Ejercicios de sincronización entre procesos

Sistemas Operativos

30 de Agosto de 2016

1. Ejercicio 1

1.1. Enunciado

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. Cada parada está representada por su número en el recorrido. Hay N paradas y M colectivos.
Cada pasajero comienza esperando en una parada (la cual 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 pide al colectivero que le marque la tarifa con la función `pedirleBoleto()`. Esta función devuelve el número de colectivo. Luego espera que el colectivero haga `marcarTarifa()` y finalmente el pasajero ejecuta `pagarConSUBE()` (se asume que todos los pasajeros tienen SUBE). Luego de pagar, procede a `viajar()`. Cuando el pasajero termina de `viajar()`, 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.
El colectivero recibe como parámetro la capacidad (cantidad de de pasajeros) del colectivo, que, ni bien comienza, está vacío y el identificador del colectivo (entre 0 y M).
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, si hay pasajeros en la parada, abre la puerta delantera (`abrirPuertaDelantera()`) y las personas que esperaban en la parada comienzan a ascender en orden siempre y cuando haya capacidad (notar que a esta altura ya se sabe cuántos pasajeros van a bajar).
Las personas proceden a subir y el colectivero, amablemente, los atiende de a uno marcando en la máquina con `marcarTarifa()`, pero nunca lo hace 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, este espera a que terminen de descender todos los pasajeros que así lo desean, procede a `cerrarPuertaTrasera()` y `avanzar()` nuevamente a la siguiente parada, donde la dinámica será la misma.
Puede asumir:

- El colectivero se detiene en todas las paradas y abre la puerta delantera, sin importar si hay pasajeros esperando para bajar o subir.
- El cálculo de la cantidad máxima de pasajeros que el colectivero deja subir puede realizarse considerando la cantidad de pasajeros que ya se sabe que descenderán (no es un problema que brevemente se vea superada la capacidad del colectivo mientras se espera que terminen de descender).

1.2. Solución

```
struct FIFO{
    mutex m = mutex(1);
    queue<mutex> cola;
    int signals = 1;
    wait() {
        m.wait();
        if (signals > 0) {
            signals--;
            m.signal();
        }
        else {
            yo = mutex(0);
            cola.push(yo);
            m.signal();
            yo.wait();
        }
    }
    signal() {
        m.wait();
        if(cola.emtpy()) {
            signals++;
        }
        else {
          cola.pop().signal();
        }
        m.signal();
    }
    bool is_empty() {
        m.wait();
        bool ret = cola.empty();
        m.signal();
        return ret;
    }
};


// Variables Globales
FIFO cola[N] = N * FIFO();
int se_bajan[M] = M * 0;
mutex mutex_bajan[M] = M * mutex(1);
mutex boleto[M] = M * mutex(0);
mutex tarifa[M] = M * mutex(0);
mutex sube[M] = M * mutex(0);
mutex mutex_paradas[N] = N * mutex(1);
semaphore bajense[M] = M * semaphore(0);
semaphore yabaje[M] = M * semaphore(0);

Pasajero(p):
    cola[p].wait();
    int c = pedir_boleto();
    boleto[c].signal();
    tarifa[c].wait();
```

```
    pagar_con_sube();
    sube[c].signal();
    viajar();
    dirigirse_a_puerta_trasera();
    mutex_bajan[c].wait();
        se_bajan[c]++;
    mutex_bajan[c].signal();
    bajense[c].wait();
    bajar();
    mutex_bajan[c].wait();
        se_bajan[c]--;
    mutex_bajan[c].signal();
    yabaje[c].signal();

Colectivero(cap_total, id):
    int parada = 0;
    int capacidad = cap_total;
    while(true) {
        mutex_bajan[id].wait();
            int b = se_bajan[id];
        mutex_bajan[id].signal();
        if(b>0) {
            capacidad += b;
            abrir_puerta_trasera();
            while(b>0) {
                bajense[id].signal();
                ya_baje[id].wait();
                b--;
            }
            cerrar_puerta_trasera();
        }
        abrir_puerta_delantera();
        mutex_paradas[parada].wait();
            while(capacidad>0) {
                if(cola[parada].empty())
                    break;
                capacidad--;
                cola[parada].signal();
                boleto[id].wait();
                marcar_tarifa();
                tarifa[id].signal();
                sube[id].wait();
            }
        mutex_paradas[parada].signal();
        cerrar_puerta_delantera();
        avanzar();
        parada = (parada+1)%N;
        detener();
    }
```

