo

  // Subo al colectivo
  int idColectivo = indicarDestino();
  destino[idColectivo].signal();
  tarifa[idColectivo].wait();
  pagarConSube();
  pago[idColectivo].signal();

  viajar();

  // Indico que quiero bajar
  dirigirseAPuertaTrasera();
  esperandoParaBajar[idColectivo].getAndInc();

  // Espero que me indiquen que baje
  puertaTrasera[idColectivo].wait();
  bajar();
  capacidad[idColectivo].getAndInc();

  // Si queda gente que quiere bajar, le indico al siguiente que baje
  if (esperandoParaBajar[idColectivo].getAndDec() > 1) {
    puertaTrasera[idColectivo].signal();
  }
}
```

Colectivero

```
process Colectivero(capacidadInicial, idColectivo) {
  int  paradaActual         = 0;
  bool puertaTraseraAbierta = false;

  capacidad[idColectivo].set(capacidadInicial);

  while (true) {
    detener();

    // Si hay gente que quiere bajar, abro la puerta trasera
    if (esperandoParaBajar[idColectivo].get() > 0) {
      abrirPuertaTrasera();
      puertaTraseraAbierta = true;
      puertaTrasera[idColectivo].signal();
    }

    abrirPuertaDelantera();

    lockParada[paradaActual].lock();
    // Mientras tenga capacidad y haya gente esperando...
    while (capacidad[idColectivo].get() > 0 && ! parada[paradaActual].empty()) {
      pasajero = parada[paradaActual].pop();
      lockParada[paradaActual].unlock();

      // Le indico al siguiente pasajero que suba
      capacidad.getAndDec();
      pasajero.signal();

      // Protocolo de pago del boleto
      destino[idColectivo].wait();
      marcarTarifa();
      tarifa[idColectivo].signal();
      pago[idColectivo].wait();

      lockParada[paradaActual].lock();
    }
    lockParada[paradaActual].unlock();

    cerrarPuertaDelantera();

    // Espero a que hayan bajado todos para cerrar la puerta trasera
    if (puertaTraseraAbierta) {
      while (esperandoParaBajar.get() > 0) {}
      cerrarPuertaTrasera();
      puertaTraseraAbierta = false;
    }

    // Seguimos viaje
    avanzar();
    paradaActual = (paradaActual + 1) % N;
  }
}
```