Tutoriat SO

① q = foarte mic (exemplu=1) → foarte ineficient
context switch Rus

q = foarte mare ⇒ am avea un FIFO

Unde punem ~~pauza~~ cuanta? q > 80% din toate procesele
Ca să nu avem starvation

La context switch se schimbă PCB-ul

② Într-un mediu ideal, alg. cel mai bun ar fi SJF preemptive / SRTF

③ FCFS (first come served first served)
- se execută multe procese lungi, iar cele mai mici au wait time prea mare.

SJF (shortest job first scheduling) non-preemptive
- trb să avem cât mai puține context-switch-uri

exemplu

P₁ ⊥ 1000
P₂ ⊥ 1000
P₃ ⊥ 1000

q = 500

P₁ - wait 1000
P₂ - 1500
P₃ - 1000+1000 = 2000

~~[illegible]~~
450013

P₁	P₂	P₃	P₁	P₂	P₃
0 – 500	501 – 1000	1001 – 1500	2000	2500	

alt exemplu : 1000
500
2000
1
4000

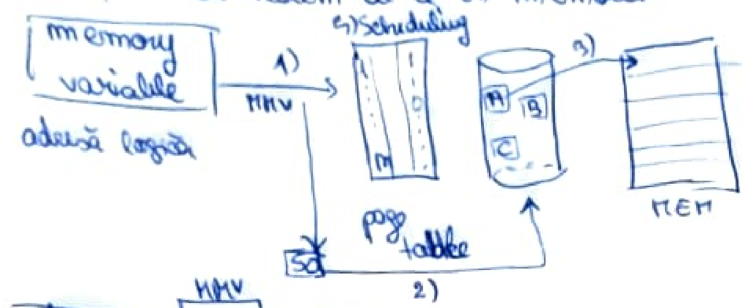
RR (Round Robin)
- trb să nu dea în niciuna de mai sus

Priority + RR
- aging ⇒ fără starvation

(4) Ce se întâmplă când avem un page fault?

trb să accesăm memoria RAM ca să

1) Încercăm să vedem de ce în memorie



loc memorie

Threshing = faci mai multe calcule să iei din disk să pui în RAM decât să faci calcule să fol. variabila

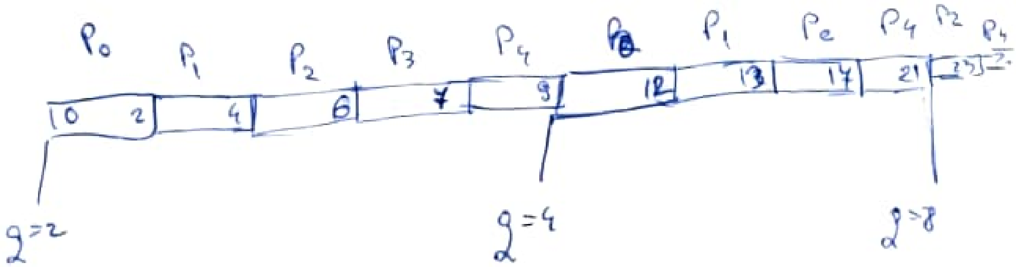
= faci asta în continuu

(5) ~~term~~ =
Dacă elimini „term" / stofela ⇒ nu va mai fi executat niciodată

Schema RR (?)

(6)

	t_ajuns	burst time
P_0	0	~~5~~ ~~3~~ 0
P_1	0	~~3~~ ~~1~~ 0
P_2	(2)	~~8~~ ~~6~~ 2
P_3	4	~~1~~ 0
P_4	6	~~7~~ ~~5~~ 1



RR=? $q=2$

Se dublează ⇒ dacă un proces nu se termină în cuanta de timp → se dublează.

$(n-1) \cdot q$ ⇒ cât de mult așteaptă un proces

Average waiting time pentru P_2 (index de la 2) → $2+7+6 = 15$

Dacă waiting-time-ul $\geq (m-1)\,l$ ⇒ alg e greşit

F.A.T.

(1)

block = 128 bytes
pointer are = 2 bytes
126 bytes storare per block

| ⇒ Linked list

Tabela FAT nu e același lucru cu directory mesh ce (?)

12 caractere nume
2 bytes → pointer

| ⇒ 14 bytes o linie director

↳ Împărțim e vedem câte încap într-un block