

Università degli Studi di Padova

## Gestione di eventi asincroni

# SCD

Anno accademico 2006/7  
 Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, [tullio.vardanega@math.unipd.it](mailto:tullio.vardanega@math.unipd.it)

Corso di Laurea Specialistica in Informatica, Università di Padova 1/18

Università degli Studi di Padova

## Gestione di eventi asincroni

### Esigenze

- ❑ È desiderabile poter reagire velocemente al verificarsi di eventi
  - Asincroni rispetto al flusso di esecuzione (e dunque distinti dalle eccezioni!)
  - Non mappabili su interruzioni *hardware* (che hanno propri meccanismi di gestione)
  - Per i quali il "polling", oltre che indesiderabile, è anche inadeguato
- ❑ Rispetto al trattamento di un errore
  - Il processo gestore potrebbe non arrivare mai al suo punto di "polling"
- ❑ Rispetto al cambiamento dello stato operativo ("mode")
  - Quando questo è risultato di una emergenza
- ❑ Rispetto a calcoli per approssimazione
  - Quando conviene porre un limite temporale a un calcolo approssimato
- ❑ Rispetto a interventi dall'esterno (p.es. dell'operatore)
  - il classico "Ctrl-C"

Corso di Laurea Specialistica in Informatica, Università di Padova 2/18

Università degli Studi di Padova

## Gestione di eventi asincroni

### Una soluzione – 1

- ❑ **Costrutto selettivo asincrono**

```
select
  triggering_alternative
then abort
abortable_part
end select.
```

Richiesta di sincronizzazione su canale tipato (*entry call*)  
 Attesa temporale  
 {+ sequenza opzionale di comandi}

Sequenza di comandi qualunque, esclusa l'accettazione di sincronizzazione

Corso di Laurea Specialistica in Informatica, Università di Padova 3/18

Università degli Studi di Padova

## Gestione di eventi asincroni

### Una soluzione – 2

- ❑ La prima alternativa recepisce l'evento asincrono
- ❑ La seconda alternativa specifica l'esecuzione abbandonabile qualora l'evento asincrono avesse luogo
- ❑ L'esecuzione prima predispone la ricezione dell'evento e poi dà inizio al lavoro
  - Se il lavoro completa prima dell'arrivo dell'evento asincrono, l'attesa viene cancellata e l'esecuzione procede normalmente
  - Altrimenti si finalizza il lavoro, si esegue la sequenza opzionale di comandi di abbandono (se presente) e poi si procede normalmente
  - Vi è *race condition* tra il determinarsi delle due alternative, per questo l'abbandono del lavoro deve avvenire in modo ben ordinato
    - A cura della macchina virtuale che realizza il modello concorrente ma anche del programmatore!

Corso di Laurea Specialistica in Informatica, Università di Padova 4/18

Università degli Studi di Padova

## Gestione di eventi asincroni

### Una soluzione – 3

- ❑ **Il costrutto selettivo asincrono comporta 2 flussi di esecuzione concorrenti tra loro**
  - La realizzazione tipica è infatti detta "two-thread model"

```
protected Error_Manager is
  procedure Notify (Error : Message);
  entry Wait (Error : out Message);
  private
  ...
end Error_Manager;
```

```
loop
  ...
  select
    Error_Manager.Wait(Error);
  case Error is
    when ... =>
      ... -- corrective actions
    end case;
  then abort
  loop
    ... -- normal work
  end loop;
end select;
end loop;
```

Corso di Laurea Specialistica in Informatica, Università di Padova 5/18

Università degli Studi di Padova

## Gestione di eventi asincroni

### Implicazioni – 1

- ❑ Il trasferimento asincrono di controllo (ATC) è sintatticamente semplice ma ha implicazioni molto pesanti sulla complessità del modello
- ❑ L'interazione tra l'evento asincrono e l'esecuzione abbandonabile (EA) diviene particolarmente complessa quando l'uno e/o l'altro sono
  - Attese temporali
  - Attese selettive finite
  - Attese di sincronizzazione

Corso di Laurea Specialistica in Informatica, Università di Padova 6/18

Università degli Studi di Padova Gestione di eventi asincroni

## Implicazioni – 2

**❑ Interazione con attese temporali**

```
task A;
task body A is
T : Time;
D : Duration;
begin
...
select
delay until T;
then abort
delay D;
end select;
end A;
```

```
task B;
task body B is
T : Time;
D : Duration;
begin
...
select
delay D;
then abort
delay until T;
end select;
end B;
```

La EA si completa **solo** quando il processo sia stato risvegliato dalla sua attesa D!

