Monitores: módulos de programa con más estructura, y que pueden ser

implementados tan eficientemente como los semáforos.

Mecanismo de abstracción de datos:

• Encapsulan las representaciones de recursos.

• Brindan un conjunto de operaciones que son los únicos medios para manipular

esos recursos.

Contiene variables que almacenan el estado del recurso y procedimientos que

implementan las operaciones sobre él.

Ya no hay variables compartidas, se hace todo dentro del monitor.

Los procedimientos dentro de un mismo monitor, no se pueden ejecutar de forma concurrente.   
La em, explicita uno de los dos tipos de exclusión, no se van a ejecutar dos de forma concurrente, hasta que no se duerma o no haya terminado ningún otro no va a poder otro empezar, la sincronización por condicion si lo vamos a implementar, con la variable condición dormimos un proceso-

Hasta que el que se ejecute, no se duerma o no haya terminado, hasta ese momento ninguno podrá ejecutar dentro de este monitor. Estado del recurso dentro del monitor, no existe la posib de que dos mofid a la variable em. de forma implícita.

Por condición, con variables condición, vamos a poder dormir un proceso, no puede continuar por algo, y deja que otro pueda ejecutar un procedi u otro arranca. un programa concurrente por procesos activos, concurrentemente y que en algún momento o llamado a los monit para trabajar con algún recurso compartido, son monitores pasivos, no estan corriendo sino que cuando un proceso hace un llamado, recién ahí se empezara a ejecutar , mientras no hace nada.

Ventajas:

El estar separados los procesos que hacen uso de los recursos compartidos y la forma en que se implementa, hace que al monitor no le importe desde donde estén llamando y a su vez a un proc no le interesa como esta implementado el proced, solo la semantica como tiene uqe trabajar ese procedimiento, vamos a tener procesos concurrentes, el monitor tiene los proce concu, van a poder llamar a los proced que tiene el monitor en su interfaz, llamando a esos procedim, el motor tendra variables o datos que representar el recurso, y otro que son para la sincronizacion para el manejo dentro del monitor, las variables no pueden ser accedidas por los procesos, LA UNICA FORMA DE TRABAJAR CON ESAS VARIABLES es solo los exportados por el monitor.

**MONITOR.** Interfaz y cuerpo- interfaz especif las op en partic para procedimientos entonces la interfaz especifica los proce que exporta el monitor, llamados por los distintos proc, para trabajar con un rec compartido, variables permanentes(en el momento de crearse el monitor)  
**procedimientos**, no funciones, no return, devuelve valor a través d param de E o ES o S

Vamos a llamar a un procedimiento de un monitor, monitor.nombredelproc(param)

Dentro del cuerpo del monitor van a poder acceder a las variables permanentes,tb a las locales, y a diferencia de las permanentes con cada llamado al procedimiento las variables se vuelven a resetear, también acceder a los parámetros que recibió el proced, a todo esto puede acceder para poder trabajar , cuando se está diseñando el monitor **nunca vamos a saber el orden en que fue llamado**, tenemos que hacerlo de manera de que si se usa en cualquier orden funcione bien igual,

Cuerpo del monitor

nombre

Declaración de variables permanentes

Variables que no se pueden inicializar en el momento de declararla, una matriz, código de inicialización para las que no pueden inicializarse en el momento.

Si tuviésemos que utilizar el codigo de inicialización, hasta que no termina de ejecutarse, no va a aceptar ningún llamado a ningún procedimiento.

Además de las variables e inicio, cuerpo e implem de cada uno de los ejecutados por el.

Un programa puede tener muchos monitores diferentes. Exclusión mutua implícita a un mismo monitor, yo puedo tener muchos que hagan lo mismo , sobre distintas variables.

No poner loop, do while true en el código de inicialización.

puedo declarar un arreglo de monitores, matriz

Va a tener un cuantificador desde dónde hasta dónde, y generamos automáticamente esa cantidad de monitores, indico a cual acceder siempre.

