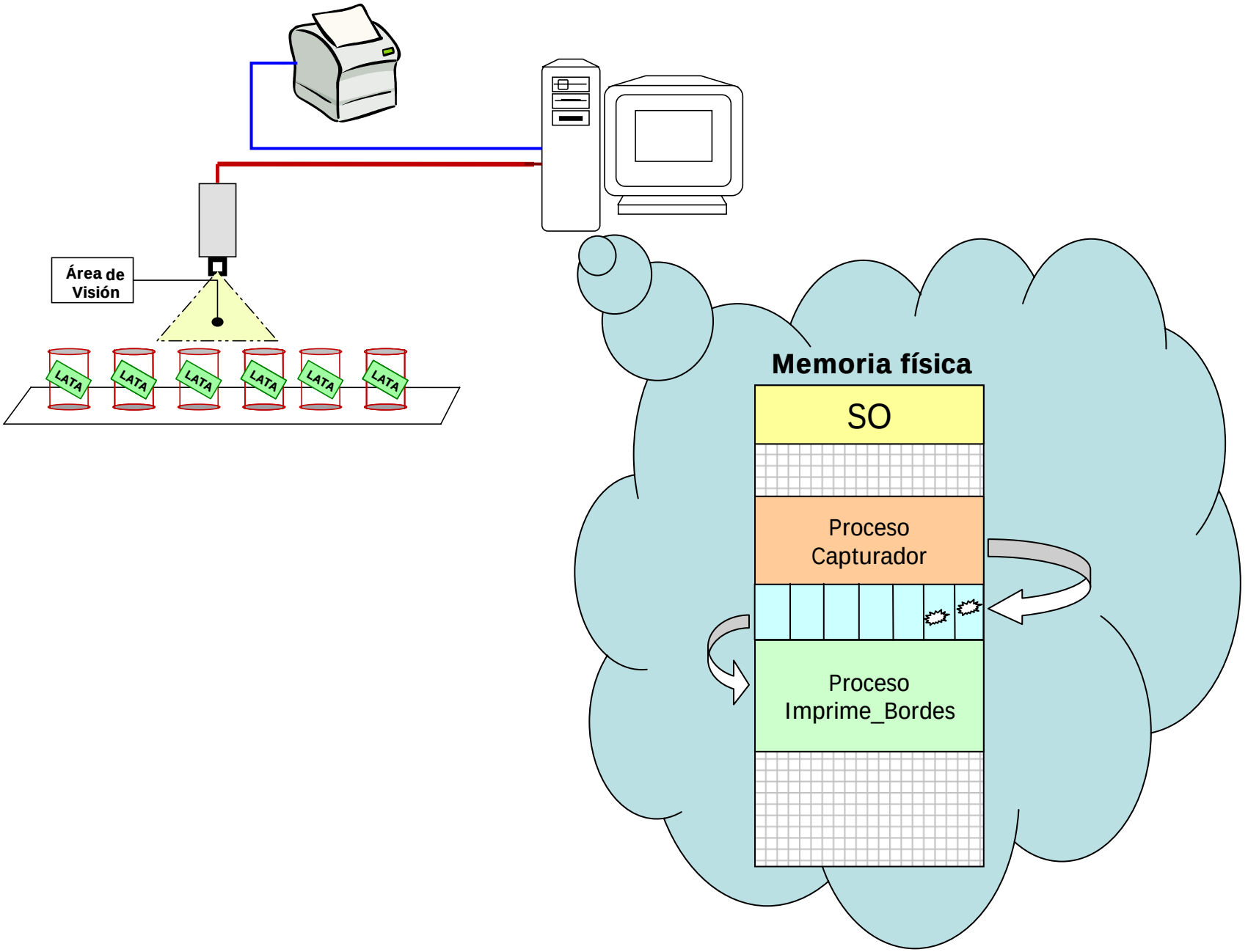
Problema del búfer limitado



Variables Globales

```
#define TAM_BUFFER 100
typedef Imagen …..;
Imagen buffer[TAM_BUFFER];
int entra=0,sale=0, contador=0;
```

Proceso Capturador

```
for(;;)
{
  ...
  Capturar imagen img
  ...
  for(;contador==TAM_BUFFER;);

  buffer(entra)=img;
  entra=(entra+1)%TAM_BUFFER;
  contador++;
  ...

}
```

