

# Algoritmo del banchiere

Lo stato di un programma può essere:

**safe** per ogni sequenza possibile di richieste, esiste almeno una sequenza di assegnazione delle risorse che garantisce la terminazione;

**unsafe** c'è almeno una sequenza possibile di richieste che porta inevitabilmente ad uno stallo;

**compromesso/doomed** si verifica sicuramente uno stallo;

**in deadlock**

L'algoritmo del banchiere impedisce di lasciare lo stato safe agendo sull'allocazione delle risorse (non sullo scheduling), nell'ipotesi che per ciascun thread si conosca l'insieme di risorse che quel thread può richiedere.

Il banchiere, per ogni richiesta, determina se soddisfarla porta ad uno stato safe ordinando i thread per richiesta residua crescente e verificando se assegnando le risorse in quell'ordine il programma termina. Se non è così, il thread richiedente viene messo in attesa.

È una versione più efficiente dell'approccio *acquire-all*, in cui i thread attendono che tutte le risorse di cui hanno bisogno siano disponibili e le acquisiscono atomicamente tutte insieme.

## Caso più risorse

Dato il vettore degli assegnamenti al processo  $i$   $a_i$ , delle richieste residue  $r_i$ , e delle disponibilità  $d$ ,

- si cerca un processo non marcato con  $r_i \leq d$  (componente per componente);
- se si trova, si marca e  $d = d + a_i$ ;
- in caso contrario, se tutti i processi sono marcati lo stato è safe, altrimenti unsafe.