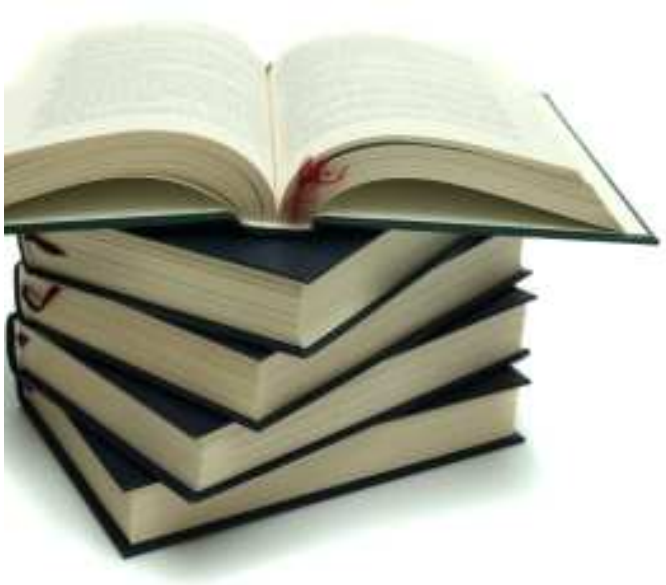
page crunch

Cursul 4



4

Planificarea execuției

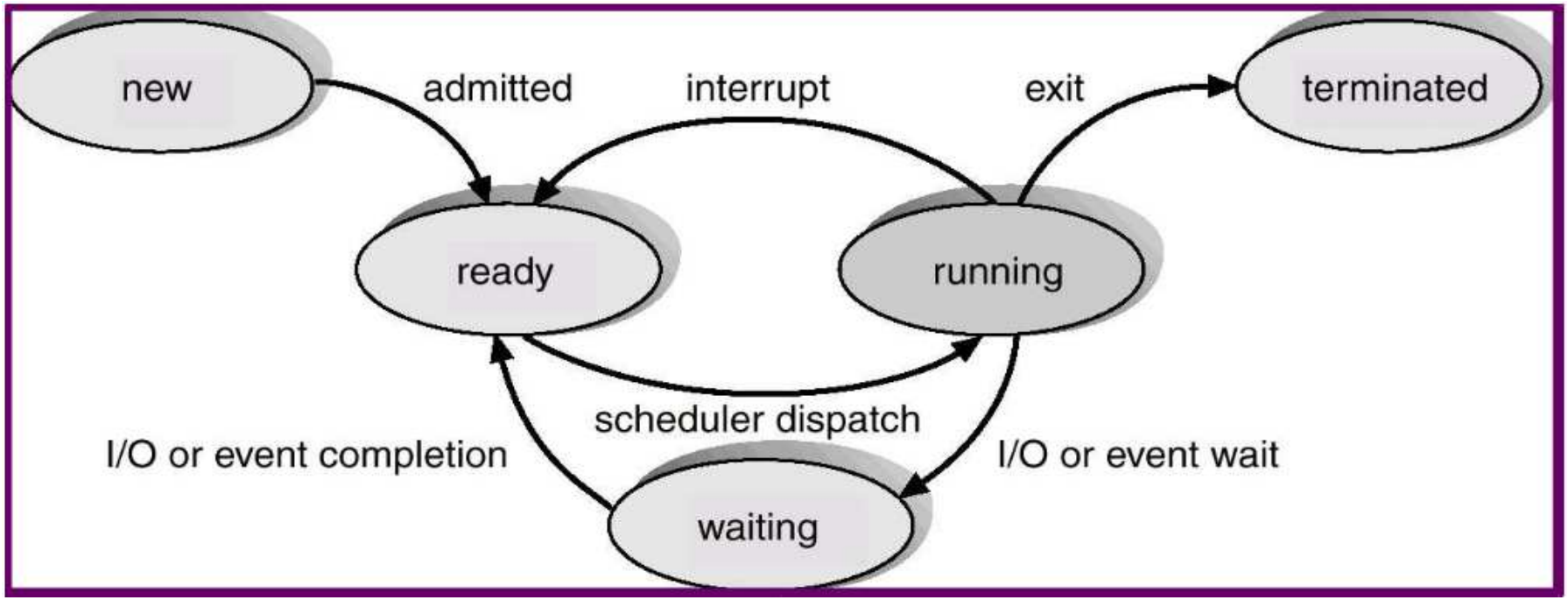
18 martie 2009

- OSC
  - Capitolul 5 – CPU Scheduling
- MOS
  - Capitolul 2 – Processes and Threads
    - Secțiunea 5 - Scheduling