Proceso Imprime_Bordes

```
for(;;)
{
  ...
  for(;contador==0;);

  img=buffer(sale);
  sale=(sale+1)%TAM_BUFFER;
  contador--;
  ...
  Extraer Borde img
  Imprime resultado
  ...
}
```

SOLUCION CON SEMAFOROS

Variables Globales

```
#define TAM_BUFFER 100
typedef Imagen …..;
Imagen buffer[TAM_BUFFER];
int entra=0,sale=0;
Semaforo lleno=0, vacio=TAM_BUFFER, cerradura=1;
```

Proceso Capturador

```
for(;;)
{
  ...
  Capturar imagen img
  ...
  espera(vacio);
  espera(cerradura);
  buffer(entra)=img;
  entra=(entra+1)%TAM_BUFFER;
  señal(cerradura);
  señal(lleno);
  ...
}
```

Proceso Imprime_Bordes

```
for(;;)
{
  ...
  espera(lleno);
  espera(cerradura);
  img=buffer(sale);
  sale=(sale+1)%TAM_BUFFER;
  señal(cerradura);
  señal(vacio);
  ...
  Extraer Borde img
  Imprime resultado
}
```

SOLUCION CON RCC

Variables Globales

```
#define TAM_BUFFER 100
typedef Imagen …..;
cerradura: shared record {
           Imagen buffer[TAM_BUFFER];
           int entra=0,sale=0, contador=0; }
```

Proceso Capturador

```
for(;;)
{
  ...
  Capturar imagen img
  ...
  region cerradura when (contador < TAM_BUFFER)
  {
    buffer(entra)=img;
    entra=(entra+1)%TAM_BUFFER;
    contador++;
  }
  ...
}
```

SOLUCION CON RCC

Proceso Imprime_Bordes

```
for(;;)
{
  ...
  region cerradura when (contador > 0)
  {
    img=buffer(sale);
    sale=(sale+1)%TAM_BUFFER;
    contador--;
  }
  ...
  Extraer Borde img
  Imprime resultado
}
```

SOLUCION CON MONITORES

Variables Globales

```
#define TAM_BUFFER 100
typedef Imagen …..;
Monitor TMbufer
{
  Imagen buffer[TAM_BUFFER];
  int entra, sale, contador;
  condition prod, cons;

  Public:
  void Insertar(Imagen img)
  {
     if (contador == TAM_BUFFER)
        espera(prod);
     buffer(entra)=img;
     entra=(entra+1)%TAM_BUFFER;
     contador++;
     señal(cons);
  }
```

```
  Imagen Extraer()
  {
    Imagen img;

    if (contador == 0)
      espera(cons);
    img=buffer(sale);
    sale=(sale+1)%TAM_BUFFER;
    contador--;
    señal(prod);
    return img;
  }
  TMbufer()
  {
    entra=0;
    sale=0;
    contador=0;
  }
}
```

SOLUCION CON MONITORES

```
TMbufer Mibufer;
void Proceso_Capturador()
{
  ...
  Capturar imagen img
  Mibufer.Inserta(img);
  ...
}

void Proceso_Imprime_Bordes()
{
  ...
  img=Mibufer.Extrae();
  Extraer Borde img
  Imprime resultado
  ...
}
```

Primer problema de los lectores/escritores

SOLUCION CON SEMAFOROS

Variables Globales

```
int nl=0;
Semaforo cerradura=1, escritura=1;
```

Procesos Lectores

```
for(;;)
{
  ...
  espera(cerradura);
  nl++;
  if (nl==1)
    espera(escritura);
  señal(cerradura);

  Leer del recurso

  espera(cerradura);
  nl--;
  if (nl==0)
    señal(escritura);
  señal(cerradura);
}
```

Procesos Escritores

```
for(;;)
{
  ...
  espera(escritura);

  Escribir en el recurso

  señal(escritura);
  ...
}
```

SOLUCION CON RCC

Variables Globales

```
cerradura: shared record {
           int nl=0; }
```

Procesos Lectores

```
for(;;)
{
  ...
  region cerradura
  {  nl++; }

  Leer del recurso

  region cerradura
  {  nl--; }
  ...
}
```

Procesos Escritores

```
for(;;)
{
  ...
  region cerradura when (nl=0)
  {
    Escribir en el recurso
  }
  ...
}
```