Corso di Laurea Specialistica in Informatica, Università di Padova 7/18

Università degli Studi di Padova Gestione di eventi asincroni

## Implicazioni – 3

**❑ Interazione con attese selettive finite**

```
task A;
task body A is
T : Time;
begin
select
delay until T;
S2;
then abort
Server.Call;
S1;
end select;
end A;
```

```
task B;
task body B is
T : Time;
begin
select
Server.Call;
S1;
then abort
delay until T;
S2;
end select;
end B;
```

```
task C;
task body C is
T : Time;
begin
select
Server.Call;
S1;
or
delay until T;
S2;
end select;
end C;
```

1 Server.Call completa prima di T:  
A inizia a eseguire S1 come EA  
B esegue S1 nell'alternativa asincrona (AA)  
C esegue S1 nell'alternativa di accettazione

2 Server.Call solo inizia prima di T:  
A completa Server.Call e poi esegue S2  
B completa Server.Call e poi esegue S2  
C completa Server.Call e poi esegue S1

3 T scade prima che Server.Call inizi:  
A esegue S2  
B inizia a eseguire S2 come EA  
C esegue S2

Corso di Laurea Specialistica in Informatica, Università di Padova 8/18

Università degli Studi di Padova Gestione di eventi asincroni

## Attese selettive finite

```
delay until T;
Server.Call;
```

	A	B	C
EA	S,C; S1	du T; S2	du T; S2
AA	du T; S2	S,C; S1	S,C; S1

```
delay until T;
Server.Call;
```

	A	B	C
EA	S,C; S1	du T; S2	du T; S2
AA	du T; S2	S,C; S1	S,C; S1

```
delay until T;
Server.Call;
```

	A	B	C
EA	S,C; S1	du T; S2	du T; S2
AA	du T; S2	S,C; S1	S,C; S1

T →

Corso di Laurea Specialistica in Informatica, Università di Padova 9/18

Università degli Studi di Padova Gestione di eventi asincroni

## Implicazioni – 4

**❑ L'ATC rende perfettamente la semantica dell'attesa selettiva finita!**

```
task body C is
T : Time;
begin
Completed := False;
select
delay until T;
then abort
-- Server.Call_ATC (Completed);
-- "Completed" set to True in Server
end select;
if Completed then
S1;
else
S2;
end if;
end C;
```

```
task C;
task body C is
T : Time;
begin
--
select
Server.Call;
S1;
or
delay until T;
S2;
end select;
end C;
```

Chiaro sintomo di ridondanza nel potere espressivo!

Corso di Laurea Specialistica in Informatica, Università di Padova 10/18

Università degli Studi di Padova Gestione di eventi asincroni

## Implicazioni – 5

**❑ Interazione con attese di sincronizzazione**

```
task A;
task body A is
...
begin
...
select
C,E;
then abort
D,E;
end select;
end A;
```

```
task B;
task body B is
...
begin
...
select
D,E;
then abort
C,E;
end select;
end B;
```

**Caso 1 - C,E diventa pronto per primo:**  
A sincronizza con C, ma può anche farlo con D se C,E non completa prima che D,E diventi pronto  
B sincronizza con C, ma può anche farlo con D se C,E non completa prima che D,E diventi pronto

**Caso 2 - D,E diventa pronto per primo:**  
A sincronizza con D, ma può anche farlo con C se D,E non completa prima che C,E diventi pronto  
B sincronizza con D, ma può anche farlo con C se D,E non completa prima che C,E diventi pronto

