Ejercicio de sincronización entre procesos

Sistemas Operativos

03 de Septiembre de 2015

1. Ejercicio

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