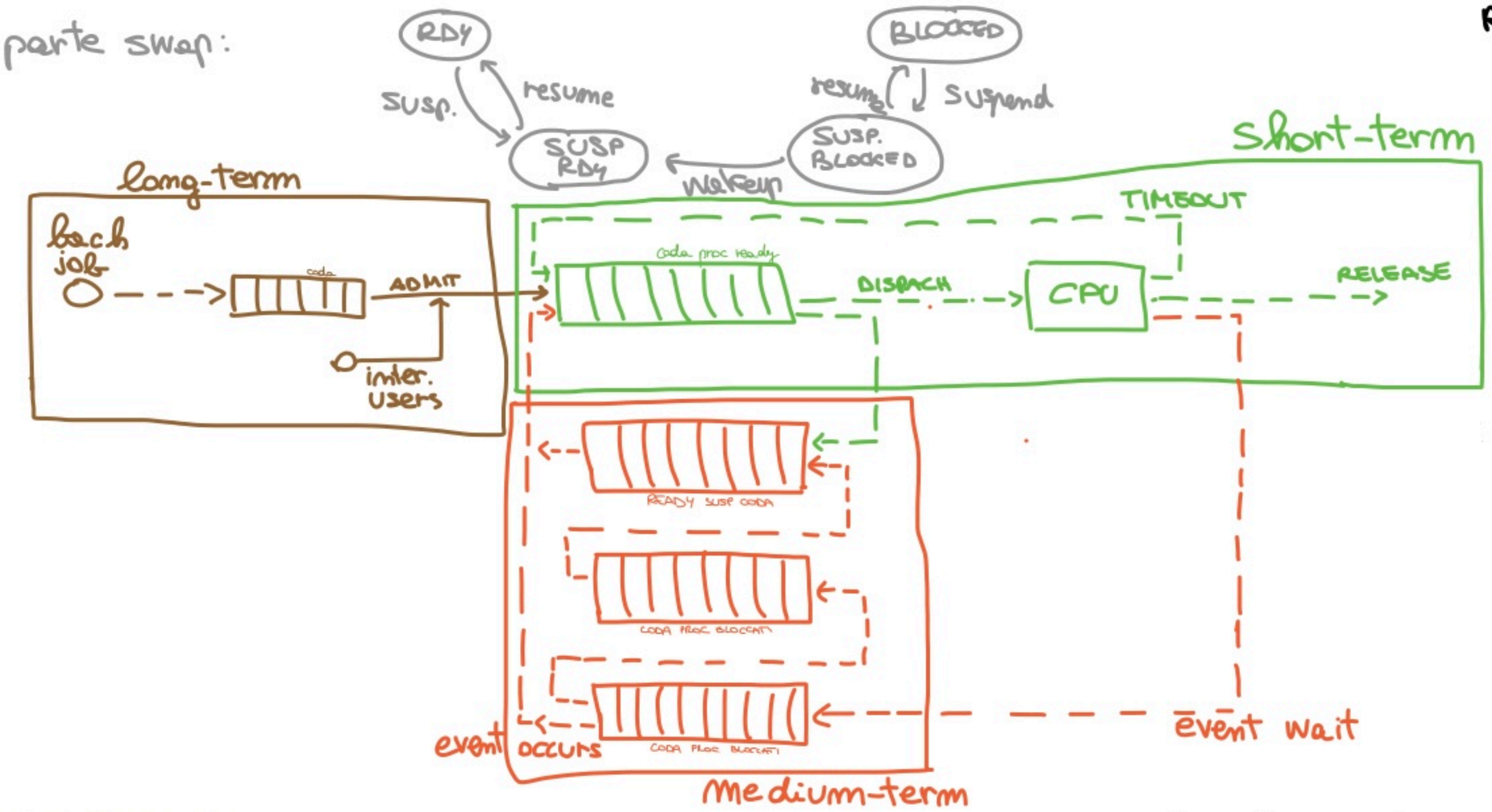
SCHEDULER

→ componente che si occupa di **decidere** quale programma eseguire in ogni istante.