2. Ejercicio 2

2.1. Enunciado

Desde la AFA, nos pidieron que organicemos la venta de entradas para los partidos de las eliminatorias. Para eso, nos encargaron simular la misma mediante el uso de semáforos.
Para modelar la venta de entradas contaremos con 3 tipos de procesos: N **Boletero(i)**s (con $0 \leq i < N$), 1 **Organizador()** que se encargará de organizar a la gente en la cola para las entradas, e infinitos **Fanático()**s que intentarán comprar entradas.
Al comenzar la venta, los fanáticos se amontonan contra las vallas para comprar entradas. El organizador los irá distribuyendo, sin ningún orden particular pero de a uno por vez, asignándoles una boletería para que compren su entrada. Cada boletería cuenta con una cola de hasta P personas. Una vez que estas colas estén llenas, el organizador deberá esperar a que algún boletero venda una entrada para asignar la próxima persona a la boletería que corresponda. Además, siempre que haya personas detrás de las vallas y haya al menos una boletería con menos de P personas, el organizador podrá mandar a alguna persona del tumulto a cualquier boletería con capacidad disponible.
Una vez comenzada la venta, el organizador deberá **pedirDNI()** a cada fanático antes de hacerlo pasar a la boletería, el cual deberá mostrarle el DNI al grito de **VamosArgentina()**. Una vez que el fanático le mostró el DNI al organizador y alentó a la selección, el organizador deberá avisarle que **PasePorLaBoletería(i)** para que el fanático pase por la boletería número i. Una vez en la cola de la boletería, si hay otra persona que no sea la que está comprando la entrada en ese momento, el fanático deberá preguntarle al fanático delante de él si **HoyHaceUnGolLaPulga()**, aunque el fanático de adelante está tan concentrado en que no se le colen que no le responderá.
Una vez que un fanático esté primero en la fila, el boletero le avisará al grito de **ElQueSigue()** para que éste le pida **UnaEntradaPorFavor()**. Luego el boletero procederá a **Cobrarle()** para que el fanático, en el medio de la emoción del partido, le diga **QuedateConElVuelto()**. Una vez dicho esto, el fanático se retira para que el boletero pueda llamar al siguiente para que compre su entrada.

2.2. Solución

Variables Globales:

```
cola<Sem> filaBoleteria[N]();
hayAlguien = Sem(0);
vallas = Sem(0);
esperaBoleteria = mutex(0);
mutexBoleteria = mutex[N](1); //protejo las filas de cada boleteria
int boleteria = 0;  //boleteria asignada
entrada = mutex(0); //mutex de sincronizacion
asignado = mutex(0);//mutex de sincronizacion
boleteriaDisponible = Sem(N*P);
alguienEnBoleteria = Sem[N](0);
```

Organizador:

```
while(1) {
    hayAlguien.wait();
    pedirDNI();
    vallas.signal();
    boleteriaDisponible.wait();
    for (int i=0; i<N; i++) {
        mutexBoleteria[i].wait();
        if (filaBoleteria[i].len < P) {
            mutexBoleteria[i].signal();
            pasePorLaBoleteria(i);
            boleteria = i;
```

```
            esperaBoleteria.signal();
            asignado.wait();
            break;
        } else {
            mutexBoleteria[i].signal();
        }
    }
}
```

Fanático:

```
hayAlguien.signal();
vallas.wait();
vamosArgentina();
esperaBoleteria.wait();
yo = mutex(0);
mutexBoleteria[boleteria].wait()
filaBoleteria[boleteria].push(yo);
if (filaBoleteria[boleteria].len > 1) {
    hoyHaceUnGolLaPulga();
}
mutexBoleteria[boleteria].signal();
alguienEnBoleteria[boleteria].signal();
asignado.signal();
yo.wait();
unaEntradaPorFavor();
entrada.signal();
yo.wait();
quedateConElVuelto();
```

Boletero(i):

```
while(1) {
    alguienEnBoleteria[i].wait();
    mutexBoleteria[i].wait();
        fan = filaBoleteria[i].pop();
    mutexBoleteria[i].signal();
    boleteriaDisponible.signal();
    elQueSigue();
    fan.signal();
    entrada.wait();
    cobrarle()
    fan.signal();
}
```