

Semafori binari

- ◆ **Definizione**

- ◆ variante dei semafori in cui il valore può assumere solo i valori 0 e 1

- ◆ **Cosa servono?**

- ◆ servono a garantire mutua esclusione, semplificando il lavoro del programmatore
- ◆ hanno lo stesso potere espressivo dei semafori "normali"

- ◆ **Invariante dei semafori binari:**

- ◆ $0 \leq n_v + \text{init} - n_p \leq 1$, oppure
- ◆ $0 \leq \text{s.value} \leq 1$

- ◆ **Nota:**

- ◆ molti autori considerano una situazione di errore un'operazione **V** su un semaforo binario che abbia già valore 1

Semafori binari - Implementazione

```
class BinarySemaphore {
    private int value;
    Queue queue0 = new Queue();
    Queue queue1 = new Queue();
    BinarySemaphore() { value = 1; }
    void P() {
        [enter CS]
        int pid = <process id>;
        if (value == 0) {
            queue0.add(pid);
            suspend(pid);
        }
        value--;
        if (queue1.size() > 0) {
            int pid = queue1.remove();
            wakeup(pid);
        }
        [exit CS]
    }
}
```

```
void V() {
    [enter CS]
    int pid = <process id>;
    if (value == 1) {
        queue1.add(pid);
        suspend(pid);
    }
    value++;
    if (queue0.size() > 0) {
        int pid = queue0.remove();
        wakeup(pid);
    }
    [exit CS]
}
```

Semafori - Implementazione tramite semafori binari

- ♦ **E' possibile utilizzare un semaforo binario per implementare un semaforo generale**

- ♦ un semaforo **mutex** per garantire mutua esclusione sulle variabili
- ♦ un semaforo *privato* **delay[i]** per ogni processo **i** che partecipa
- ♦ una coda per garantire fairness

```
class Semaphore {  
  
    private BinarySemaphore mutex;  
    private BinarySemaphore delay[];  
    int value;  
    Queue queue = new Queue();  
  
    Semaphore(int v, int N) {  
        value = v;  
        mutex = new BinarySemaphore(1);  
        delay = new BinarySemaphore[N];  
        for (int i=0; i < N; i++)  
            delay[i] =  
                new BinarySemaphore(0);  
        queue = new Queue();  
    }  
}
```

Semafori - Implementazione tramite semafori binari

```
void P() {
    mutex.P();
    value--;
    if (value < 0) {
        pid = <id del processo
            che ha invocato P>;
        queue.add(pid);
        mutex.V();
        delay[pid].P();
    } else {
        mutex.V();
    }
}
```

```
void V() {
    mutex.P();
    value++;
    if (value <= 0) {
        pid = queue.remove();
        delay[pid].V();
    }
    mutex.V();
}
```