Monitores - Explicación practica

Notas (Clase del 18/9/2025 - Presencial) :

> Desde un procedure de un monitor se puede llamar a procedimiento de otro monitor

> Un monitor nos asegura exclusión mutua por su uso, ya que hasta que no termine el process no se puede acceder por fuera y/o hasta que no se duerma internamente en el monitor.

> wait(vc) -> Lo duerme al final de la variable condición.

> Passing the condition se usa cuando importa el orden de llegada, por lo que se usa una cola y un contador para la cant de personas de esa cola ya que no podemos usar un empty(cond) de la variable condición (Aparte de la variable booleana libre).
> Encaso de que sea por un orden que no es de llegada se usa un arreglo ya que debe despertar específicamente a un proceso que cumpla con el orden dado

> Barrera, passing the condition, timer

> En el ejemplo 5 usa el delay(minutos) no debemos modelar el monitor ni nada del reloj ?

> Ejemplo si tenemos más de una cancha que es un monitor y hacer un monitor el cual tenga internamente un arreglo de las canchas **Teóricamente esta mal** ya que si dos personas de canchas distintas entran al mismo monitor una persona debe esperar.
> Esto ya que son dos cosas independientes (por más que funcionen igual) se debe poner por separado.
> La solución correcta es : Hacer dos monitores distintos (Monitor Cancha[id:0..1])

> Passing the condition - if, cliente servidor usa un while

> El while lo usamos para re chequear la información

Alocación SJN : Wait con prioridad

> Se realiza Passing the condition, manejando el orden explícitamente por medio de una cola ordenada y variables de condición privada.

```
monitor Shortest_Job_Next
{
bool libre = true;
cond turno[N];
```

```
cola espera;

// Lo que hace en este proceso es que en caso de que no este libre el recurso
encola ordenado mediante el tiempo de uso, y luego se queda esperando a poder
usarlo
procedure request(int id, int tiempo){
    if(libre)
        libre = false;
    else
        encolarOrdenado(espera, id, tiempo)
        wait(turno[id])
}

// Lo que hace este proceso es chequear la cola y en caso de que no este vacia
lo que hace es desencolar el proximo elemento y lo despierta con un signal
procedure relese(){
    if (empty(espera))
        libre = True
    else
        desencolar(espera,id)
        signal(turno[id])
}

}
```

> Puede usarse el empty(cola) ya que es una estructura explicita que tenemos que implementar

Buffer limitado : Sincronización por condición básica

> En este caso vemos como un monitor simula el uso de un buffer con un arreglo el cual tiene un espacio limitado y como se trata como productor y consumidor

```
monitor Buffer_Limitado
{

typeT buffer[N];
int ocupado = 0; int libre = 0; int cantidad = 0;
cond not_vacio, not_lleno;

// Este procedimiento funciona como el productor.
// Lo que hace es esperar a que la cantidad no sea N (no este lleno) (debe ir
en un while asi no pisa otros valores) mientras se queda esperando que le
```

```
avisen que puede depositar un valor
// Deposiat el dato, pone libre en la sig posición (manejando el vector como
una cola circular), incrementa la cantidad de elementos y despierta a algun
proceso en caso que este esperando para retirar

procedure depositar(typeT  dato){
    while (cantidad == N )  await(not_lleno);
      buffer[libre] = dato
      libre = (libre +1 ) mod N
      cantidad ++
      signal(not_vacio)
}

// Este proceso funciona como consumidor
// Lo que hace es evaluar en el while si la cantidad es 0 , en caso de serlo
se duerme en not_vacio, hasta que lo despierten
// Luego guarda el resultado en el parametro (usando la variable ocupado),
setea ocupado para la cola circular, decrementa la cantidad de ocupados y por
ultimo despierta a algun productor si quiere depositar

procedure retirar(typeT &resultado){
    while (cantidad == 0)  wait(not_vacio);
        resultado = buffer[ocupado]
        ocupado = (libre + 1) mod N
        cantidad --
        signal(not_lleno)
}

}
```

Lectores y escritores : Broadcast Signal

> En este ejemplo simulamos el uso de una base de datos, donde hay lectores y escritores. Los escritores solo pueden entrar si no hay lectores o escritores. Y los lectores solo si no hay nadie escribiendo.

> Usamos la técnica del "Broadcast Signall" o más bien el signal all. El cual pone a competir a todos los lectores que desean leer la BD.

```
Monitor Controlador_RW{

    int nw = 0; int nr = 0  // Manejamos los escritores y lectores
    cond ok_leer; cond ok_escribir;
```

```
    procedure pedir_Leer(){
        // Para que pueda leer no debe haber nadie escribiendo
        while (nw > 0) wait(ok_leer)
            nr ++
    }

    procedure liberar_lectura(){
        nr--
        // Si soy el ultimo lector despierto si hay algun escritor esperando
        if (nr == 0)
            signal(ok_escribir)
    }

    procedure pedir_Escribir(){
        //Para escribir no tiene que haber lectores ni escritores
        while ( (nr > 0) OR (nw > 0) ) wait(ok_escribir)
            nr++

    }

    procedure liberar_Escritura(){
        nr--
        // Acá debo liberar para lectores o escritores
        signal(ok_escribir)
        signal_all(ok_leer)
    }

}
```

> Acá en el liberar escritura puedo poner un signal_all a los escritores pero seria un poco menos eficiente ya que hago competir a todos contra todos.

Lectores y escritores : Passing the condition

> Las variables se mantienen como el ejercicio anterior pero se suman dos variables **dr y dw** las cuales nos indica cuantos procesos están dormidos en la variable condición respectiva.

> Nos aseguramos que cuando un proceso termina la lectura y es el ultimo, le pasa el lugar a un escritor y ningún otro proceso le va quitar la chance de entrar.

> Se le da la prioridad a los lectores cuando la BD esta vacía.

> Cambian los whiles por los ifs

```
Monitor Controlador{

    int nw = 0, nr = 0;
    int dw = 0, dr = 0; //Nuevas variables
    cond ok_leer, ok_escribir;

    procedure pedir_Leer(){
        if (nw > 0) {
            dr++ //Incrementamos en uno la cant de personas esperando para
leer
            wait(ok_leer)
        }else{
            nr++
        }
    }

    procedure liberar_Lectura(){
        nr--
        if (nr == 0) and (dw > 0){
            //Si soy el ultimo lector, despierto al sig escritor (prioridad)
            dw-- // Decremento el contador de espera
            signal(ok_escribir) //Lo despierto
            nw++ // Incremento ya la variable de escritores
        }

    }

    procedure pedir_Ecribir(){
        if ( (nr > 0) OR (nw > 0) ){
            dw++ //Incremento la cant de escritores en espera
            wait(ok_escribir)
        }else{
            nw++
        }

    }

    procedure liberar_Escritura(){
        if (dw > 0){
            //Si hay algun lector dormido lo despierta
            dw--
            signal(ok_escribir)
        }else{
            nw-- // Resto recien acá la cant de escritores en la BD
            if (dr > 0){
```

```
                nr = dr // Marco que los que estan dormidos ya estan en la BD
                dr = 0 // Marco que no hay niguno esperando
                signal_all(ok_leer) //Se deja pasar a todos - competir
            }
        }
    }
}
```