Sistemi Operativi (M-Z)

C.d.L. in Informatica
(Laurea Triennale)
A.A. 2023-2024

Scheduler & Algoritmi di Scheduling

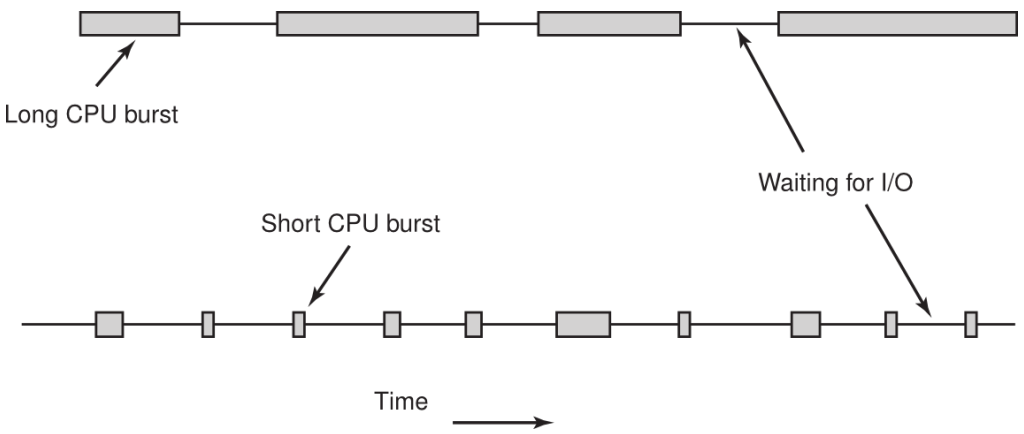
Prof. Mario F. Pavone

Dipartimento di Matematica e Informatica
Università degli Studi di Catania
mario.pavone@unict.it

Scheduling

- Quale processo/thread eseguire successivamente?
- **Scheduler & Algoritmo di Scheduling**
- Vecchi PC vs. Nuovi PC
- **Scheduler**:
  - effettua la scelta a secondo ***tipo di macchina***
  - goal: uso efficiente CPU

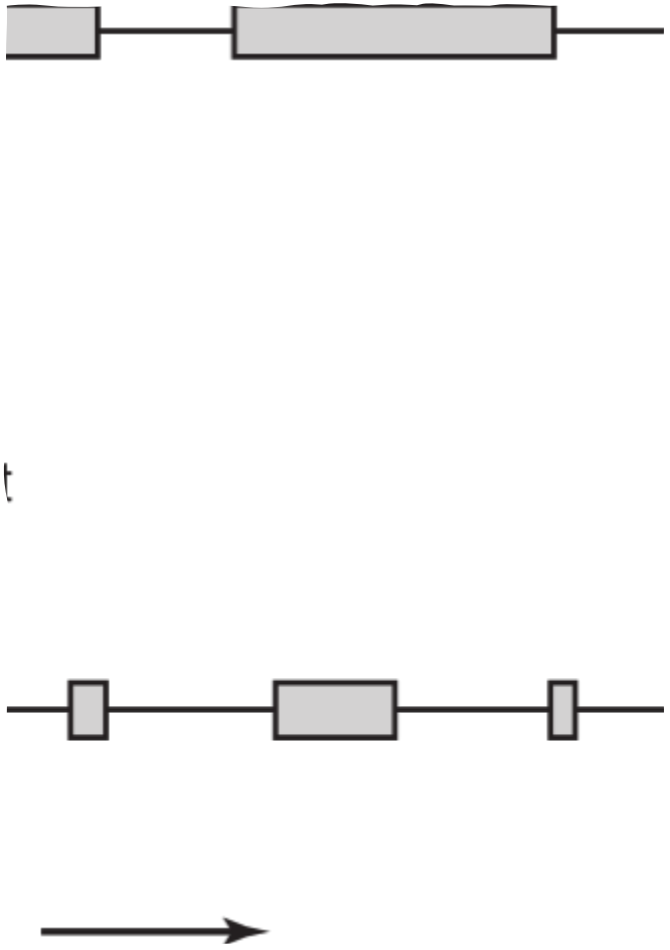
Scheduling



CPU veloci => I/O bound

- Esecuzione si alterna con richieste di I/O
- tipologie di processi:
  - **CPU-bounded** => burst CPU lunghi e attese I/O poco frequenti
  - **I/O-bounded** => burst CPU brevi e attese I/O frequenti

