Resolución Parciales

Consigna 1er Parcial 2025 ATIC

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

Resolución

Inciso 1

```
int N = 1.000.000
int porcion = N / 4
sem barrera = 0;
sem T = 1;
sem mutexC = 1;
sem mutexF = 1;
char v [N];
int globalF = 0;
int globalC = 0;
int total = 0;
process worker [id: 0...3]{
    int inicio = id * porcion;
    int cantC = 0;
    int cantF = 0;
    int fin = inicio + porcion - 1;
    for i inicio to fin {
        if(v[i] = 'F'){
            cantF ++;
        }
        else if (v[i] = 'C'){
            cantC ++;
        }
    }
    P(mutexF)
    globalF += cantF;
    V(mutexF)
    P(mutexC)
    globalC += cantC;
```

```
    V(mutexC)
    P(T)
    total+= 1;
    if(total == 4) for i 1 to 4 V(barrera)
    V(T)
    P(barrera)
    Imprimir(cantC, cantF)
}
```

Inciso 2

```
sem mutex = 1;
sem espera[B] = ([B] 0)
Cola c;
boolean libre = true;
process Barco [id: 0..B-1]{
    // llegando
    int sig = 0;
    P(mutex)
    if(libre){
        libre = false;
        V(mutex)
    }
    else {
        push(c, id);
        V(mutex)
        P(espera[id])
    }
    Descargar()
    P(mutex)
    if(Empty(c)){
        libre = true
    }
    else {
        sig = pop(c)
        V(espera[sig])
    }
    V(mutex)
}
```

Inciso 3

```
Monitor Admin [id: 0...1]{
    cond espera;
    int esperando = 0;
    bool libre = true;
    cond irse;
    cond emple;

    procedure llegada(){
        esperando ++;
        signal(emple)
        wait(espera)
```

```
    }

    procedure proximo (){
        if(esperando > 0){
            signal(espera)
            esperando --;
            wait(irse)
        }
        else wait(emple)
    }

    procedure salida (){
        signal(irse)
    }
}
process camion [id: 0..29]{
int tipoCereal = ... ;
Admin[tipoCereal].llegada();
// descarga
Admin[tipoCereal].salida();
}
```

Consigna 7/10/2024

Se debe simular el uso de un sistema virtual de venta de entradas para un evento musical. El sistema cuenta con C cajeros virtuales que atienden indefinidamente. Sin embargo, como la venta de entradas comienza a una hora determinada, sólo atienden a partir del aviso de un Timer. Una vez que reciben dicho aviso, los cajeros atienden de acuerdo con el orden de llegada de los compradores. La atención consiste en recibir la solicitud del comprador (datos para el pago) y responderle si pudo comprar (o no) junto al comprobante de la operación. Para este evento se cuenta con E entradas y N compradores, donde cada comprador puede solicitar a lo suma una entrada. Resuelva usando SEMÁFOROS.

```
sem mutexPersona [N] = ([N] 0)
sem mutex = 1;
sem barrera = 0;
sem llegue = 0;
sem mutexCant = 1;
int cant = E;
int entradas = E;
Cola c;
solicitud solicitudes [N];
process Timer (){
    delay() // espera cant de tiempo determinada
    for i 1 to C V(barrera)
}
process Caja[id: 0...C-1]{
    text solicitud;
    int sig = -1;
    bool pudeComprar = true;
    respuesta r;
    P(barrera)
    while (true) {
```

```
        P(llegue)
        P(mutex)
        Solicitud sig = pop(c)
        V(mutex)
        P(mutexCant)
        if(cant == 0) {
            pudeComprar = false;
            r = 'No hay entradas disponibles'
        }
        else {
            pudeComprar = true;
            cant --;
            r = ...;
        }
        V(mutexCant)
        solicitudes[sig] = (r,pudoComprar)
        V(mutexPersona[sig])
    }
}
process Persona[id:0..N-1]{
    text solicitud;
    P(mutex)
    push(c(id,solicitud))
    V(mutex)
    V(llegue)
    P(mutexPersona[id]
    text resp;
    bool pude;
    resp, pude = solicitudes[id];
}
```