Pedro Carballo - 143507 2 hojas

Programación Concurrente ATIC y Redictado - Parcial Práctico de MC - Primera Fecha - 14/05/2025

1. Resolver con ***SEMÁFOROS*** el siguiente problema. Se tiene un vector A de 1.000.000 de caracteres, del cual se debe obtener la cantidad de veces que aparecen los caracteres "F" y "C", utilizando ***4 procesos Worker***. Al terminar todos los procesos deben imprimir la cantidad de veces que aparece cada una de esas dos letras (F y C). **Nota:** maximizar concurrencia; únicamente se pueden usar los 4 procesos *Worker*.

2. Resolver con **SEMÁFOROS** el siguiente problema. Existen ***B barcos*** areneros que deben descargar su contenido en una playa. Los barcos se descargan de a uno por vez, y de acuerdo con orden de llegada. Una vez que el barco llegó, espera a que le llegue su turno para comenzar a descargar su contenido, y luego se retira. **Nota:** sólo se pueden usar los procesos que representen a los barcos; cada barco descarga sólo una vez; suponga que existe la función *DESCARGAR()* que simula que el barco está descargando su contenido en la playa.

3. Resolver con **MONITORES** el siguiente problema. En una acopiadora de cereales hay ***2 empleados***, uno para atender a los camiones de maíz y otro para los de girasol. Hay 30 camiones que llegan para descargar su carga (15 de maíz y 15 de girasol), cuando el camión llega espera hasta que el empleado correspondiente le avise que puede descargar el cereal. Cada empleado hace descargar los camiones que le corresponden de a uno a la vez y de acuerdo con orden de llegada. **Nota:** maximizar concurrencia; el camión sabe qué tipo de cereal lleva; todos los procesos deben terminar

Pedro Carballo - 14350/7

Hoja 1 de 2

(A)

1	B
2	B
3	B

① SEMÁFOROS

```
char A[1.000.000]; // ya cargado
int termine = 0;        sem espero = 0;
sem t = 1;

process Worker[id: 0..3]
{ char aux;
  int local cant F = 0;
  int local cant C = 0;

  for (i = id*250.000 to (((id+1)*250.000)-1))
    { aux = A[i];   // obtengo caracter
      IF (aux = 'F')        // si es = F
        { local cant F++; }   // incremento var F local
      else
        IF (aux = 'C')      // si es = C
          { local cant C++; }   // incremento var C local
    }

  P(sem F);
  cant F = cant F + local cant F;   // actualizo cant F global
  V(sem F);

  P(sem C);
  cant C = cant C + local cant C;   // actualizo cant C global
  V(sem C);

  P(t);
  termine ++;
  IF (termine == 4) { for (i=0...3) } v(espero);}   // si soy el ultimo en terminar aviso a todos
  V(t);
  P(espero);
  imprimir('F', cant F); imprimir('C', cant C);   // imprimo resultados
}

int cant F = 0;
int cant C = 0;
sem F = 1;
sem C = 1;
```

② SEMÁFOROS

```
bool libre = true;          sem espera[B] = ([B] 0);
cola esperando;             Sem mutex = 1;

process Barco [id: 0..B-1]
{ contenido c = ... ;   int auxId;
  // Llego a la playa
  P(mutex);
  IF(!libre);                    // si no está libre
   { esperando.push(id);         // me encolo
     V(mutex);
     P(espera[id]);              // espero
   }else { libre = false;        // si está libre lo marco como ocupado
           V(mutex);
  // Descargar(c);               // descargo

  P(mutex);

  IF (esperando.empty())         // si no hay barco esperando
                                 // marco como libre
      { libre = true;
    }
    else
        { auxId = esperando.pop();   // si hay barco esperando
          V(espera[auxId]);          // desencolo al siguiente
                                     // le doy el paso al
                                     siguiente
    }
  V(mutex);
}
```

Pedro Carballo - 14350/7

③ MONITORES

// se asume que 0 = maiz, 1 = girasol } tipos de cereal

// empleado 0 → se encarga de maiz
// empleado 1 → se encarga de girasol

```
process Camion[id:0..29]
{ int tipoCereal = ...;
  admin[tipoCereal].pedido();
  // DescargoContenido();
  admin[tipoCereal].libero();
}
```

```
process Empleado[id:0..1]
{
  for (i=0..14)
  { admin[id].proximo();
  }
}
```

```
Monitor Admin[id:0..1]
{ cond aviso; int esperando=0;
  cond siguiente; cond meFui;

  procedure pedido()
  { esperando++;          // incremento
    signal(aviso);        // aviso a empleado
    wait(siguiente);      // espero que me avise el empleado
  }

  procedure libero()
  { signal(meFui); }      // aviso que termine

  procedure proximo()
  { // si no hay nadie esperando, espero
    if (esperando==0){ wait(aviso); }
    esperando--;          // decremento
    signal(siguiente);    // aviso al sig. camión
    wait(meFui);          // espero que termine
  }
```