Scheduling



- **quando** viene attivato lo scheduler:
  - **terminazione** e **creazione** di processi;
  - **chiamata bloccante** (es., I/O) e arrivo del relativo interrupt;
  - **blocco I/O** => può influire sulla scelta del successivo
  - interrupt periodici:
    - sistemi **non-preemptive** (senza prelazione);
    - sistemi **preemptive** (con prelazione);
- collabora con il **dispatcher**:
  - essere il più veloce possibile
  - **latenza di dispatch**.

Obiettivi degli algoritmi di scheduling

- Ambienti differenti:
  - **Batch**: non-preemptive
  - **Interattivi**: preemptive
  - **real-time**: preemptive non sempre necessaria
- **Obiettivi comuni**:
  - **equità** nell'assegnazione della CPU;
    - processi comparabili devono avere servizi comparabili
  - **bilanciamento** nell'uso delle risorse;
    - tutte le parti del sistema devono essere impegnate

Obiettivi degli algoritmi di scheduling (metriche prestazioni)

- Obiettivi tipici dei ***sistemi batch***:
  - massimizzare il **throughput** (o produttività);
  - minimizzare il **tempo di turnaround** (o tempo di completamento);
  - minimizzare il **tempo di attesa**;
- **Massimizzare throughput non necessarimante minimizza il tempo di turnaround**
- Obiettivi tipici dei ***sistemi interattivi***:
  - minimizzare il **tempo di risposta**;
- Obiettivi tipici dei ***sistemi real-time***:
  - **rispetto delle scadenze**;
  - **prevedibilità**.

Scheduling nei sistemi batch

- **First-Come First-Served** (FCFS) o per ordine di arrivo;
  - **non-preemptive**;
  - semplice coda **FIFO**.
- **Shortest Job First** (SJF) o per brevità:
  - **non-preemptive**;
  - presuppone la **conoscenza del tempo impiegato** da ogni lavoro;
  - ottimale solo se i lavori sono tutti subito disponibili.
- **Shortest Remaining Time Next** (SRTN):
  - **versione preemptive dello SJF**

In generale: (4a+3b+2c+d)/4

Esempio

Processo	Durata
P_1	24
P_2	3
P_3	3

t.m.a.: (0+24+27)/3 = 17
t.m.c.: (24+27+30)/3 = 27

Esempio
SJF non è ottimale

Processo	Arrivo	Durata
P_1	0	2
P_2	0	4
P_3	3	1
P_4	3	1
P_5	3	1

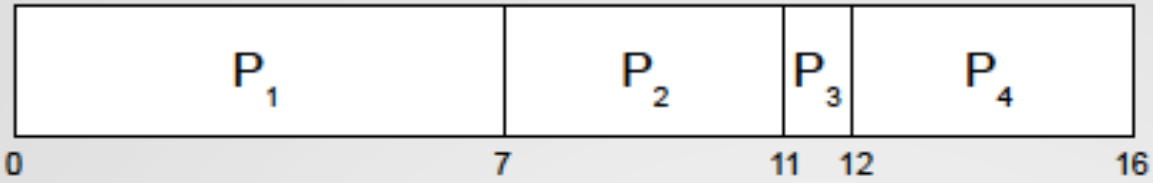
	t.m.a.
SJF	(0+2+3+4+5)/5 = 2.8
altern.	(7+0+1+2+3)/5 = 2.6

Processo	Arrivo	Durata
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Scheduling nei sistemi batch