Cuprins

- Tipuri/criterii/algoritmi de planificare
- Planificare pentru sisteme batch
- Planificare pentru sisteme interactive
- Planificare pentru sisteme real-time
- Planificarea execuției în Windows
- Planificarea execuției în Linux

Reminder – stările unui proces

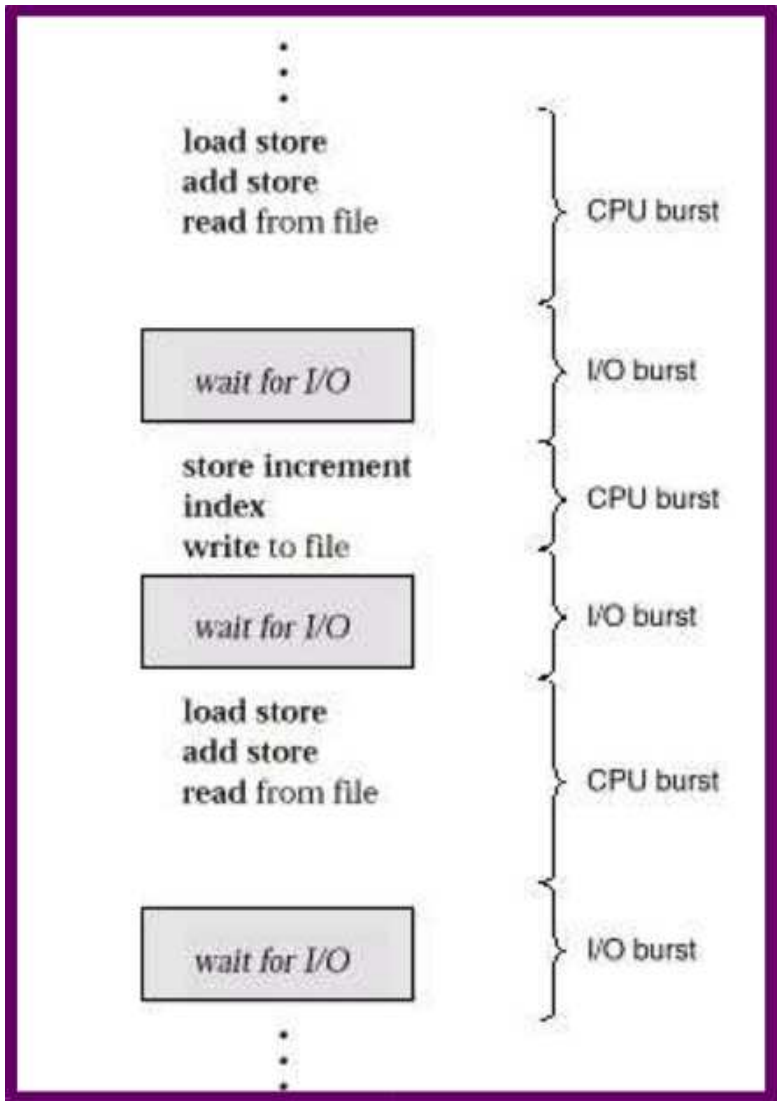


- READY – gata de execuție
- RUNNING – rulează
- WAITING – așteaptă încheierea unei operații de I/E

