Universidad de los Andes

Infraestructura computacional

Concurrencia

Universidad de los Andes

Sincronización - señalamiento

- Semáforos:
  - Sirven para señalamiento y para exclusión mutua
  - Compuestos de:
    - contador (entero)
    - cola de procesos
  - Dos operaciones:
    - P(): solicitar
    - V(): liberar
  - Las operaciones ejecutan atómicamente

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – operaciones
  - `contador` indica el número de threads que pueden entrar antes de que se empiecen a bloquear
  - Si es negativo, indica cuántos threads están bloqueados
  - Si `contador` = 1, es un semáforo binario (similar a un mutex excepto por la apropiación)

```
P ( s ) {
  s.contador--
  if ( s.contador < 0 ){
    s.cola ← proceso
    dormir proceso
  }
}
```

```
V ( s ) {
  s.contador++
  if ( s.contador <= 0 ){
    q ← s.cola
    despertar q
  }
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – ejemplo: productor- consumidor
  - Inicialmente
    - `llenos.contador` = 0
    - `vacíos.contador` = tamaño buffer
    - `mutex` es un semáforo binario (`contador` = 1)

```
Productor( buzon, mensaje ){
  P( vacíos )
  P( mutex )
  buffer.add( mensaje )
  V( mutex )
  V( llenos )
}
```

```
Consumidor( buzon, mensaje ){
  P( llenos )
  P( mutex )
  mensaje = buffer.get( )
  V( mutex )
  V( vacíos )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – ejemplo: lectores-redactores
  - Inicialmente
    - `nLectores` = 0
    - `lectores y escritores` son semáforos binarios (`contador` = 1)

```
entrarLeer(){
  P( lectores )
  nLectores++
  if ( nLectores == 1 )
    P( escritores )
  V( lectores )
}
```

```
salirLeer(){
  P( lectores )
  nLectores--
  if ( nLectores == 0 )
    V( escritores )
  V( lectores )
}
```

```
entrarEscribir(){
  P( escritores )
```

```
salirEscribir(){
  V( escritores )
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – ejercicio: sincronización de barrera
  - Hay N threads; los primeros N-1 se bloquean
  - Al llegar el N, todos continúan con la ejecución

```
sincronizarBarrera(){
 …
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – solución sincronización de barrera:
  - Inicialmente
    - `nThreads` = 0
    - `mutex` es un semáforo binario
    - `barrera.contador` = 0

```
sincronizarBarrera(){
  P( mutex )
  nThreads++
  if ( nThreads == N ) V( barrera )
  V( mutex )
  P( barrera )
  V( barrera )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – ejercicio. Lectores – redactores sin inanición :
  - Inicialmente
    - `nLectores` = 0
    - `Lectores, escritores y paso` son semáforos binarios

```
entrarLeer(){
  P( paso )
  V( paso )
  P( lectores )
  nLectores++
  if ( nLectores == 1 )
    P( escritores )
  V( lectores )
}
```