→ ne esistono diverse tipologie:
- → massimizzano throughput;
- → minimizzano latenza proc;
- → evitare che un proc. venga ignorato;
- → completare entro un tempo def;
- → massimiz. la percentuale di utilizzo del proc;

→ esistono anche diversi livelli:
- **long-term**, sceglie quali iniz: NEW → ..... → EXIT ← n° processi generale
- **short-term**, azioni molto rapide: READY ↔ RUN → BLOCK ← le priorità
- **medium**, gestisce la parte di swap: ↑↓ SUSP. READY ← ↓↑ SUSP. BLOCKED

parte swap:



→ **dispatcher**: scheduler di breve periodo, eseguito più freq. Viene invocato da: → interrupt; → sys-calls; → segnale;

↳ i criteri possibili sono → user-oriented, minim. il 'response time'
→ sys-oriented, utilizzo effic. processore

POLITICHE DI SCHEDULING

- **preemptive**, possibilità di interrompere il processo in esec; migliorano tempo risp
- **non-preemptive**, opposto a sopra; lento, rischia di bloccare proc (deadlock)

→ la decisione di eseg. un programma piuttosto di un altro è data da priorità assegnate:
- → statica, non cambia, basso overhead e inefficiente;
- → dinamica, maggiore overhead ma più reattiva;

FIFO o FCFS

↳ non-preemptive, processi serviti in base all'arrivo: → sceglie il più vecchio

↳ non è alg. princip.

$T_{ATT} = T_{FINE} - T_{ARR} - T_{S.R}$

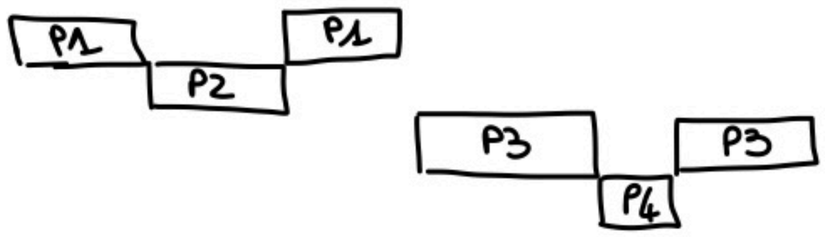
✗ rallenta proc. brevi e rischio deadlock

✓ favorisce proc. CPU-bound (sfruttano pesantem. proc)

RR- Round Robin

↳ criterio FIFO, preemptive (∃ quanto di tempo detto 'timeslice') quindi superato questo tempo viene interrotto e messo in coda:

↳ non è alg. primap



SRR- Selfish RR

↳ a differenza di sopra i processi hanno delle priorità (in base alla vecchiaia) utilizzando 2 code: P entra in Coda → invecchia → aumenta priorità → diventa come quella dei proc ready

$V_{new} \geq V_{ready}$

velocità invecc. nella coda dei nuovi

velocità inv. nella coda dei ready

↳ se $V_{new} = V_{ready}$ allora degenera in una FIFO

↳ la durata del quanto di tempo per SRR-RR deve essere media in modo da non frammentare troppo i processi ma neanche troppo lunga da diventare FIFO

SJF - Shorted Job First

→ fase in cui viene impiegata solo la CPU

→ associa ad ogni processo la lunghezza del prox CPU burst e sceglie quello a burst minore;

→ può essere sia non-preemp che preemptive, ovvero un processo può essere interrotto se ne arriva uno con burst minore che lo 'scavalca'; SRTF (short-rem-time-first)

→ SJF minimizza il tempo di attesa medio di tutti i processi;