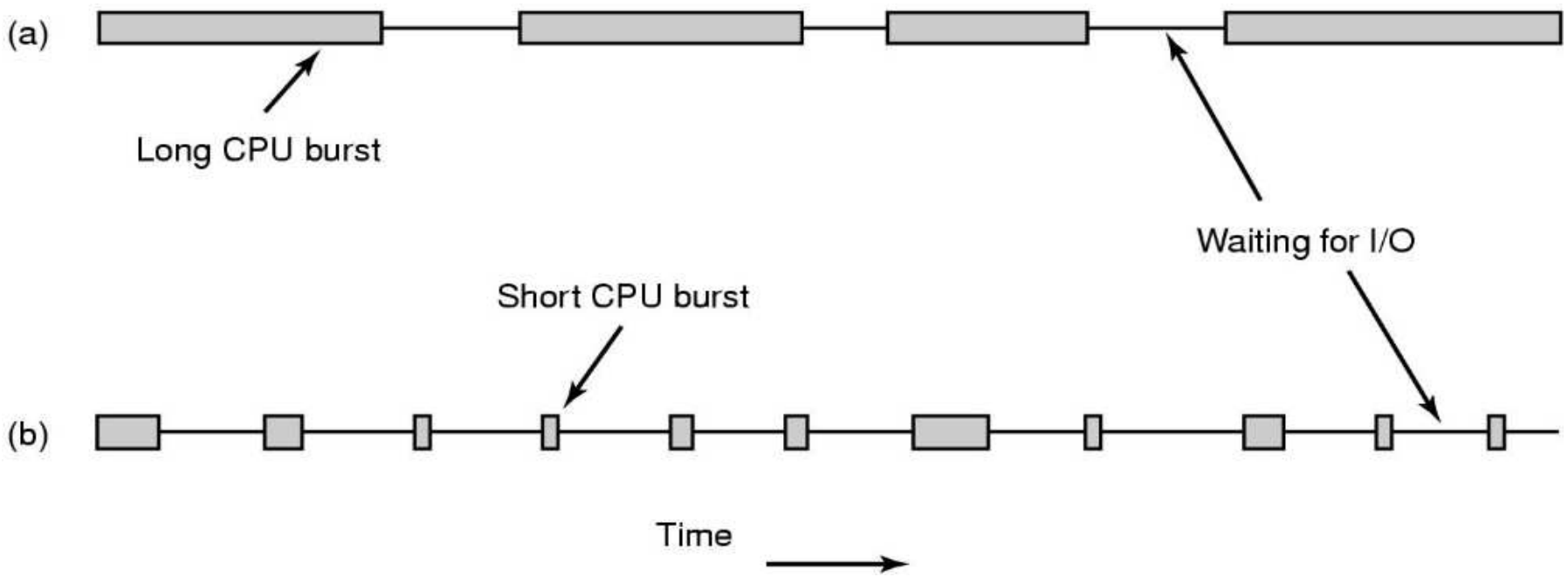
Planificarea proceselor

- (CPU/Process) Scheduling
  - alocarea proceselor pe procesor/procesoare
- Suportul pentru multitasking
- Scheduler
  - subsistem al SO
  - algoritmi de planificare

Comportarea proceselor



Comportarea proceselor



(a) procese CPU-bound

(b) procese IO-bound

De ce planificarea execuției?

- Eficiență – optimizare acces la procesor
  - un proces este executat până când așteaptă un eveniment blocant sau îi expiră cuanta
  - SO (scheduler-ul) alege alt proces pentru execuție (planifică) după un procedeu (algoritm de planificare)

- Interactivitate și corectitudine (fairness)
  - procesele pot fi întrerupte pentru rularea altor procese cu prioritate mai mare
  - toate procesele au șanse egale de obținere a a procesorului
    - în cazul planificării cu priorități, unele procese sunt „mai egale”

Tipuri de planificare a execuției

- Cooperativă/preemptivă
- Cooperativă
  - se face planificare doar atunci când procesul se blochează, se termină sau cedează de bună voie procesorul
- Preemptivă
  - procesele rulează un interval maxim de timp (cuantă)
  - procesul este suspendat și se planifică altul
  - este necesară o întrerupere de ceas

Când se apelează planificatorul?

- Crearea unui proces
- Terminarea unui proces
- Blocarea unui proces
  - operații de I/E
  - operații de sincronizare cu alte procese
- Terminarea unei operații de I/E
- Expirarea cuantei de timp alocate unui proces
- Existența unui proces cu prioritate mai mare

În ce constă planificarea execuției?

