

Clase Auxiliar V

Prof: L. Mateu

Aux: M. Leyton

31 de agosto de 2004

1. Transbordador

La municipalidad de Geek Island posee una barcaza para el transbordo de vehículos desde y hacia el continente. La barcaza tiene capacidad para llevar un solo vehículo. Los siguientes procedimientos describen la rutina de los isleños que viven en la isla y los turistas que la visitan:

```
int isleno(Auto a) {      | int turista(Auto a) {
  for(;;) {                | for (;;) {
    cruzarAContinente(a);   |   cruzarAIsla(a);
    trabajar();             |   turistear();
    cruzarAIsla(a);         |   cruzarAContinente(a);
    dormir();               |   dormir();
  }                         | }
}                           | }
```

En donde trabajar, turistear y dormir son procedimientos dados. Para coordinar el uso adecuado de la barcaza, actualmente se usa la siguiente implementación de `cruzarAContinente` y `cruzarAIsla`:

```
nSem semBarcaza; /* nMakeSem(1); en nMain */
int enContinente=TRUE;
Transbordador t;

void cruzarAIsla(Auto a) {      | void cruzarAContinente(Auto a);
  nWaitSem(semBarcaza);        |   nWaitSem(semBarcaza);
  if (!enContinente)           |   if (enContinente)
    navegarAContinente(NULL,t); |     navegarAIsla(NULL, t);
  navegarAIsla(a, t);          |   navegarAContinente(a, t);
  enContinente=FALSE;          |   enContinente=TRUE;
  nSignalSem(semBarcaza);      |   nSignalSem(semBarcaza);
}                               | }
```

En donde `navegarAContinente` y `navegarAIsla` son procedimientos dados y se demoran un tiempo considerable. Esta implementación no es eficiente porque puede ocurrir que habiendo un vehículo en el continente, la barcaza navegue vacía a la isla (y viceversa). No es de extraña entonces que se formen filas de vehículos en espera de la barcaza en ambas orillas.

Utilizando el esquema cliente/servidor y el sistema de mensajes de `nSystem`, escriba una nueva implementación de `cruzarAIsla` y `cruzarAContinente` que permita aprovechar eficientemente la barcaza. Observe que:

- La barcaza no puede navegar dos veces seguidas en el mismo sentido.
- Un usuario que llama a `cruzarA...` no puede continuar mientras que su vehículo no haya sido transportado y la barcaza no haya llegado la otra orilla.
- La barcaza no puede transportar un vehículo si éste no llegó a la orilla de partida.
- En cada viaje, si existen vehículos en la orilla, la barcaza debe transportar el vehículo que llegó primero a la orilla.
- Ud. no debe detener la barcaza si hay vehículos esperando en alguna de las orillas.
- Ud. sí debe detener la barcaza si no hay ningún vehículo que transportar en ninguna de las dos orillas.

2. Solución Transbordador

```
enum{ENISLA, ENCONT, AISLA, ACONT};
nTask trans;
```

```
typedef struct Msg{
    Auto a;
    int sentido;
    nTask emisor;
} Msg;
```

```
Msg* makeMsg(Auto a, int sentido ){
    Msg *m= nMalloc(sizeof(Msg));

    m->a=a;
    m->sentido=sentido;
    m->emisor=nCurrentTask();
    return m;
}
```

```
void initTrans(){
    trans=nEmitTask(transMain);
}
```

```
void cruzarAIsla(Auto a){

    Msg *m=makeMsg(a, AISLA);
    nSend(trans, m);
}
```

```
void cruzarAContinente(Auto a){

    Msg *m=makeMsg(a, ACONT);
    nSend(trans, m);
}
```

```
void add2Queue(Msg *msg, FifoQueue aislaq,
               FifoQueue acontq){
    if(msg->sentido==AISLA)
        PutObj(aislaq, (void *) msg);
    else
        PutObj(acontq, (void *) msg);
}
```

```
void transMain(){

    FifoQueue aislaq = MakeFifoQueue();
    FifoQueue acontq = MakeFifoQueue();

    nTask emisor; //Esta variable no es necesaria
    int lugar=ENCONT; //Elegido arbitrariamente

    while(TRUE){
        Msg* msg;
        if(EmptyFifoQueue(aislaq) &&
           EmptyFifoQueue(acontq) ){

            //Esperamos por un pasajero
            msg=(Msg *)nReceive(&emisor,-1);

            //Guardamos msg en la cola respectiva
```

```
        add2Queue(msg, aislaq, acontq);
    }
```

```
//Aqui sabemos que al menos una de
//las colas tiene un auto
if( lugar == ENISLA ){

    //Si alguien de la isla quiere ir al
    //continete lo llevamos
    if(!EmptyFifoQueue(acontq)){
        msg=(Msg*)GetObj(contq);
    }
    else { //!EmptyFifoQueue(islaq)
        //vemos si algun auto llego tarde
        msg=(Msg *)nReceive(&emisor,0);
        while(msg!=NULL){
            if(msg->sentido!=ACONT)
                add2Queue(msg, aislaq, acontq);
            else
                break;

            msg=(Msg *)nReceive(&emisor,0);
        }
    }
```

```
//Ojo que msg puede ser nulo aqui
//esto deberia ser verificado.
//(Esta simplificado por claridad)
navegarAContinente(msg->a,t);
nReply(msg->emisor,1);
lugar=ENCONT;
}
else {

    //EL CASO EN CONTINENTE ES SIMETRICO
    //QUEDA PROPUESTO COMO EJERCICIO
```

```
    }//if
    }//while(TRUE)
} //metodo
```