Correttezza temporale

SCD

Anno accademico 2007/8
Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Premesse – 1

- La correttezza funzionale di un programma concorrente non dovrebbe dipendere dall'ordine d'esecuzione dei suoi processi
- Può essere però necessario dimostrare che il non-determinismo dell'esecuzione non possa condurre a situazioni di stallo (*deadlock*) o di avanzamento senza progresso (*livelock*)
- Il modello di concorrenza che abbiamo esaminato finora è non-deterministico rispetto a
  - L'ordinamento d'esecuzione dei processi (*dispatching*)
  - La scelta di una tra più alternative di selezione aperte
  - La scelta di una tra più operazioni possibili entro una risorsa protetta

Premesse – 2

- I sistemi a tempo reale, intrinsecamente concorrenti, devono assicurare correttezza temporale oltre che funzionale
- Ciò richiede che il programma sia capace di esercitare controllo sul proprio grado di non-determinismo
  - Implicitamente : per selezione di politiche di accodamento e/o selezione e assegnazione di attributi (configurazione)
  - Esplicitamente : in forma algoritmica

Politiche di ordinamento

- FPP : *fixed priority preemptive*
  - Originariamente la sola opzione disponibile
    - **pragma** Task_Dispatching_Policy (FIFO_Within_Priorities) : *thread* con priorità uguale sono gestiti con accodamento FIFO
  - Nel seguito assumeremo questa politica!
- FPNP : come sopra senza prerilascio
  - L'arrivo di un *thread* a priorità maggiore non comporta prerilascio
  - Il rilascio è volontario (cooperativo) tramite invocazione di "delay 0.0"
    - **pragma** Task_Dispatching_Policy (Non_Preemptive_FIFO_Within_Priorities)
- RR : *round robin*
  - Quanto predefinito o fissato da Ada.Dispatching.Round_Robin.Set_Quantum (…)
    - **pragma** Task_Dispatching_Policy (Round_Robin_Within_Priorities)
- EDF : *earliest deadline first*
  - I *thread* hanno un attributo "scadenza" (*deadline*) che ne determina l'urgenza
    - Scadenza relativa iniziale (tramite **pragma** Relative_Deadline (…)) e poi assoluta via Ada.Dispatching.EDF.Set_Deadline (…)
    - **pragma** Task_Dispatching_Policy (EDF_Across_Priorities)

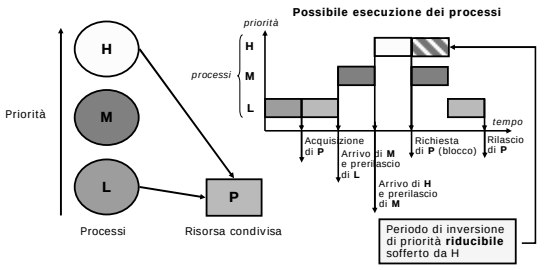
Priorità d'esecuzione – 1

- I processi hanno un attributo predefinito con effetto sull'ordinamento
  - Priorità di base
- Assegnabile tramite comando di configurazione `pragma Priority(N)`
  - Dove `N` è un valore intero in un intervallo fissato
  - Se non configurata esplicitamente la priorità viene assunta essere uguale alla priorità di base del processo padre
  - L'unità `main` a tempo d'esecuzione viene vista come un processo implicito (dunque possibile padre di processi)

Priorità d'esecuzione – 2

- La mutua esclusione in accesso a risorse protette può confliggere con le politiche di ordinamento
- Un regime di ordinamento a priorità si impegna a garantire che, a ogni istante, il processo in esecuzione sia sempre quello a priorità maggiore
- Se ciò non accade la situazione viene detta di "inversione di priorità"
  - Rischio di violazione della proprietà di correttezza temporale

Priorità d'esecuzione – 3



Priorità d'esecuzione – 4

- **Il processo H dell'esempio è ritardato da 2 componenti di inversione di priorità**
  - **Il blocco della risorsa condivisa P da parte di L, meno importante di H**
    - **Durata irriducibile** (intrinseca del modello)
  - **L'esecuzione di M, meno importante di H, ma più importante di L, che ritarda il rilascio di P a discapito di H**
    - **Durata riducibile**
- **Il modello deve essere capace di ridurre al minimo la durata riducibile**
  - **Varie tecniche consentono di farlo con diversa efficacia**
  - **Tutte ispirate all'ereditarietà della priorità maggiore**
  - **Esercizio 13**
    Mostrare come questa tecnica risolva il problema di pagina 6