→ STIMA PROSSIMO BURST: $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$ → coeff compreso [0,1]

t prox. burst; t burst corrente

SPN - Shortest process next

↳ non-preemptive, processa prima il processo con runtime stimato minore

↳ può dare problemi di starvation

HRRN - Highest Resp Ratio Next

↳ risolve prob. SJF-SPN, non-preemptive: calcola priorità ed esegue fino alla fine

→ PRIORITÀ: $P = \frac{T_{ATT} + T_{EXEC}}{T_{EXEC}}$ favorisce proc. brevi e che attendono da molto

SRT - Shortest Remaining Time

↳ SPN preemptive con stima tempi di esec., ovvero riprende sempre quelli che finisce prima e non da priorità

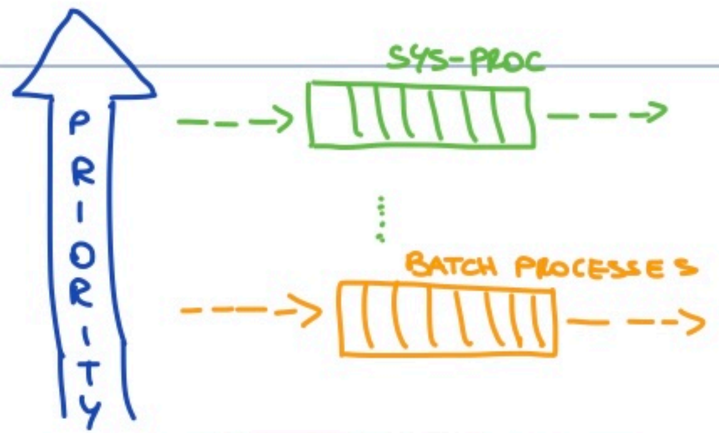
Multilev Queues

↳ coda processi ready → foreground (interattivi) → usa alg. **RR**

→ background (batch) → usa **FCFS**

esec. non accorpata prog.

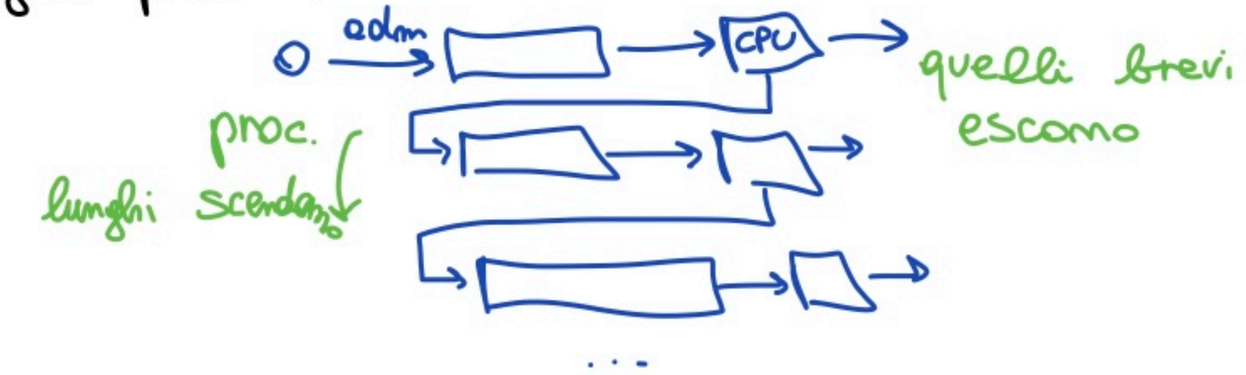
→ bisogna eseguire uno scheduling anche tra code: • prima foreg. poi backg;
• ed ogni coda un tempo CPU fissato



Multilev Feedback Queues

↳ politiche FIFO o RR

↳ nuovi processi entrano nella coda più alta (maggior priorità);
i più lunghi scendono di coda (**CPU-bound**) mentre i più brevi o **I/O-bound** rimangono a maggior priorità



→ definito da n° code, alg. per ogni coda, criteri processi;