**Caso 3 - C,E e D,E sono entrambi pronti:**  
A sincronizza solo con C  
B sincronizza solo con D

Corso di Laurea Specialistica in Informatica, Università di Padova 11/18

Università degli Studi di Padova Gestione di eventi asincroni

## Attese di sincronizzazione

```
C,E .....
D,E .....
} 1
```

	A	B
EA	D,E	C,E
AA	C,E	D,E

```
D,E .....
C,E .....
} 2
```

	A	B
EA	D,E	C,E
AA	C,E	D,E

```
D,E .....
C,E .....
} 3
```

	A	B
EA	D,E	C,E
AA	C,E	D,E

Corso di Laurea Specialistica in Informatica, Università di Padova 12/18

Università degli Studi di Padova Gestione di eventi asincroni

## Implicazioni – 6

**❑ Interazione con trasferimenti di coda**

```

task A:
-
task body A is
begin
-
select
B.E1;
then abort
S;
end select;
-
end A;

task B is
entry E1;
entry E2;
end B;

task body B is
begin
-
accept E1 do
-
resqueue E2 (with abort!);
end E1;
-
accept E2 do
-
end E2;
-
end B;
    
```

**Caso 1.1 (Con)** – E1 è subito pronto: S può cominciare solo dopo l'accodamento su E2

**Caso 1.2 (Senza)** – E1 è subito pronto: S non viene eseguito

**Caso 2.1 (Con)** – E1 diventa pronto dopo l'inizio di S: S esegue e B.E1 (inclusa E2) viene cancellata se e quando S completa

**Caso 2.2 (Senza)** – E1 diventa pronto dopo l'inizio di S: S esegue ma il comando `select` di A non completa fin quando E2 non abbia completato

Corso di Laurea Specialistica in Informatica, Università di Padova 13/18

Università degli Studi di Padova Gestione di eventi asincroni

## Abbandono – 1

**❑ In situazioni estreme può essere necessario che un processo abbandoni la propria esecuzione**

- Quando il recupero del processo "guasto" è considerato impossibile
- Questo effetto viene ottenuto tramite comando `abort process_name;`
- Un processo può invocarlo su qualunque altro processo a lui visibile

**❑ Questo evento non deve però causare inconsistenze di stato nel sistema**

Corso di Laurea Specialistica in Informatica, Università di Padova 14/18

Università degli Studi di Padova Gestione di eventi asincroni

## Abbandono – 2

**❑ Il processo nominato nel comando diventa "anormale" e gli viene impedito di interagire con qualunque altro processo**

- Ogni processo non ancora completato e dipendente da tale processo diventa anormale a sua volta

**❑ Quando un processo diventa anormale la sua esecuzione viene immediatamente interrotta a meno che non sia "delicata"**

Corso di Laurea Specialistica in Informatica, Università di Padova 15/18

Università degli Studi di Padova Gestione di eventi asincroni

## Abbandono – 3

**❑ Le azioni "delicate" ritardano l'effetto di un ordine di abbandono fino al loro completamento**

- Ogni esecuzione entro una risorsa protetta
- Ogni sincronizzazione
- Ogni attesa di terminazione di dipendenti
- Ogni finalizzazione

**❑ Un processo in corso di abbandono è comunque in stato "anormale"**

Corso di Laurea Specialistica in Informatica, Università di Padova 16/18

Università degli Studi di Padova Gestione di eventi asincroni

## Critica dell'abbandono

**❑ "An abort statement should be used only in situations requiring unconditional termination."**

- ARM 9.8

**❑ "The existence of this statement causes intolerable overheads in the implementation of every other feature of tasking. Its 'successful' use depends on a valid process aborting a wild one before the wild one aborts a valid process — or does any other serious damage. The probability of this is negligible. If processes can go wild, we are much safer without aborts."**

- C.A.R. Hoare (On Ada 83)

Corso di Laurea Specialistica in Informatica, Università di Padova 17/18

Università degli Studi di Padova Gestione di eventi asincroni

## Stati d'esecuzione di processo

Corso di Laurea Specialistica in Informatica, Università di Padova 18/18