

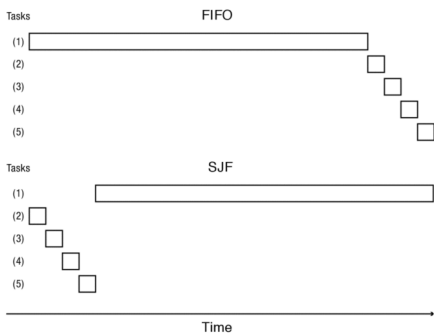
# Scheduling su uniprocessore

## FIFO

Esecuzione dei task in ordine di arrivo, fino al completamento o rilascio della CPU. Ha overhead minimo, e funziona bene se i task hanno durata simile, ma se si presenta un task lungo seguito da task brevi, la latenza di questi aumenta significativamente.

## Shortest Job First

Se si implementa anche prerilascio, shortest remaining time first (SRTF). È ottimo rispetto al tempo medio di risposta.



FIFO:  $\frac{100+101+102+103+104}{5} = 102$ , SJF:  $\frac{1+2+3+4+104}{5} = 22$ .

Problemi:

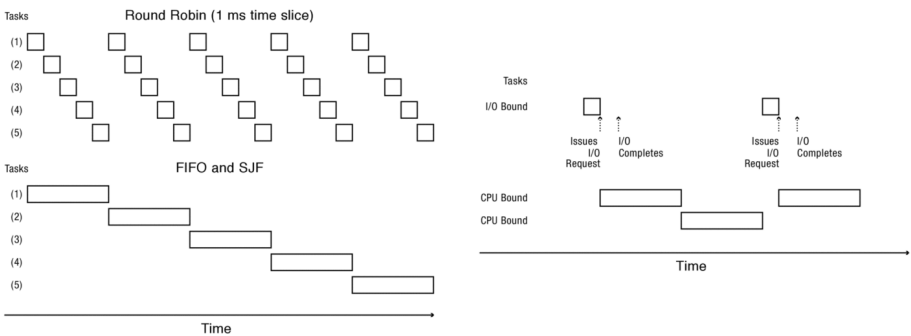
- nella pratica non sappiamo a priori la lunghezza dei task;
- bassa fairness, i task lunghi possono essere rimandati indefinitamente;
- pessimo per la varianza del tempo di risposta – task brevi eseguiti più velocemente possibile, lunghi più lentamente possibile.

## Round robin

I task sono eseguiti in ordine per un quanto di tempo prestabilito (20-120 μs, troppo alto diventa FIFO, basso introduce overhead eccessivo). Il tempo di risposta è limitato da ( $\#$ processi pronti · quanto di tempo).

Problemi:

- latenza media alta con task di durata simile;
- non fair con processi misti I/O e CPU bound: i task I/O hanno bisogno di latenza minore, e completano raramente il quanto di tempo, finendo a parità di lavoro svolto più volte in fondo alla coda.



## Max-Min Fairness

Massimizzare l’allocazione minima di risorse per ciascun processo.

Se i context switch avessero costo zero potremmo ad ogni istante mandare in esecuzione il processo che ha ricevuto meno tempo di CPU. I processi I/O-bound tenderebbero a forzare il prerilascio da parte di processi CPU-bound. Non è pratico perché probabilmente portebbe a cambiare processo ad ogni istruzione.

Si può approssimare questo comportamento permettendo ai processi di superare la propria giusta parte di un quanto di tempo. Quando scade, lo scheduler manda in esecuzione il processo con tempo di CPU accumulato minimo. Si può prerilasciare la CPU se arriva un processo con tempo accumulato molto minore di quello in esecuzione. Anche questa approssimazione però ha overhead alto, visto che richiede una coda di priorità.

## MFQ

Altra flashcard.