FCFS vs. SJF vs. SRTN

- **FCFS:**



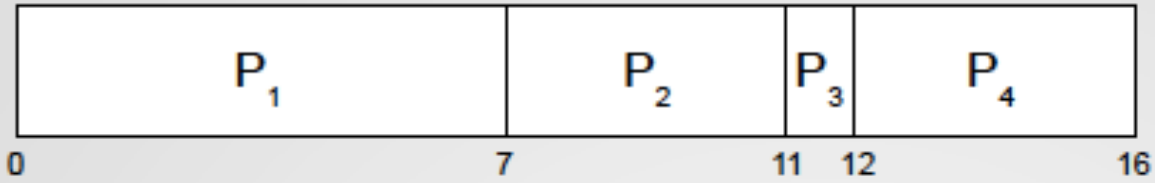
Processo	Arrivo	Durata
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

  - tempi di attesa: P_1=0; P_2=5; P_3=7; P_4=7 (media 4.75);
  - tempi di completamento: P_1=7; P_2=9; P_3=8; P_4=11 (media 8.75);

Scheduling nei sistemi batch

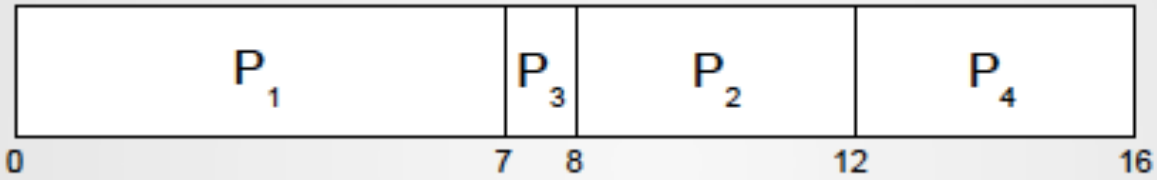
FCFS vs. SJF vs. SRTN

- **FCFS:**



Processo	Arrivo	Durata
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

  - tempi di attesa: P_1=0; P_2=5; P_3=7; P_4=7 (media 4.75);
  - tempi di completamento: P_1=7; P_2=9; P_3=8; P_4=11 (media 8.75);
- **SJF:**

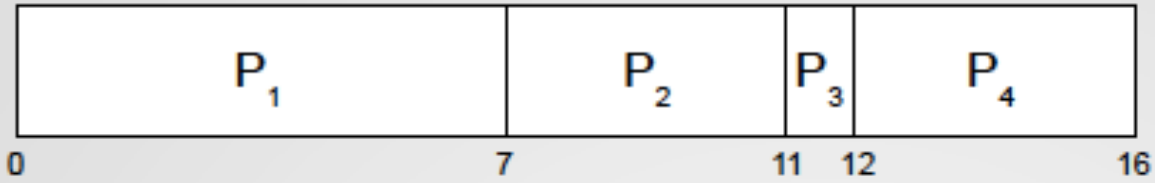


  - tempi di attesa: P_1=0; P_2=6; P_3=3; P_4=7 (media 4);
  - tempi di completamento: P_1=7; P_2=10; P_3=4; P_4=11 (media 8);