Ejemplo básico del uso de monitores

cinco proc empleados que hacen algún producto, hay un proc coordinador que cada cierto tiempo ve cuántos producto se han hecho.

el contador dentro del monitor, y al admin e contador. NEcesitamos un monitor, que adentro va a tener un contador, y dos proced , uno llamado por los empleados para increm en 1 , y otro llamado para ver cuantos elem fueron agregados.

monitor total{

| process empleado[id: 0..4] {  while (true)  { ……  TOTAL.incrementar();  ……  }  }  process coordinador{  int c;  while (true)  { ……  TOTAL.verificar(c);  ……  }  } | cant =0  proced increm(){  cant:=cant+1//dentro del monitor  }  proced verificar(R:in out){  R:= cant  } |
| --- | --- |

el trabajo con la variable dentro del monitor. No deberíamos tener un proc para setear un valor n para ver que tiene.

código de los empleados

Tenemos dos procesos A y B, donde A le debe comunicar un valor a B (múltiples veces).

A no le da un valor hasta que a no lo envio y viseversa. buffer para uno, unico proc consumidor y productor.

necesitamos algo que nos permita manejar sinc por condición, sino se genera busy waiting.

Los monitores tienen las variables condición, esta cv tiene internamente asco una cola de procesos dormidos o demorados, cola, no podemos dejar ni sacar elemento, usaremos funciones de las variables condición, tmp podemos ver sus elem ahí dentro, estructura interna del monitor. Nos brindan operaciones, tengo que usar la variable dentro del monitor donde fue declarada. Operaciones con esta variable condición

La sincronización por condición es programada explícitamente con variables

condición → cond cv;

El valor asociado a cv es una cola de procesos demorados, no visible directamente al

programador. Operaciones sobre las variables condición:

• **wait(cv)** → el proceso se demora al final de la cola de cv y deja el acceso

exclusivo al monitor. Se queda dormido y libera el monitor para que otro pueda usarlo.

• **signal(cv) → despierta al proceso que está** al frente/**primero de la cola** (si hay alguno) y lo saca de ella. El proceso despertado recién podrá ejecutar cuando readquiera el

acceso exclusivo al monitor. Cuando logra acceder, no empieza de nuevo la op, sigue del wait.

• **signal\_all(cv)** → despierta todos los procesos demorados en cv, quedando vacía la

cola asociada a cv.

¿Cuándo puedo acceder a usar el monitor? depende de disciplinas de señalización:

se dif en que cuando el que es despertado puede usar.

signal and continue, la que usamos!!

dispoierta al proc pero continua usando el monitor hasta que termina

signal and wait, el qeu hace el signal compite y el despertado continua ejecutando.

NO SON USADAS EN LA PRACTICA

empty, si hay proc o no dormido

wait con prioridad → además de parametro la variable condición, cierta prioridad, demora el proceso pero en vez de el orden de llegada, los ordena por la prioridad dada.

minrank permite verla prioridad del priemr proc que esta dormido, no lo despierta ni nada, solo me dice la prioridad.

como es el comportamiento de los proc de acuerdo a

**• Signal and Continued:** el proceso que hace el signal continua usando el monitor, y el proceso despertado pasa a competir por acceder nuevamente al monitor para continuar con su ejecución (en la instrucción que lógicamente le sigue al wait).

**• Signal and Wait:** el proceso que hace el signal pasa a competir por acceder nuevamente al monitor, mientras que el proceso despertado pasa a ejecutar dentro del monitor a partir de instrucción que lógicamente le sigue al wait.

signal y vacia, ese signal no tiene efecto posterior, si no hay nadie dormido el primer proc que haga un wait se queda dormido.

**Comparamos las variables condición de los monitores con P y V de Semaforos.**

WAIT → El proceso siempre se duerme

P→ El proceso sólo se duerme si el semáforo es 0.

SIGNAL → Si hay procesos dormidos despierta al primero de ellos.

En caso contrario no tiene efecto posterior.

V → Incrementa el semáforo para que un proceso dormido o que haga un P continue.

No sigue ningún orden al despertarlos.

