DISEÑO DETALLADO

GABRIEL GONZÁLEZ CASTRO
BORJA GONZÁLEZ ENRÍQUEZ
FRANCISCO GARCÍA VÁZQUEZ
JUAN FERNÁNDEZ OTERO

CAMBIOS DISEÑO INICIAL

REQUISITOS FUNCIONALES

1. Los procesos lectores se podrán ejecutar concurrentemente la SC.
2. Los procesos escritores accederán a la SC en exclusión mutua.
3. Los procesos lectores accederán a la sección crítica en exclusión mutua con los procesos escritores.
4. Los procesos que no entren en la SC cuando precisen, serán suspendidos.

5.Si hay procesos lectores y hay procesos escritores, los segundos accederán primero a la sección crítica.

6.Una vez que un proceso escritor este sólo podrá ser adelantado por N procesos lectores.

7.Cualquier tipo de nodo podrá ejecutar cualquier tipo de proceso.

8. Dentro de escritores distinguiremos, escritores tipo I (pagos) y escritores tipo II (reservas y anulaciones).

9. Los escritores tipo I serán prioritarios.

CASOS DE USO

1. La SC está vacía y entra un proceso. Se ejecuta
2. Hay un proceso de gradas en la SC, entra un proceso de eventos y ejecuta su SC.
3. Hay un proceso de gradas en la SC, entra un proceso de pagos y se suspende a la espera de que acabe el de gradas
4. Hay un proceso de reservas en la SC, entra otro de reservas y se suspende a la espera de que acabe el de reservas .

5. Llegan indefinidos procesos lectores a un nodo, y un proceso escritor. Se acaba ejecutando el proceso escritor.

6.Un proceso escritor se ejecuta siempre antes que un proceso de lectores.

CONSIDERACIONES INICIALES

CONSIDERACIONES INICIALES

- De acuerdo, con el primer diseño consideramos un sistema distribuido, con comunicación mediante paso de mensajes.
- Primero resolveremos el acceso a la SC entre los nodos mediante el paso de testigo.

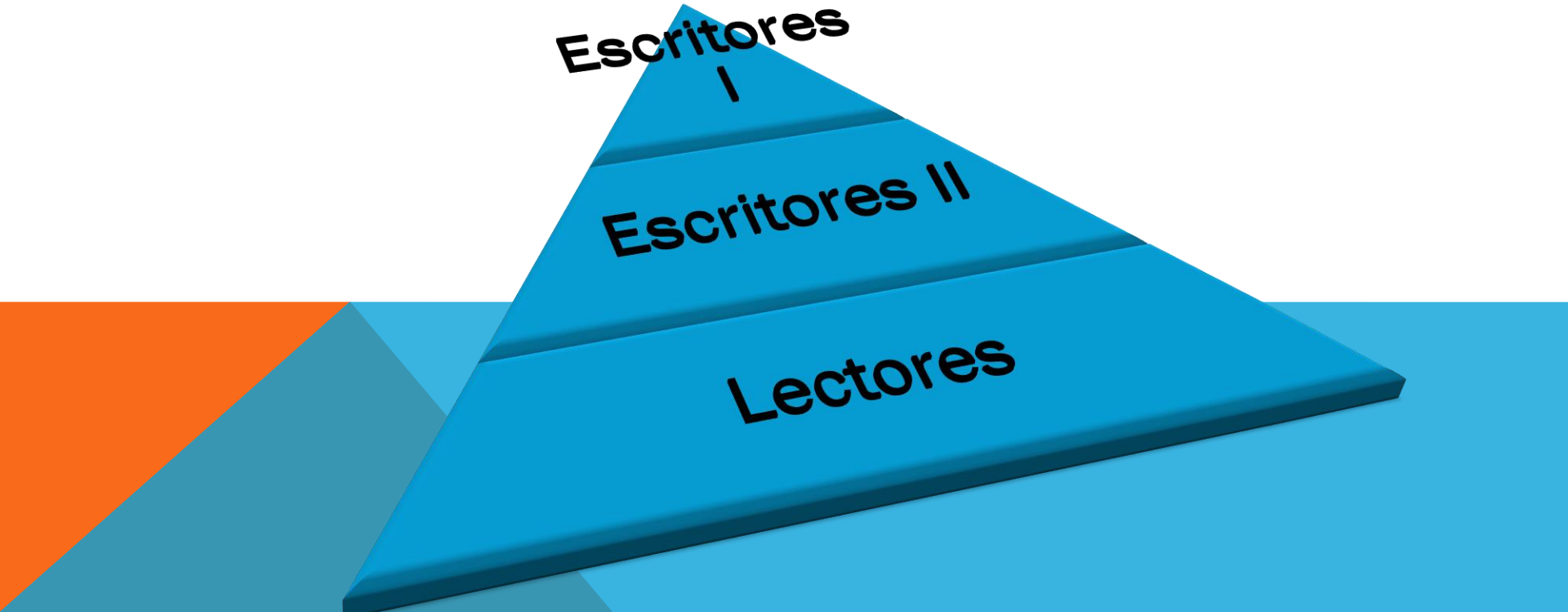
CONSIDERACIONES INICIALES II

- Luego, resolveremos la concurrencia dentro del nodo. Para ello usaremos los mecanismos usados para este caso (típicamente semáforos)
- Sin testigo no entra ningún nuevo proceso a la SC .
- Número de nodos estáticos e igual a 3.

