Implementazione dei monitor tramite semafori

- **Per Andrews ("Concurrent programming") è necessario utilizzare:**
  - un semaforo di mutua esclusione **e**
  - per ogni variabile di condizione **cond**$_i$, una coppia (**c**$_i$, **nc**$_i$)
    - **c**$_i$ è un semaforo correlato alla condizione, inizializzato a 0
    - **nc**$_i$ è il numero di processi che sono in attesa del verificarsi della condizione
- **Implementazione**
  - entrata nel monitor `{ e.P(); }`
  - uscita dal monitor `{ e.V(); }`
  - wait su cond$_i$ `{ nc`$_i$`++; e.V(); c`$_i$`.P(); e.P(); }`
  - signal su cond$_i$ `if (nc`$_i$ `> 0) { nc`$_i$`--; c`$_i$`.V(); }`

Implementazione dei monitor tramite semafori

- **Implementazione precedente**
  - è incompleta
  - non implementa signal urgent, ma signal & continue
  - il processo riattivato con la signal viene messo in esecuzione DOPO tutti quelli in coda di ingresso

Implementazione dei monitor tramite semafori

- **Ingredienti**
  - un modulo di gestione stack (per urgent)

```
interface Stack {
  void push(Object x);
  Object pop();
  boolean empty();
}
```

  - un semaforo di mutua esclusione **e**
  - per ogni variabile di condizione **cond**$_i$, una coppia (**c**$_i$, **nc**$_i$)
    - **c**$_i$ è un semaforo correlato alla condizione, inizializzato a 0
    - **nc**$_i$ è il numero di processi che sono in attesa del verificarsi della condizione
  - un "allocatore" di semafori
    (o alternativamente un semaforo per ogni processo)

Implementazione dei monitor tramite semafori

- **Inizializzazione**

```
Semaphore e = new Semaphore(1);
Stack stack = new Stack();
```

- **Entrata nel monitor**

```
e.P();
```

- **Wait su $cond_i$**

```
nc_i++;
if (!stack.empty()) {
  Semaphore s = stack.pop();
  s.V();
} else {
  e.V();
}
c_i.P();
```

- **Signal su $cond_i$**

```
if (nc_i > 0) {
  nc_i--;
  c_i.V();
  Semaphore s =
    new Semaphore(0);
  stack.push(s);
  s.P();
  /* free(s) / garbage coll.
   */
}
```

- **Uscita dal monitor**

```
if (!stack.empty()) {
  Semaphore s = stack.pop();
  s.V();
} else {
  e.V();
}
```

Sezione 7

7. Message passing