**Monitor semáforo**

S == 1 repres e valor interno del sem

Pos var cond, para q los proc q estan intentando hacer el p mientras el semáforo está en 0 se queden dormidos

Implementacion de los dos proce, P si s>0 solo decremento, pero sino =0, dormir al proceso en la variable condicion pos, los duerme en el orden que fueron llamados y los despierta igual, por si hay algún proc dormido hago un signal de pos

chequeamos la condicion cada vez que el proceso es despertado, si volvio a ser cero no podria continuar. en vez de un if para cheqear uan sola vez la cond, mientras sea 0 el proc se duerme en pos, otro hace v y se despierta, por la dudas vuellve a chequear

La dif con los semaforos, se mantiene un certo orden en el paso de la operac p, cuando un proc hace V estaria despertando al primero, el unico dormido qe podria llegar a pasar, no se cuple un orden total, uno le gana el acceso y pasa antes cierto orden entre los dormidos

deberiamos usar un signal all despierto a todos los dormidos, pasan a competir, y uno logra entrar para ejec p y pasar al sem , igual que los semaforos q utilzabamos.

¿si quieren pasar en el orden que fueron intentando?

qeu el primero se el q logree terminar, orden en la forma en qe pasan al p, como simular lso semafoos con estos cambios, fair que deje avanzar a todos, los proc demorados ntenta haer p, q lo hagan en el ordeen qeu la pidieron. se respete el orden para poder pasar el P , aademas otro proc no les puedan ganar a ls qe acaban de ser despertados.

passing the conditions, si es 0 o mayor, s > 0 continua no se duerme.

q hace la tecnica?

pro dormido esperando qe s>0 , hy un proc que es la que increm en 1 el contador, la deja con 1, si yo marco el cambio en S, si dejo con 1 S vamos a tener problemaas q un prco peude ganarle los q no peuden ser despertados, entones lo hace sin modif el valor de S,

le paso al proc que despierto el derecho de poder continuar, sin modif la cond, para qe otro no gane la “prioridad”

siempre qeu quier respetar un orden, tengo que usar la tecnica passing the condition, para poder ordenar la ejecucion, no puedo ordenar quien logra acceder, de los que acceden a quien dejo pasar.

despierto a alguien con signal increm la cantidad

si no habia naadie , espera= 0 no hya nadie dormido

---------------------------------------------------------------------------------------------------------------------------

sinc y comunicacion monitores

forados por: procesos que trabaja continuamente, y por otor lado los monitores, pasivos, esperan a que les hagan pedidos. compuesto por mas de un proceso y podemos tener uno o mas monitores. arreglos o matrices de monitores.

cada monitor sus variables, procedimientos, se manejan como monitors independientes. Ejemplo 1, uno solo.

son lo unicos componentes, monitores y procesos, no hay variables compartidas, o son del proceso o son del monitor.

estructura dentro del monitor, procesos comunies llamados, estructura interna dentro ddel monitor, tres partes, dec de var permanentes, estado del monitor, representa un recurso compartido, las variables permanentes repres un estado, estos procedim puede tener un conj de variables locales y se generaria una copia cada vez a q se invoca.

cod de iinic, únicamente para inic las variables permanentes, cola o matriz, se puede hacer en el codigo de inicializacion entre llaves, hasta q el codigo de inicializacion no termino de ejecutarse no va a llamar a ningún pedido, n empieza digamos.

los procesos no acceden a la variable, sino que pasa como parámetro y el proc que como param de alida devuelve el valor. si queremos comunicar se hace a través de los llamados a los procedim, y no se comunican de forma directa los proce ni a traves de una var compartida.

no se mantiene orden en los llamados. no puede haber mas de un pro ejec a la vez en un monitor, no se mantiene el orden de llegada, entra cualquiera, uno de ellos logra pasar, sin repetaar orden, si necesitamos eso va a ser a mano.