Scheduling nei sistemi batch

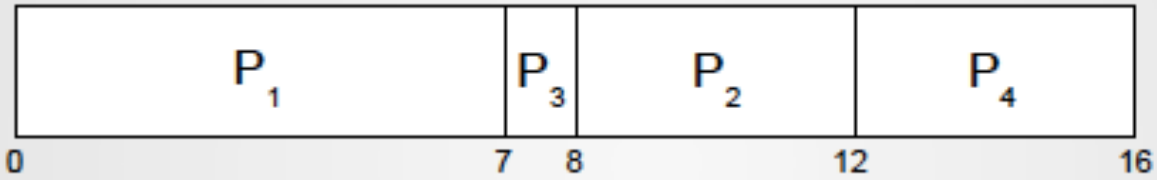
FCFS vs. SJF vs. SRTN

- **FCFS:**

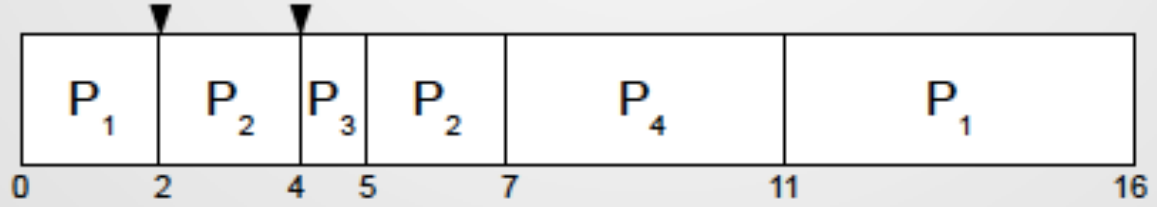


Processo	Arrivo	Durata
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

  - tempi di attesa: P_1=0; P_2=5; P_3=7; P_4=7 (media 4.75);
  - tempi di completamento: P_1=7; P_2=9; P_3=8; P_4=11 (media 8.75);
- **SJF:**



  - tempi di attesa: P_1=0; P_2=6; P_3=3; P_4=7 (media 4);
  - tempi di completamento: P_1=7; P_2=10; P_3=4; P_4=11 (media 8);
- **SRTN:**



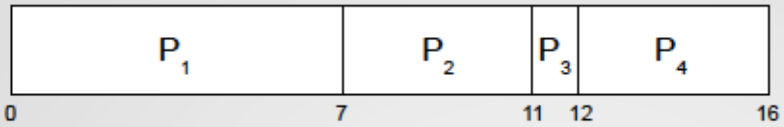
Scheduling nei sistemi batch

FCFS vs. SJF vs. SRTN

Diagrammi di Gantt

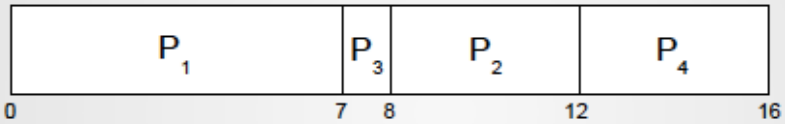
Processo	Arrivo	Durata
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- **FCFS**:



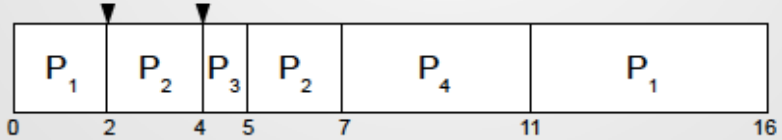
  - tempi di attesa: P_1=0; P_2=5; P_3=7; P_4=7 (media 4.75);
  - tempi di completamento: P_1=7; P_2=9; P_3=8; P_4=11 (media 8.75);

- **SJF**:



  - tempi di attesa: P_1=0; P_2=6; P_3=3; P_4=7 (media 4);
  - tempi di completamento: P_1=7; P_2=10; P_3=4; P_4=11 (media 8);

- **SRTN**:



  - tempi di attesa: P_1=9; P_2=1; P_3=0; P_4=2 (media 3);
  - tempi di completamento: P_1=16; P_2=5; P_3=1; P_4=6 (media 7).

Scheduling nei sistemi interattivi

- **Scheduling Round-Robin** (RR):
  - versione con prelazione del FCFS;
  - **preemptive** e basato su un **quanto di tempo** (timeslice);
  - mantenere una lista di processi eseguibili