Priorità d'esecuzione – 5

- **Ereditarietà immediata della priorità più elevata**
  - ***Immediate priority ceiling (inheritance) protocol → ICPI-P***
  - **A ogni risorsa protetta viene attribuita una priorità non inferiore alla priorità maggiore fra quelle dei suoi processi cliente**
    - *Priority ceiling*
  - **Ogni volta che un processo cliente esegue all'interno della risorsa esso assume immediatamente la priorità della risorsa per tutta (e sola) la durata dell'esecuzione protetta**
- **Questa politica azzera la durata riducibile del periodo di inversione di priorità**

Priorità d'esecuzione – 6

- **In ambiente mono-processore ICPI da solo è sufficiente a garantire mutua esclusione**
  - **Quando un processo è in esecuzione entro una risorsa protetta esso esegue per definizione in preferenza a tutti gli altri processi cliente e anche a tutti i processi "non cliente" a priorità non superiore a quella della risorsa**
  - **Priorità della risorsa strettamente maggiore di quella dei suoi clienti**
    - Garanzia assoluta di mutua esclusione
  - **Priorità della risorsa uguale alla maggiore tra quelle dei suoi clienti**
    - Garanzia assoluta di mutua esclusione, ma solo in assenza di prerilascio tra processi a pari priorità

Priorità d'esecuzione – 7

- **ICPI ha altre 2 proprietà importanti in ambiente monoprocessore**
  - **Ogni processo subisce al più 1 ritardo (blocco) da inversione di priorità irriducibile e solo al suo rilascio**
  - **Se tutte le risorse condivise sono accedute sotto regime ICPI e le loro priorità sono coerentemente assegnate allora non si può verificare stallo**
    - **Esercizio 14**: Individuare le precondizioni di stallo impedite da ICPI su monoprocessore
- **Il programma diventa erroneo se un processo tenta di accedere un risorsa avendo priorità superiore a esso (*ceiling violation*)**

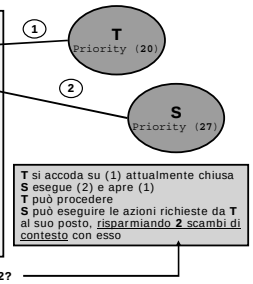
Priorità d'esecuzione – 8

- **La realizzazione del regime ICPI si presta a una interessante ottimizzazione**
  - **Il processo in esecuzione entro una risorsa protetta può eseguire anche le richieste effettuate su di essa da altri processi che si trovino in code con guardia aperta dalle modifiche apportate allo stato della risorsa**
    - ***Proxy model***
- **Questa ottimizzazione**
  - **Preserva l'esecuzione in mutua esclusione nella risorsa**
  - **Riduce l'onere di cambio di contesto tra processi clienti**

Correttezza temporale

Priorità d'esecuzione – 9

```
protected Gate_Control is
 pragma Priority (28);
 entry Stop_And_Close;
 procedure Open;
private
 Gate : Boolean := False;
end Gate_Control;
protected body Gate_Control is
 entry Stop_And_Close when Gate is
 begin
  Gate := False;
 end Stop_And_Close;
 procedure Open is
 begin
  Gate := True;
 end Open;
end Gate_Control;
```



T si accoda su (1) attualmente chiusa
S esegue (2) e apre (1)
T può procedere
S può eseguire le azioni richieste da **T** al suo posto, risparmiando **2** scambi di contesto con esso

Perché 2?

Correttezza temporale

Priorità d'esecuzione – 10

- **L' ereditarietà di priorità (PE) comporta che ogni processo abbia 2 attributi di priorità**
  - **Priorità di base → assegnata alla definizione del processo**
  - **Priorità attiva → a fini di ordinamento, `max`{PB,PE}**
- **Si ha PE**
  - **All'accesso in risorsa protetta**
  - **Durante l'attivazione di un processo figlio il padre ne assume la priorità (se >) per limitare il suo tempo di blocco**
  - **Durante un *rendezvous* il servente assume la priorità del cliente (se >) per la durata della sincronizzazione**
    - Ma il servente esegue alla sua priorità fuori dalla sincronizzazione

Correttezza temporale

Politiche di accodamento

- **Coda dei processi pronti → politica di ordinamento**
  - **A priorità maggiore e FIFO tra priorità uguali**
    - Ogni processo che diventa pronto viene posto in fondo alla coda tra i processi pronti alla sua stessa priorità
    - Un processo in esecuzione scalzato da prerilascio viene posto in testa alla coda dei processi pronti alla sua stessa priorità
- **Coda su canale tipato con guardia**
  - **FIFO all'interno della stessa coda**
  - **A priorità (attiva) tra tutti processi in tutte le code attualmente aperte della stessa entità (servente o risorsa protetta)**
    - Con ciò risolvendo il problema-chiave 4 in ambito di sincronizzazione (C07, pag. 3)
    - Tramite **pragma** Queuing_Policy (...) con argomento FIFO_Queuing o Priority_Queuing