

Parcial 14-05-2025

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 números, del cual se debe obtener el promedio de sus valores utilizando 10 procesos Worker. Al terminar todos los procesos deben imprimir el resultado. **Nota:** maximizar concurrencia; únicamente se pueden usar los 10 procesos *Worker*.
2. Resolver con **SEMÁFOROS** la siguiente situación. A una acopiadora de cereales llegan 20 camiones que deben descargar su cereal. Los camiones se descargan de a uno por vez, y de acuerdo con orden de llegada. Una vez que el camión llegó, espera a que llegue su turno para comenzar a descargar su cereal. **Nota:** sólo se pueden usar procesos que representen a los camiones; cada camión descarga sólo una vez; la descarga del camión se simula por medio de la función *DESCARGAR()* llamada por el camión.
3. Resolver con **MONITORES** el siguiente problema. En una cátedra hay un profesor, un auxiliar y 100 alumnos. Cada alumno continuamente hace consultas que pueden ser de dos tipos: TEORICAS o PRÁCTICAS; cada vez que tiene una consulta para hacer, se la envía por mail al docente correspondiente, y espera a que este le envíe la respuesta. El profesor sólo atiende consultas TEORICAS y el auxiliar sólo consultas PRÁCTICAS; cada uno resuelve sus consultas de acuerdo con orden de llegada. **Nota:** maximizar concurrencia; el alumno sabe de qué tipo es cada consulta; los procesos **NO** deben terminar.

1 - Semáforos

Aclaración : No hago semáforos hace bastante!!! jajajjsjsj.

Mejor me conviene dejar una variable sem espero = 0 y cuando es el numero 10 que haga 10 veces V a esa variable. Después lo demás lo dejaría con la misma lógica.

```
int A[1.000.000]; //No tengo un semaforo del arreglo ya que cada proceso
recorre su parte y no se van a cruzar en algun momento.
```

```
int suma = 0;
sem mutexSuma = 1;
```

```
int cant = 0;
sem mutexCant = 1;
```

```
sem espera([N] 0); //Lo uso para que esperen a que este la suma de todos
```

```
process worker[id:1..10]{
    int cantidadRecorrer = 1.000.000 / 10;
    int inicio = id * cantidadRecorrer; //Yo arranco desde acá
    int fin = inicio + cantidadRecorrer; //No tiene que ser -1?
    int promedio;
    int sumaMia = 0;
```

```

    for i: inicio to fin{
        sumaMia+= A[i];    //Sumo lo que hay en la posición.
    }

    P(mutexSuma);
    suma+= sumaMia; //Carga la suma de sus campos en la variable compartida
    V(mutexSuma);

    P(mutexCant);
    cant++; //Avisa que ya sumo lo suyo
    if (cant = 10){
        for i: 1 to 10{
            V(espera[i]); //Los despierta incluso a el
        }
    }
    V(mutexCant);

    P(espera[id]); //Se duerme esperando que todos terminen

    P(mutexSuma);
    promedio = suma;
    V(mutexSuma);

    promedio = ( promedio * 10 );
    //Informa el promedio
}

```

2 - Semáforos

Este seria passing the batton (No se como es el algoritmo pero lo hago como entiendo yo).

```

sem espera([20] 0)

c colaEspera;
sem mutexCola = 1;

bool libreDescarga = True;
sem mutexLibre = 1;

process camion[id: 1..20]{
    int idprox;

    P(mutexLibre);

```

```

if ( not libreDescarga){
    V(mutexLibre); // ???

    P(mutexCola);
    colaEspera.push(id); // Encola su id indicando que es el proximo
    V(mutexCola);

    P(espera[id]) //Se duerme hasta ser el proximo

}else{
    libreDescarga = false; //Lo usa
    V(mutexLibre); // ???
}

Descargar()

P(mutexCola);
if (colaEspera.isEmpty()){
    V(mutexCola);

    P(mutexLibre);
    libreDescarga = True; //Lo libera ya que no hay alguien que siga
    V(mutexLibre);

} else {
    colaEspera.pop(idprox); //id indica que es el proximo
    V(mutexCola);
    V(espera[idProx]); //Le avisa al proximo que puede pasar
}

}

```

3 - Monitores

Acá no hago que los profesores respondan la respuesta dentro del monitor ya que mientras uno responde el alumno no puede dejarle su consulta en el mail.

Tengo dos opciones :

1. Hacer un monitor para cada mail cosa de que el proceso se fije en su monitor correspondiente.
2. Que haya un solo monitor y que en caso de ser de un tipo deje el mensaje en un lado correspondiente y los dos procesos de profesores accedan al mismo monitor para ver las preguntas y responderlas (Debería si o si tener dos vectores y es lo mismo separar

el monitor). Me parece mejor ya que cada uno no debe esperar si el otro profesor esta enviando o esperando una respuesta.

3. No es una arreglo para despertarse.

```
Monitor MailPractico{
    sem vectorEsperaRespuesta[100]    //El semaforo que usa para esperar
    respuesta
    txt vectorRespuesta[100]          //Donde se almacena la respuesta

    cola c;    //La cola donde se guarda id y consulta

    cond tengoConsulta;

    procedure esperarConsulta(txt out consulta,txt out id){
        if ( c.isEmpty()){
            wait(tengoConsulta);    //Si no hay consultas me duermo
        }
        c.pop(id,consulta) //Saca la consulta con el id y se retorna por los
    parametros
    }

    procedure enviarRespuesta(id,respuesta){
        vectorRespuesta[id] = respuesta;    //Deja la respuesta
        signal(vectorEsperaRespuesta[id]);    //Le avisa que ya le respondio
    }
}

Monitor MailTeorico{
    sem vectorEsperaRespuesta[100]    //El semaforo que usa para esperar
    respuesta
    txt vectorRespuesta[100]          //Donde se almacena la respuesta

    cola c;    //La cola donde se guarda id y consulta

    cond tengoConsulta;

    procedure esperarConsulta(txt out consulta,txt out id){
        if ( c.isEmpty()){
            wait(tengoConsulta);    //Si no hay consultas me duermo
        }
        c.pop(id,consulta) //Saca la consulta con el id y se retorna por los
    parametros
    }

    procedure enviarRespuesta(id,respuesta){
```

```

        vectorRespuesta[id] = respuesta;    //Deja la respuesta
        signal(vectorEsperaRespuesta[id]);  //Le avisa que ya le respondio
    }

    procedure enviarPregunta(int in id,txt in consulta,txt out respuesta){
        c.push(id,consulta);    //Encola su id y consulta
        signal(tengoConsulta);  //Avisa que dejo una consulta
        wait(vectorEsperaRespuesta[id]); //Espera a que le responda
        respuesta = vectorRespuesta[id]; //Guarda la respuesta
    }
}

process profesor{
    txt consulta;
    int id;
    txt respuesta;
    while(True){
        MailTeorico.esperarConsulta(consulta, id);
        //Responde consulta
        MailTeorico.enviarRespuesta(id,respuesta);
    }
}

process auxiliar{
    txt consulta;
    int id;
    txt respuesta;
    while(True){
        MailPractico.esperarConsulta(consulta, id);
        //Responde consulta
        MailPractico.enviarRespuesta(id,respuesta);
    }
}

process alumno[id: 1..100]{
    txt consulta;
    txt tipo;
    txt respuesta;
    while (true){
        //Escribe la consulta.... y setea el tipo
        if (tipo = teorica){
            MailTeorico.enviarPregunta(id,consulta,respuesta);
        }else{
            MailPractico.enviarPregunta(id,consulta,respuesta);
        }
        //Lee la respuesta
    }
}

```

}

}