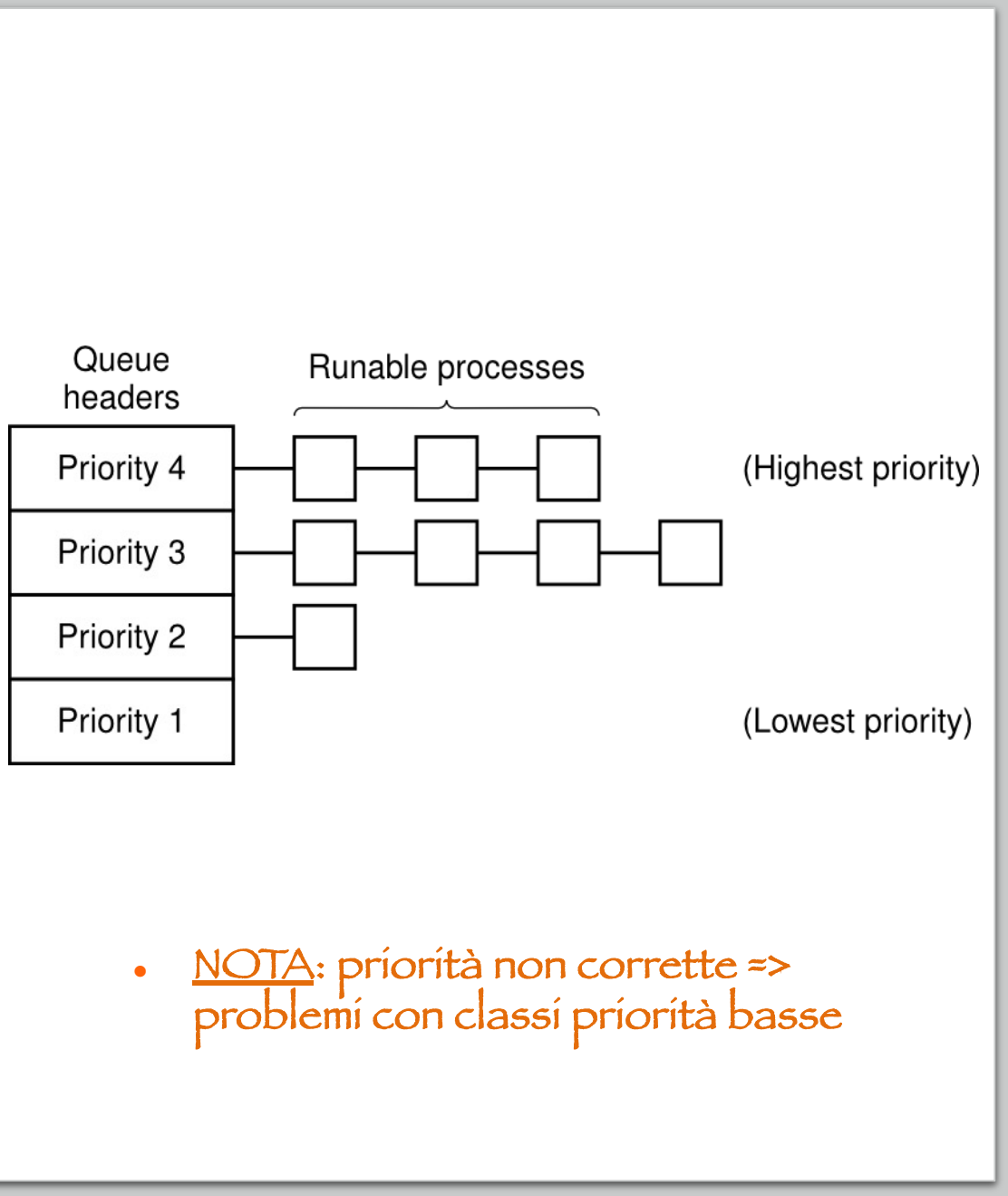
  - quanto deve essere lungo il timeslice?
    - **process switch** or **context switch**
    - valori tipici sono 20-50ms;
  - con ***n*** processi e un quanto di ***q millisecondi***, ogni processo avrà diritto a circa ***1/n*** della CPU e attenderà al più ***(n-1)q ms***

Scheduling nei sistemi interattivi

- **Vincolo RR: tutti i processi con uguale importanza**
- **Scheduling a priorità**:
  - **regola di base**: si assegna la CPU al processo con più alta priorità;
  - **assegnamento delle priorità**:
    - **statiche** o **dinamiche**
      - **comando *nice* in Unix**
    - **favorire processi I/O bounded**
      - **priorità *1/f*, con *f* = frazione quanto utilizzato**
    - **SJF come sistema a priorità**
      - **priority: inverse task length**