DISEÑO

DISEÑO

- **Consideramos 3 tipos de procesos**
  - Escritores I : de tipo Pagos
  - Escritores II : de tipo Anulaciones y Reservas
  - Lectores : de tipo Eventos y Gradas



DISEÑO - ALGORITMO

- Vamos a usar el paso de testigo para la resolución de la contienda entre nodos

- Este algoritmo ofrece un menor número de mensajes enviado que otros algoritmos como puede ser el algoritmo basado en tickets.

DISEÑO - TESTIGO

- El mensaje que se envía, junto al testigo , se incluye un matriz de peticiones atendidas . Donde cada vector es el vector de peticiones atentidas de un tipo (Escritor I, Escritor II, Lector). También nº de adelantamientos de lectores.

- $\begin{pmatrix} 1 & 0 & 4 \\ 1 & 5 & 4 \\ 2 & 7 & 2 \end{pmatrix}$

Nº Petición atendida para el 3º nodo, de Escritor de tipo I

DISEÑO

- En cada nodo habrá a su vez una matriz de peticiones, de forma que cada nodo que tenga un proceso que vaya a entrar en su SC envía que tipo de proceso quiere entrar. Envía todos los nodos el tipo de proceso que quiere entrar y el id del nodo.

DISEÑO : ESCRITORES

- Cuando proceso de tipo escritor quiera entrar en la SC este envía el mensaje antes comentado, a partir de ahí, si se están ejecutando lectores en el nodo con el testigo se empecerá a contar el número de adelantamientos. Cuando se alcance el nº máximo no entraran más lectores.
- En caso de pasar el testigo a otro nodo, este ya sabe el nº de adelantamientos que se han producido.

DISEÑO – ESCRITORES II

- Cuando entre un escritor se resetea la variable que cuenta el número de adelantamientos

DISEÑO – LECTORES

- Cuando el nodo que tiene el testigo acabe los procesos lectores y haya una petición de un proceso escritor, se enviarán mensajes para saber si han acabado todos los procesos lectores de todos los nodos. Estos contestarán siempre que no haya procesos lectores ejecutándose o cuando terminen.

DISEÑO – MÓDULOS

- Main
- Receptor-peticiones
- Receptor-lector

PSEUDOCÓDIGO -LECTOR

```
if (!testigo) {
    if (lectores) {
        lectores++;
        adel();
        wait(sem_lectores);
         signal(semLectores);

    }
    else lectores++;
    wait(sem_lectores);
    solicitar_testigo();
    if (escritores){
        wait(sem_lectores2);
         }
    wait(mutexEscritoresI);
    wait(mutexEscritoresII);
    signal(sem_lectores);


SC;
lectores--;
```

PSEUDOCÓDIGO -LECTOR II

```
if (lectores || adelantamiento = max) {
    if (escritoresI != 0) {
        signal(mutexEscritoresI);
    }
    if (escritoresII != 0) {
        signal(mutexEscritoresII);
    }
    else {
        signal(mutexEscritoresI);
        signal(mutexEscritoresII);
    }
}
```

ESCRITOR II

```
if (!testigo) {
    if (EscritoresII == 0)
        solicitar_testigo();
}
escritores2++;
wait(EscritoresMutex2);
if (escritoresI) {
    escritoresesperandoII++
        wait(paso_esc2);
    escritoresesperandoII--;
}
wait(EscritoresMutex1);
SC;
escritores2--;
```

ESCRITOR II

```
if (escritoresI)
signal(EscritoresMutex1);
else {
     if (escritoresII)
          signal(EscritoresMutex2);
     else {
          if (lectores)
               signal(sem_lectores2);
     }
}
```

ESCRITOR I

```
if (!testigo) {
    if (EscritoresI == 0)
        solicitar_testigo;
}
escritores1++;
wait(EscritoresMutex1);
wait(EscritoresMutex2);
SC;
escritores1--;
```

ESCRITOR I

```
if (escritoresI)
signal(EscritoresMutex1);
else {
    if (escritoresII_esperando)
        signal(paso_esc2)

    else {
        if (escritoresII)
            signal(EscritoresMutex2);
        else {
            if (lectores)
                signal(sem_lectores2)
        }
    }
}
```