- Selecția unui proces/thread pentru rulare
  - algoritm de selecție
  - se parcurge coada READY
  - coada poate fi arbore, heap, listă, cozi multiple
- Schimbarea de context
  - se salvează contextul procesului curent
  - se încarcă procesul selectat

Criterii de planificare

- Criterii de selectare a procesului
- Utilizarea procesorului
  - gradul de “ocupare” a procesorului
- Productivitate (throughput)
  - câte procese sunt încheiate în unitatea de timp
- Turnaround time
  - timpul scurs de la crearea unui proces până la terminarea lui

Criterii de planificare (2)

- Timpul de așteptare
  - cât timp se așteaptă în coada READY
- Timpul de răspuns
  - timpul scurs de la crearea unui proces până la începerea execuției, sau
  - cât durează o tranziție WAITING->RUNNING
- Echitate/corectitudine (fairness)
  - procesele cu aceeași prioritate trebuie tratate similar

Criterii de planificare (3)

- Algoritmii de planificare urmăresc
  - maximizarea utilizării procesorului
  - maximizarea productivității
  - minimizarea turnaround time
  - minimizarea timpului de așteptare
  - minimizarea timpului de răspuns (interactivitate)

Notații

- TT – turnaround time
- TTM – turnaround time mediu
- J1, J2, ..., J# - jobul 1, 2, ..., # (batch)
- P1, P2, ..., P# - procesul 1, 2, ..., #

Planificare pentru sisteme batch

- Criterii importante
  - throughput
  - turnaround time
  - utilizarea procesorului
- Algoritmi de planificare
  - First-Come, First-Served
  - Shortest Job First
  - Shortest remaining time next
  - Planificarea pe trei niveluri

First-Come, First-Served

- Planificare în ordinea intrării în sistem
- Un proces care cere procesorul este trecut într-o coadă de așteptare
- Procesele care se blochează sunt trecute la sfârșitul cozii
- + ușor de înțeles și implementat
- - procesele CPU-bound încetinesc procesele I/O-bound (efectul de convoi)
- - timp mediu de așteptare destul de mare

Exemplu FCFS

- J1, J2, J3
- Joburile intră simultan în sistem
- Timpii de execuție: 24, 3, 3

- FCFS: ordinea J1, J2, J3
  - TT(J1) = 24; TT(J2) = 27; TT(J3) = 30
  - TTM = (24 + 27 + 30) / 3 = **27**

- Alt algoritm: ordinea J2, J3, J1
  - TT(J1) = 30; TT(J2) = 3; TT(J3) = 6
  - TTM = (30 + 3 + 6) / 3 = **13**

Shortest Job First

- Problemă?
  - trebuie cunoscută durata de execuție a unui job
- Planificarea în ordinea duratei de execuție

Exemplificare SJF

- J1, J2, J3, J4
- Job-urile intră simultan în sistem
- Timpii de execuție: 12, 20, 8, 4

- FCFS: J1, J2, J3, J4
  - TT(J1) = 12; TT(J2) = 32; TT(J3) = 40; TT(J4) = 44
  - TTM = (12 + 32 + 40 + 44) / 4 = 32

- SJF: J4, J3, J1, J2
  - TT(J4) = 4; TT(J3) = 12; TT(J1) = 24; TT(J2) = 44
  - TTM = (4 + 12 + 24 + 44) / 4 = 21

Shortest remaining time first

- Trebuie cunoscut timpul de execuție a jobului
- Versiune preemptivă a algoritmului SJF
  - când un nou job este submis pentru execuție
  - ... și timpul de execuție al acestuia este mai mic decât timpul rămas din execuția jobului curent
  - => jobul curent este suspendat și noul job este executat

Exemplificare SRTF

- J1, J2, J3, J4
- Timpii de intrăre în sistem: 0, 1, 2, 3
- Timpii de execuție: 8, 4, 9, 5