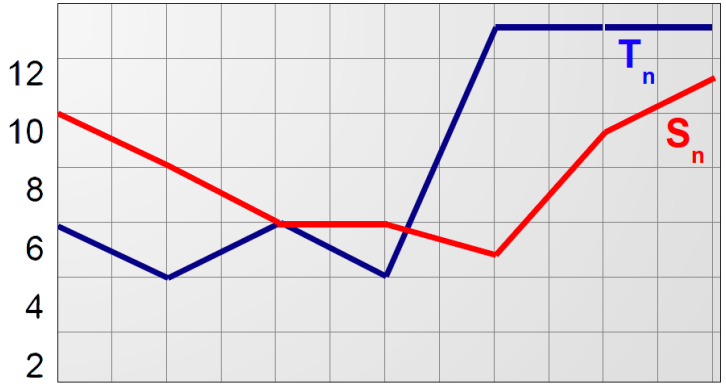
Scheduling nei sistemi interattivi

- **prelazione *vs.* non-prelazione** (ex. SJF vs. SRTN)
- **possibile problema**: **starvation**
- **possibile soluzione**: **aging**
  - **Incremento graduale della priorità**
- **Classi di Priorità**
  - **Scheduling di priorità tra le classi**
  - **Scheduling Round-Robin all'interno delle classi**



- NOTA: priorità non corrette => problemi con classi priorità basse

Scheduling nei sistemi interattivi



- **Shortest Process Next (SPN)**:
  - **idea**: applicare lo **SJF ai processi interattivi**;
  - **problema**: **identificare** la durata del prossimo burst di CPU;
  - **soluzione**: **stime** basate sui burst precedenti;

$$S_{n+1} = S_n(1-a) + T_n a \qquad 0 \le a \le 1$$

  - ***a* determina la stima lungo, media o corta**

- esempio: **a=1/2**

T_n	6	4	6	4	13	13	13
S_n	10	8	6	6	5	9	11

Scheduling nei sistemi interattivi

- Scheduling garantito:
  - fare promesse reali e mantenerle
  - stabilire una percentuale di utilizzo e rispettarla.
  - n utenti connessi avranno 1/n della potenza CPU (idem processi)
  - calcolare quantità CPU spettante
    - (T_c / n) = tempo dalla creazione diviso n
  - tenere traccia quanta CPU ricevuta
  - l'algoritmo esegue il processo con rapporto minore

Scheduling nei sistemi interattivi

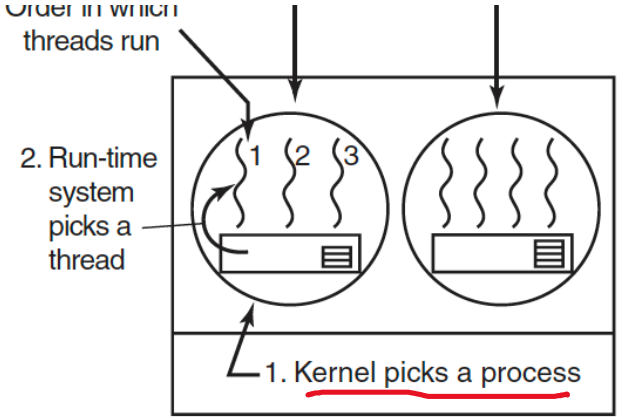
- **Scheduling a lotteria**:
  - idea base: **biglietti con estrazioni a random**
    - **non lo stesso numero di biglietti**
    - **processo da eseguire con estrazione**
  - processi importanti: **biglietti extra**
  - **criterio semplice e chiaro**
  - reagisce velocemente ai cambiamenti
  - possibilità di avere **processi cooperanti**
  - **risolvere problemi difficili da gestire con altri metodi**