SOLUCION CON MONITORES

Variables Globales

```
Monitor TLect_Esc_1
{
  int nl, nlbloq;
  bool escribiendo;
  condition lector, escritor;

  Public:
  void Abrir_Lectura()
  {
     if (escribiendo)
     {
        nlbloq++;
        espera(lector);
        nlbolq--;
     }
     nl++;
     señal(lector);
  }
```

```
 void Cerrar_Lectura()
  {
     nl--;
     if (nl==0)
        señal(escritor);
  }
 void Abrir_Escritura()
  {
     if ((nl<>0) || escribiendo)
        espera(escritor);
     escribiendo=true;
  }
 void Cerrar_Escritura()
  {
     escribiendo=false;
     if (nlbloq>0)
        señal(lector);
     else
        señal(escritor);
  }
}
```

SOLUCION CON MONITORES

```
 TLec_Esc_1()
  {
    nl=0;
    nlbloq=0;
    escribiendo=false;
  }
}
```

```
Tlect_Esc L_E;
void Proceso_Lector()
{
  ...
  L_E.Abrir_Lectura();
  Leer del recurso
  L_E.Cerrar_Lectura();
  ...
}

void Proceso_Escritor()
{
  ...
  L_E.Abrir_Escritura();
  Escribir en el recurso
  L_E.Cerrar_Escritura();
  ...
}
```

Problema de los filósofos comensales

SOLUCION CON SEMAFOROS

Variables Globales

```
#define NUMF 5;
Semaforo palillo[N]={1,1,1,1,1};
```

Filósofo i

```
for(;;)
{
  ...
  Pensando ...
  espera(palillo[i%NUMF]);
  espera(palillo[(i+1)%NUMF]);

  A comer ...

  señal(palillo[i%NUMF]);
  señal(palillo[(i+1)%NUMF]);
}
```

?

SOLUCION CON SEMAFOROS

Variables Globales

```
#define NUMF 5;
Semaforo palillo[N]={1,1,1,1,1};
```

Filósofo Par

```
for(;;)
{
  ...
  Pensando ...
  espera(palillo[i%NUMF]);
  espera(palillo[(i+1)%NUMF]);

  A comer ...

  señal(palillo[i%NUMF]);
  señal(palillo[(i+1)%NUMF]);
}
```

Filósofo Impar

```
for(;;)
{
  ...
  Pensando ...
  espera(palillo[(i+1)%NUMF]);
  espera(palillo[i%NUMF]);

  A comer ...

  señal(palillo[(i+1)%NUMF]);
  señal(palillo[i%NUMF]);
}
```

SOLUCION CON RCC

Variables Globales

```
#define NUMF 5;
cerradura: shared record {
           bool palillos[NUMF]={false, ...; }
```

Filósofo i

```
for(;;)
{
  Pensando ...
  region cerradura when (palillo[i%NUMF]==false) &&
                        (palillo[(i+1)%NUMF] ==false)
  {
    palillo[i%NUMF]=true;
    palillo[(i+1)%NUMF]=true;
  }
  A comer ...
  region cerradura
  {
    palillo[i%NUMF]=false;
    palillo[(i+1)%NUMF]=false;
  }
}
```

SOLUCION CON MONITORES

Variables Globales

```
#define NUMF 5
Monitor TFilosofos
{
  enum estoy {pensando, hambriento, comiendo};
  enum estoy estado[NUMF];
  condition Fil[NUMF];

  void Comprobar(int i)
  {
    if ((estado[(i+4)%NUMF]<> comiendo) &&
       (estado[(i+1)%NUMF]<> comiendo)  &&
       (estado[i]== hambriento))
    {
      estado[i]=comiendo;
      señal(Fil[i]);
    }
  }
```

SOLUCION CON MONITORES

Variables Globales

```
  Public:
  void Coger(int i)
  {
    estado[i]=hambriento;
    Comprobar(i);
    if (estado[i]<> comiendo)
      espera(Fil[i]);
  }
  void Soltar(int i)
  {
    estado[i]=pensando;
    Comprobar((i+4)%NUMF);
    Comprobar((i+1)%NUMF);
  }
  void TFilosofos()
  {
    for(int i=0;i<NUMF;i++)
      estado[i]=pensando;
  }
}
```

SOLUCION CON MONITORES

```
TFilosofos filo;
void Filosofo(int i)
{
  Pensando ...
  filo.Coger(i);
  A comer ...
  filo.Soltar(i);
  ...
}
```