excl mutua dentro de un monitor, es implicita, se ejecuta uno solo dentro,

la sincron es cn las variables cndicion, son variables de los monitores, el unc q la accede es el q las declaro, son colas de proc demoraos esperadno a que se de una sit, duermo al fiinal de la cola, si o si, en l ejec de ese proc en la variable cond, duerme al proc qeu hizo el lamado en la cola. cuando es despertado, continua su ejec con la sentencia desp del wait,

como despierno uno? signal! despierta al q etaba dormido, lo saca de la cola per no le da acceso a la cola inmediatamente, lo envia a competir por acceder nuevamente al monitor, en algun momento, cuando logra accdere continua con la ejec

si quiero despertar a todos, especie de barrera, podria llamar a signal all , los saca de la cola y los manda a tods a competir para acceder al monitor y acc al monitor

**Ejercicio 1**

Existen N personas que desean utilizar un cajero automático. En este primer caso no se debe tener en cuenta el orden de llegada de las personas (cuando está libre cualquiera lo puede usar). Suponga que hay una función UsarCajero() que simula el uso del cajero.

El monitor que representa? el cajeero? o admin el acceso a rec compartido,

en este caso e.m para acceder al cajero, me aalcanza para tener un monitor que represente el cajero, llamo al minitor, y dentro del proceso se hace el usoo. dentro del monitor uso del recurso compartido.

Monitor Cajero{

Procedure PasarAlCajero ()

{ UsarCajero();

}

}

Process Persona [id: 0..N-1]

{ ….

Cajero.PasarAlCajero();

….

}

**EJERCICIO 2**

Existen N personas que desean utilizar un cajero automático. En este segundo

caso se debe tener en cuenta el orden de llegada de las personas. Suponga que

hay una función UsarCajero() que simula el uso del cajero.

hay que respetar el orden de llegada, tenemos el monitor cajero con proc pasar al cajero, , y distintos proc persona que llaman al proce del monitor.

cuando el monitor queda libre tiene q pasar el primero.

cuando queda libre? cuando termina de usarlo la persona, tmp permite acceder a armar la cola fuera del cajero, quedan todos amontonados, no respetamos el orden, hay que cambiar la idea

si tengo uqe respetar le orden de llegada, no nos sirve que ell monitor sea el recurso compartido, que administre el acceso al recurso compartido,

necesitmos un monitor que nos permina admin el acceso, proqeu las personas eu llegan deber ordenarse, lo hago en el monitor qu contenga la cola de llegada donde se forma la fila qeu espera por acceder al cajero, dentro del proc persona, y la persona llama al monitor para esperar acceder y desp a lo sumo para avisar que termino.

Process Persona [id: 0..N-1]

{ Cajero.Pasar ();

UsarCajero();

Cajero.Salir();

}

llama al monitor, para acceder cuadno logra entrar es q es su turno solo el dentro del cajero y cuando termina avisa qeu lo libera, el monitor cajero tendra q permitir pasar, alguna variable condicion para ver si el cajero esta libre o no paraa ingresar.

Monitor Cajero{

bool libre = true;

cond cola;

Procedure Pasar ()

{ if (not libre) → wait (cola);

libre = false;

}

Procedure Salir ()

{ libre = true;

signal (cola);

}

}

no puedo administrar el orden en el acceso al monitor, si quiero permitir que ejecute o no, tengo que hacerlo dentro del codigo del pro monitor, sino deberia dormirlo..

passing the condition de acuerdo al orden en que fueron llegando

increm el contador para indicar ue hay alguien mas dormirdo, siempre previo del wait sno va a estr en 0.

y cuidado con el proc q acaba de ser despertado no tenga que volver a chequear la condicion,

**que pasa si el orden no fuera el de llgada?? si es una persona anciana prioridad.**

estructura de daos para manteder el rden, tambien con la edad de la persona, depende la eadad lo inserta al final, sino al principio, exitsste un nsetar qe de acuerdo a la edad lo inserta.

en la finla si podria usar el empty porque es una e.d no una variable condicion.

saco a una persona de una e.d pero despierto primero a una variable condicion?

una cosa tengo guardado en la e.d fila y otra cosa lo qe tengo en la variable condicion, necesito varaibles condicion prvadas, una por persona, podria ser con su id, en donde x se va a dormir,