Scheduling nei sistemi interattivi

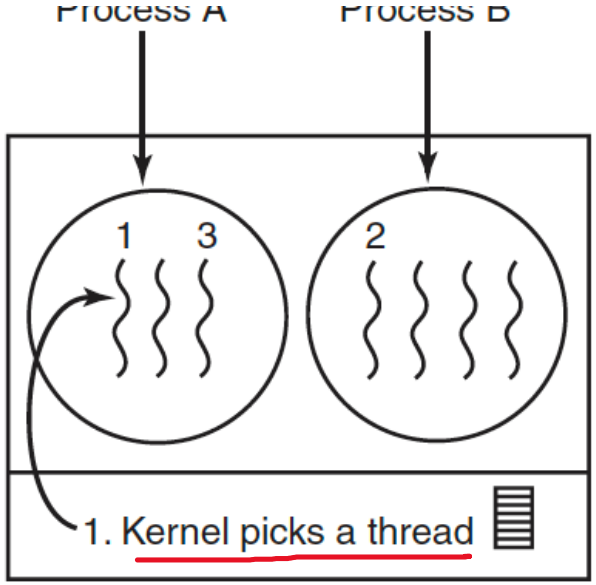
- Scheduling fair-share:
  - considerare a chi appartiene un processo
  - realizza un equo uso tra gli utenti del sistema
  - un utente con più processi avrà più "attenzione"
  - considerare la proprietà dei processi durante la schedulazione
    - a prescindere dal numero di processi

Scheduling dei Tthread

- Thread utente:
  - ignorati dallo scheduler del kernel
  - per lo scheduler del sistema run-time vanno bene tutti gli algoritmi non-preemptive visti
    - unica restrizione: assenza interrupt clock
  - possibilità di utilizzo di scheduling personalizzato
- Thread del kernel:
  - o si considerano tutti i thread uguali, oppure;
  - si pesa l'appartenenza al processo;
    - switch su un thread di un processo diverso implica anche la riprogrammazione della MMU e, in alcuni casi, l'azzeramento della cache della CPU.



Possible: A1, A2, A3, A1, A2, A3
Not possible: A1, B1, A2, B2, A3, B3



Possible: A1, A2, A3, A1, A2, A3
Also possible: A1, B1, A2, B2, A3, B3

(b)

Thread Utente vs. Thread Kernel (differenze)

- Prestazioni:
  - cambio di contesto più lento
  - livello kernel: thread bloccato non sospende il processo
    - no livello utente
  - informazione su processo proprietario thread
    - thread processo B più costoso secondo thread processo A
  - thread utente: scheduler specifico dell'applicazione.

Scheduling su sistemi multiprocessore

- multielaborazione asimmetrica
  - uno dei processori assume il ruolo di master server;
- multielaborazione simmetrica (SMP);
  - coda unificata dei processi pronti o code separate per ogni processore/core
- Politiche di scheduling:
  - Gestione della cache
  - presenza o assenza di predilezione per i processori:
    - predilezione debole: supporta ma non garantisce
    - predilezione forte: forza un processo ad un dato processore

Scheduling su sistemi multiprocessore

- Bilanciamento del carico:
  - Avvantaggiarsi di **avere più processori**
  - **Ugualmente distribuito**
  - necessaria solo in **presenza di code distinte** per i processi pronti;
  - **migrazione guidata** (*push migration*) o **migrazione spontanea** (*pull migration*);
    - possibili **approcci misti** (Linux e FreeBSD);
  - **bilanciamento del carico *vs.* predilezione del processore**.

Cosa usano i nostri Sistemi Operativi

- elementi comuni:
  - thread, SMP, gestione priorità, predilezione per i processi IO-bounded
- **Windows**:
  - scheduler basato su code di priorità con varie;
  - euristiche per migliorare il servizio dei processio interattivi e in particolare di foreground;
  - euristiche per evitare il problema dell'inversione di priorità.
- **Linux**:
  - scheduling basato su task (generalizzazione di processi e thread);
  - Completely Fair Scheduler (CFS): moderno scheduler garantito;
- **MacOS**:
  - Mach scheduler basato su code di priorità con euristiche.