```
salirLeer(){
  P( lectores )
  nLectores--
  if ( nLectores == 0 )
    V( escritores )
  V( lectores )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – ejercicio. Lectores – redactores sin inanición :
  - Inicialmente
    - `nLectores` = 0
    - `Lectores, escritores y paso` son semáforos binarios

```
entrarEscribir(){
  P( paso )
  P( escritores )
}
```

```
salirEscribir(){
  V( paso )
  V( escritores )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – implementaciones:
  - Java: Clase semaphore; métodos (entre otros muchos):
    - `acquire()`: equivale a P (adquiere una autorización para entrar)
    - `acquire(int n)`: adquiere n autorizaciones de una vez
    - `release()`: equivale a V (devuelve la autorización)
    - `release(int n)`: devuelve n autorizaciones
    - `tryAcquire()`: varios métodos parecidos a `acquire` pero no son bloqueantes (retornan un booleano indicando si lo consiguieron o no)

Universidad de los Andes

Sincronización - señalamiento

- Semáforos – implementaciones:
  - Posix: Están en `semaphore.h`; primitivas:
    - `sem_t s`: declaración de un semáforo
    - `int sem_init( sem_t *s, int shared, unsigned int n )`: inicialización
    - `int sem_wait( sem_t *s )`: equivale a P
    - `int sem_post( sem_t *s )`: equivale a V
    - `int sem_getvalue( sem_t *s, int *valn )`: retorna el valor del contador
    - `int sem_destroy( sem_t *s )`: destruye el semáforo

Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos
  - Cada monitor tiene asociado un semáforo binario (m)
  - `m.contador` = 1

Monitor M

variables, eventos

procedimiento 1

P(m)

…

V(m)

procedimiento 2

P(m)

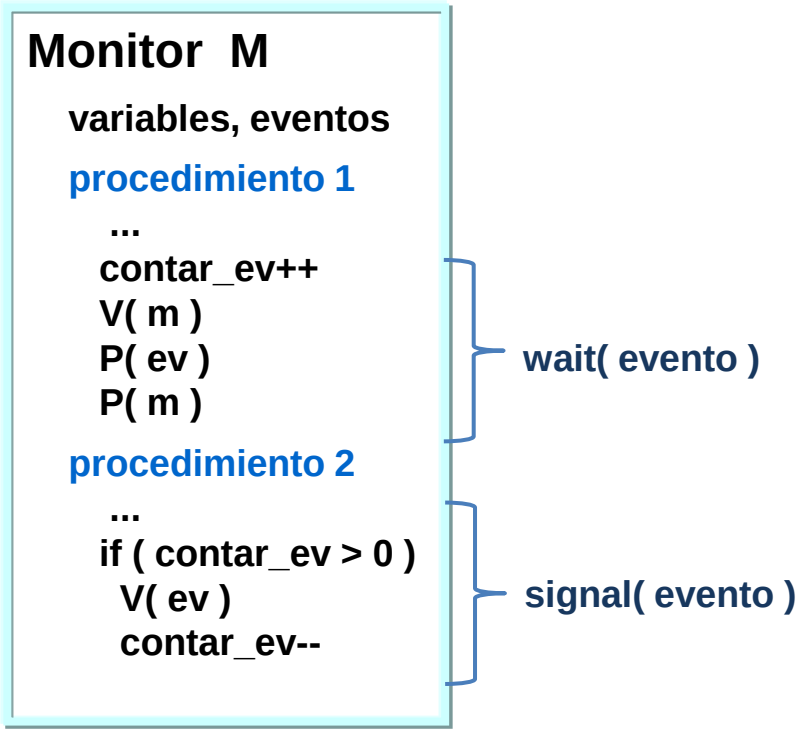
…

V(m)

Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos
  - A más de m, cada evento tiene asociado 1 semáforo binario (ev)
  - `ev.contador` = 0



Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos
  - Ejemplo: sincronización de barrera

```
private int n; //número de threads

public synchronized void barrera(){
  n--;
  if ( n == 0 ) notifyAll();
    else wait();
}
```

```
barrera(){
  P( mutex)
  n--
  if ( n == 0)
    while ( cont > 0 )
      V( cond )
      cont--
    V( mutex )
  else
    cont++
    V( mutex )
    P( cond )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos – ejercicio:
  - Reservar k:
    - Se dispone de una determinada cantidad (N) de copias de un cierto recurso
    - Hay varios threads ejecutando, y cada uno puede solicitar una cantidad arbitraria k de copias (k < N)
    - Si la cantidad solicitada está disponible, el thread se las apropia; si no, debe esperar a que se liberen copias

```
reservar( k ){
  …
}

liberar( k ){
  …
}
```

Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos – solución:
  - Reservar k en Java:
    - Inicialmente: n = N

```
private int n; //número de recursos

public synchronized void reservar( int k ){
  while ( n < k ) wait();
  n -= k;
}
```

```
public synchronized void liberar( int k ){
  n += k;
  notifyAll( );
}
```

Universidad de los Andes

Sincronización - señalamiento

- Implementación de monitores con semáforos – solución :
  - Reservar k con semáforos:
    - n = N
    - `mutex` es un semáforo binario, `s.contador` = 0

```
reservar( k ){
  P( mutex )
  while ( n < k )
    enEspera++
    V( mutex )
    P( s )
    P( mutex )
  n -= k
  V( mutex )
}
```

```
liberar( k ){
  P( mutex )
  n += k
  while ( enEspera > 0 )
    enEspera--
    V( s )
  V( mutex )
}
```

Universidad de los Andes

Sincronización - señalamiento

- Implementación de la sincronización en Java
  - Cada objeto tiene asociado un mutex para sincronización
  - Cada objeto tiene asociado un semáforo para espera
  - Cada clase tiene asociado un mutex para sincronización de métodos estáticos