- SRTF: J1(0:1), J2(1:5), J4(5:10), J1(10:17), J3(17:26)
  - TT(J1) = 17; TT(J2) = 4; TT(J3) = 24; TT(J4) = 7
  - TTM = (17 + 4 + 24 + 7) / 4 = **13**
- SJF: J1(0:8), J2(8:12),J4(12:17),J3(17:26)
  - TT(J1) = 8; TT(J2) = 11; TT(J3) = 24; TT(J4) = 14
  - TTM = (8 + 11 + 24 + 14) / 4 = **14.25**

Planificarea pentru sisteme interactive

- Criterii importante
  - timpul de răspuns
  - satisfacția utilizatorului
- Algoritmi de planificare
  - Round-Robin
  - Clase de priorități
  - Shortest process next

Round-Robin

- FCFS preemptiv
- Cuantă de timp de rulare a programului
- La expirarea cuantei de timp procesul este preemptat
  - se apelează planificatorul
  - se alege alt proces (a cărui cuantă nu a expirat)

Round-Robin (2)

- Cum trebuie să fie cuanta de timp? (mare/mică)

- Cuantă de timp mare (secunde, sute de ms)
  - productivitate ridicată
  - timp de așteptare mare
  - interactivitate scăzută

- Cuantă de timp mică (4-10 ms)
  - interactivitatea ridicată
  - timp comparabil cu o schimbare de context
  - utilizare ineficientă a procesorului