MÓDULO RECEPTOR LECTORES

```
while (1) {
     receive(PASO_ESCRITOR, &id_nodo);
     if (lectores!=0)
          wait (fin_lector);
     send(CONSULTAR_LECTORES, id_nodo);
}
```

ENVIAR FIN LECTORES

```
if (EXISTS peticion_escritor) {
    if (!testigo && lectores == 0)
        signal(fin_lectores);
    else {
        if (n_adel == MAX || NOEXISTS peticion_lectores) {
            for (i = 0; i < N - 1; i++) send(PASO_ESCRITOR, id_nodo[i]);
            for (i = 0; i < N - 1; i++) receive(CONSULTAR_LECTORES);
        }
    }

}
```

MANUAL DE REFERENCIA

MANUAL DE REFERENCIA

- Solicitar_testigo (int id ,int tipo) :void : el nodo que tenga un proceso que quiera entrar en la SC, envía un mensaje a todos los nodos con su id y con el tipo de proceso que quiera entrar.
- Enviar_testigo(int id_siguiente) :void: el nodo elige el siguiente nodo al que se le pasa el testigo.

MANUAL DE REFERENCIA II

- Consultar_lectores(int id_origen) :void : envía un mensaje a todos los nodos para saber si tienen procesos lectores en ejecución. Y se queda esperando a recibir todos los mensajes.
- Enviar_fin_lectores(int id_destino) :void: envían un mensaje conforme no hay lectores en ejecución al nodo que tiene el testigo.

PLAN DE PRUEBAS

PLAN DE PRUEBAS

- Para la evaluación del sistema supondremos un sistema con 3 nodos.
- El número de adelantamientos es de 3 lectores.

PLAN DE PRUEBAS II

1. En el sistema hay un proceso de pagos en el N1 que está ejecutando su SC y un proceso de eventos pide acceso a su SC el N2.

Resultado : Proceso de pagos seguirá ejecutándose y se suspende al proceso de eventos. Cuando termine la SC del proceso de pagos, le cederá el testigo al N2 y el proceso de eventos se ejecutará.

PLAN DE PRUEBAS III

2. Un proceso de anulaciones ejecutando su SC y piden acceso a la SC 3 procesos de reservas, dentro todo del mismo nodo :

Resultado: Cuando finalice su SC el proceso de anulaciones, se ejecutarán uno detrás de otro los procesos de reservas.

PLAN DE PRUEBAS IV

3. En un mismo nodo, está ejecutando su SC un proceso de anulaciones y pide acceso a SC un proceso de gradas y reservas.

Resultado: ejecuta su SC un proceso de anulaciones, luego proceso de reserva y por último de gradas.

PLAN DE PRUEBAS V

4. Están ejecutando su SC 3 procesos de eventos , quiere entrar en la SC un proceso de pagos. A continuación quieren entrar 4 procesos de gradas:

Resultado: se ejecutan 3 eventos y 3 gradas, luego el de pagos y por último 1 gradas.

PLAN DE PRUEBAS VI

5. Se esta ejecutando un proceso de pagos en el N1 y hay otro proceso de anulaciones suspendido esperando para ejecutar su SC. En el N2 hay esperando un proceso de pagos.

Resultado: Se ejecuta el proceso de pagos del N1 y al acabar se pasa el testigo al N2 para que ejecute el proceso de pagos. Cuando termine se vuelve a pasar al N1 para que se ejecute el proceso de anulaciones.

PLAN DE PRUEBAS VII

6. Hay 4 procesos de eventos ejecutando su SC en el N1, y 4 procesos de gradas en el N2 esperando para entrar a la SC.

Resultado: Se le pasa el testigo al N2 y se empieza a ejecutar los procesos gradas en el nodo 2.

PLAN DE PRUEBAS VIII

7. Se están ejecutando 2 procesos de gradas en el N1, a continuación pide acceso a la SC un proceso de anulaciones en el N2 y en el N3 piden acceso 5 procesos de gradas.

Resultado: Se ejecutan los 2 procesos de gradas del N1, luego 3 proceso de gradas del N3 . Cuando terminen estos 5 procesos, el N3 pasará el testigo y se ejecutará el proceso de anulaciones. Una vez terminado el mismo, se le pasará el testigo al nodo 3 y ejecutará los procesos restantes.

PLAN DE PRUEBAS IV

8. Hay un proceso de pagos en el N1 , un proceso de gradas en el N2 (con el testigo y el primero en solicitar entrar a la SC), y un proceso de pre-rreservas en el N3.

Resultado : Se ejecuta el proceso de gradas, luego el proceso de pagos del N1, y finalmente del pre-reservas del N3.

FIN