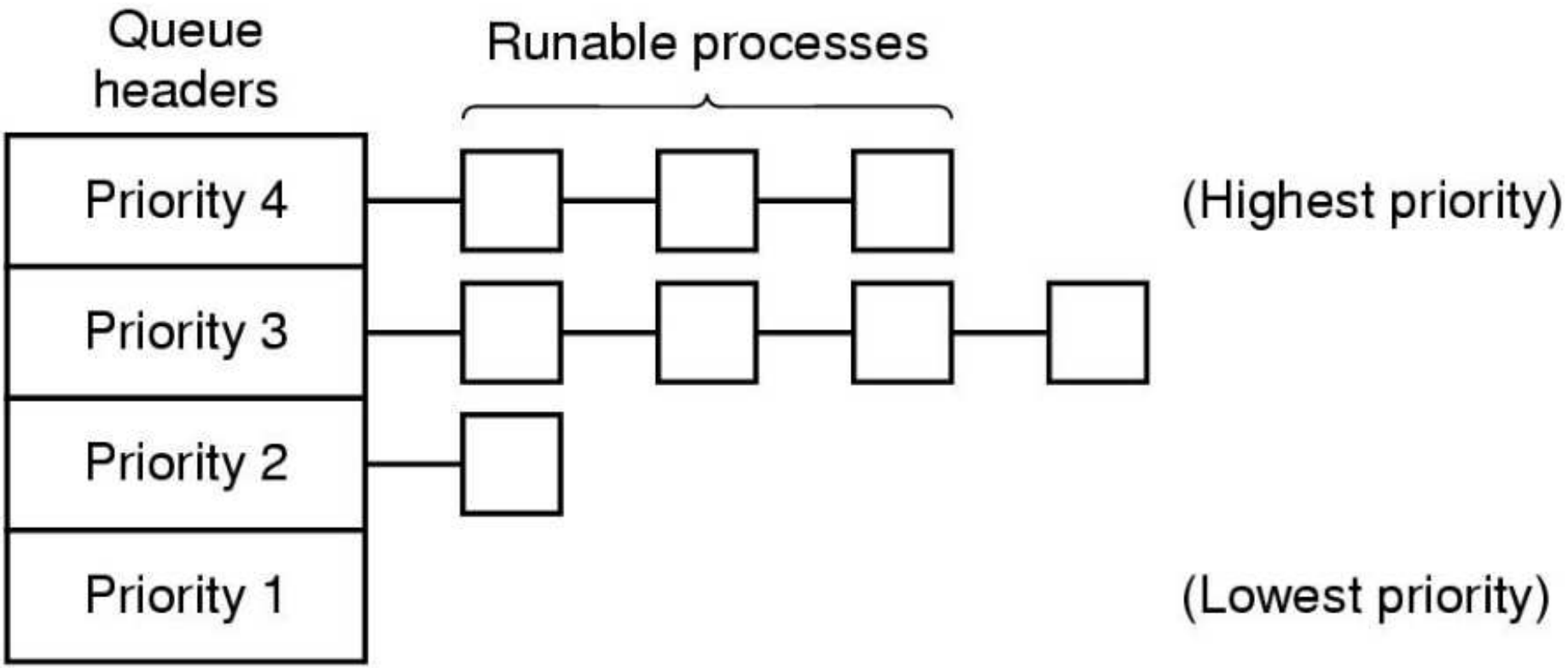
Planificare cu priorități

- Dezavantaj Round-Robin
  - toate procesele sunt „egale”
- Abordare planificare cu priorități
  - unele procese sunt „mai egale” decât altele
- Utilizatori importanți/mai puțin importanți
- Există procese mai importante/prioritare

Planificarea cu priorități (2)

- Prioritățile pot fi
  - statice
    - se cunoaște de la început care procese vor fi prioritare
  - dinamice
    - procesele I/O-bound sunt, în general, prioritare celor CPU-bound

Planificarea cu priorități (3)



Planificarea cu priorități (4)

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3540	root	15	0	470m	183m	29m	S	1	9.7	66:03.44	Xorg
27935	razvan	15	0	364m	49m	19m	S	1	2.6	6:36.10	amule
32666	haldaemo	15	0	29148	4132	3148	S	0	0.2	0:27.70	hald
1	root	18	0	10328	768	640	S	0	0.0	0:01.89	init
2	root	11	-5	0	0	0	S	0	0.0	0:00.00	kthreadd
3	root	RT	-5	0	0	0	S	0	0.0	0:00.02	migration/0
4	root	34	19	0	0	0	S	0	0.0	0:00.01	ksoftirqd/0
5	root	RT	-5	0	0	0	S	0	0.0	0:00.00	watchdog/0
9	root	10	-5	0	0	0	S	0	0.0	0:32.93	events/0
11	root	10	-5	0	0	0	S	0	0.0	0:00.00	khelper
31	root	10	-5	0	0	0	S	0	0.0	0:00.02	kblockd/0

Algoritmi cu multiple cozi

- Procesele sunt păstrate în mai multe cozi
- Fiecare coadă are asociată o prioritate
- Se rulează mai întâi procesele din coada cu prioritatea cea mai mare
- Un proces poate migra dintr-o coadă în altă coadă <--- algoritmi cu multiple cozi și feedback

Shortest process next

- Adaptare a SJF pentru sisteme interactive

- Problemă
  - nu se cunoaște timpul de execuție
- Soluție
  - estimare pe baza comportamentului anterior
  - se estimează o durată *T0*
  - procesul durează *T1*
  - estimarea pentru următoarea cuantă va fi *a * T1 + (1-a) * T0*
  - a – estimarea se uită sau nu repede
  - tehnică de estimare de tip *aging*

Planificare pentru sisteme real-time

- Criterii importante
  - îndeplinirea operațiilor în timp limitat
  - predictibilitatea
- Sisteme real-time
  - hard real-time
    - rezervarea resurselor
    - nu se folosește swapping sau memorie virtuală
  - soft real-time
    - procesele critice au prioritate maximă
    - pot cauza întârzieri mari celorlalte procese

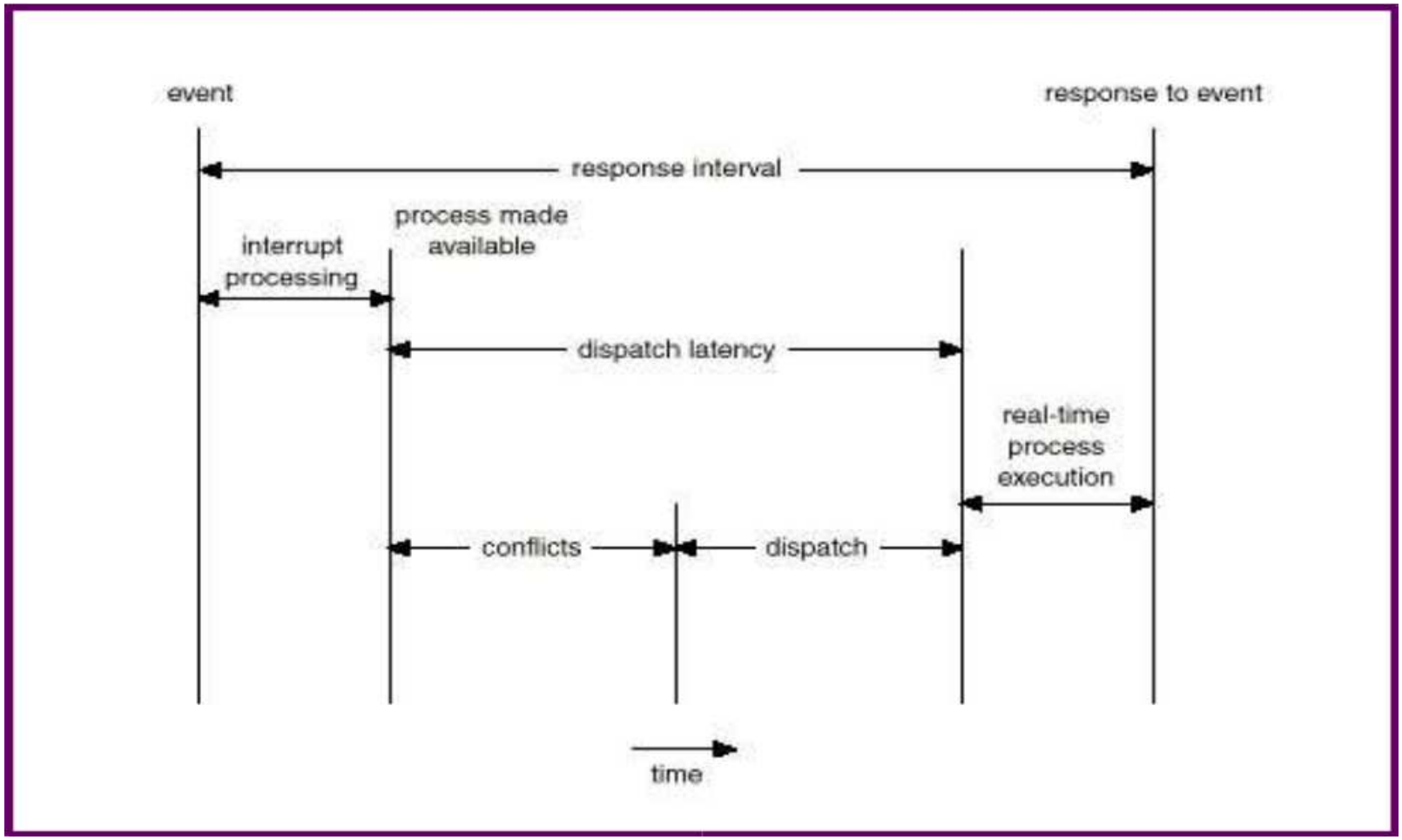
Planificare pentru sisteme real-time (2)

- Planificare real-time în sistemele time-sharing
  - trebuie să existe un planificator bazat pe priorități
  - procesele real-time trebuie să ruleze cu prioritate maximă
  - planificatorul nu trebuie să decrementeze prioritatea proceselor real-time când acestea rulează
  - planificatorul nu trebuie să întrerupă procesele real-time
  - schimbarea de context trebuie să dureze puțin

Latența unei schimbări de context

- Sumă de timpi
- Salvarea contextului curent
- Încărcarea noului context
- Așteptarea terminării unui apel de sistem
  - se pot introduce puncte de preemptare în apelurile de sistem cu durată mare
  - kernel preemptiv
- Probleme de *priority inversion*
  - priority inheritance

Latența unei schimbări de context (2)



Planificarea execuției pe Windows

- Algoritm de planificare preemptivă bazat pe priorități
- Subsistemul de planificare se cheamă dispatcher
- O schemă de priorități pe 32 de niveluri
  - 0 – prioritate sistem
  - 1-15 – clasă variabilă de priorități
  - 16-30 – clasă real-time
- Nucleu preemptiv

Apelarea dispatcher-ului

- Un thread se blochează și cedează controlul procesorului
- Este întrerupt un thread cu prioritate mai mare
  - un thread cu prioritate mai mare iese din starea BLOCKED
  - unui thread i-a fost schimbată prioritatea
- Unui thread îi expiră cuanta de timp

Algoritmul de planificare

- Este selectat procesul cu prioritatea cea mai mare
  - fiecare prioritate are asociată o coadă
  - procese din fiecare coadă sunt procese în starea READY
- Procesele din aceeași coadă sunt planificate asemănător cu Round-Robin

Priority boosting

- Unui proces îi este crescută pentru scurt timp prioritatea
- Situații
  - la încheierea unei operații de I/E
  - după așteptarea la un obiect de sincronizare
  - după ce un proces din foreground a așteptat o operație blocantă
  - thread-urile GUI sunt trezite
  - un thread nu a rulat în ultima perioadă

Priority boosting (2)

- Creșterea priorității este temporară
  - procesul rulează o singură cuantă de timp
  - se decrementează prioritatea și se rulează încă o coadă de timp
  - se continuă până când se ajunge la prioritatea de bază
- Prioritatea de bază este crescută, dar nu mai mult de 15
- La așteptarea I/E prioritatea este crescută în funcție de dispozitiv

Planificarea execuției în Linux

- Planificator preemptiv, time-sharing cu suport pentru procese real-time
- Kernel preemptiv de la versiunea 2.6
- Planificatorul gestionează 4 clase de procese
  - real-time FCFS
  - real-time RR
  - interactive
  - batch

Planificarea proceselor interactive

- 2.6 – 2.6.23 – O(1) scheduler
- Liste de procese pentru fiecare prioritate
- Două cozi de astfel de liste
  - coada activă: procesele care nu și-au consumat cuanta
  - coada expirată: procesele care nu și-au expirat cuanta
- Procesele sunt pedepsite (micșorarea priorității)
  - dacă se folosește mai mult timp de procesor decât este disponibil
  - se scade prioritatea dinamică proporțional cu cea statică
  - se ține cont de istoria procesului
- Operațiile de extragere/inserare în cozi se efectuează în O(1)

CFS

- Completely Fair Scheduler
- 2.6.23 – prezent
- Time-ordered red-black tree
- Virtual runtime
  - fiecare proces are un timp „virtual” de execuție
  - cel mai din stânga proces are timpul virtual cel mai mic – va fi următorul planificat
  - planificare: t_virtual_curent – t_virtual_stanga > threshold
- Operații în O(log n)

Planificarea proceselor real-time

- Conform cu POSIX.1b

- FCFS
  - procesele au doar prioritate statică, mai mare decât prioritatea proceselor time-sharing
  - este ales procesul cu prioritatea cea mai mare
  - un proces din această clasă va fi înlocuit doar dacă efectuează o operație blocantă

- RR
  - la fel ca FCFS, dar un proces va fi preemptat dacă își consumă cuanta de timp

Planificarea proceselor real-time (2)

- Procesele sunt planificate într-o manieră RR
- Cuanta de timp este mai mare decât pentru procesele interactive
- Procesele cu prioritate statică minimă sunt considerate procese batch

Cuvinte cheie

- planificare execuție
- context switch
- sheduler
- dispatcher
- CPU-bound
- IO-bound
- timp de așteptare
- turnaround time
- echitate (fairness)
- productivitate (throughput)
- interactivitate
- cuantă de timp
- time-sharing
- prioritate
- Round-Robin
- FCFS
- procese batch
- procese real-time

Exercițiu 1

- Un kernel preemptiv creşte sau scade timpul de răspuns? Dar throughput-ul?
- De ce este dificilă realizarea unei planificări care să fie atât echitabilă (fairness) cât și performantă (throughput)?

Exercițiu 2

- Fie un sistem ce foloseşte algoritmul de planificare shortest remaining time first. Calculați turnaround time-ul pentru următorul scenariu: J1(10, 0); J2(3, 3); J3(4, 3); J4(5, 6), unde un job este descris de J(timp de execuție, timpul la care intră în sistem).

Întrebări

?