Curs 04: Planificarea execuţiei

Sumar curs anterior

procesele sunt încapsularea execuţiei într-un sistem
procesele abstractizează procesorul, memoria şi I/O
un proces este creat de un alt proces, dintr-un executabil, prin loading
atributele unui proces sunt păstrate în PCB (Process Control Block)
pe Unix există apelurile fork() şi exec(): fork() duplică spaţiul de adrese al unui proces, exec() invocă loader-ul
procesele formează o ierarhie, procesul părinte aşteaptă încheierea proceselor copil
procesele orfane sunt adoptate de init
procesele zombie sunt cele care nu au fost încă aşteptate de procesul părinte
procesele comunică pentru transfer de date şi pentru notificări
pipe-urile anonime (folosite de shell la comanda cmd1 | cmd2) sunt folosite doar între procese înrudite

Multitasking şi scheduling

pe un sistem există mai multe procese
sistemul dispune de un număr limitat de procesoare
sistemul de operare trebuie să asigure accesul tuturor proceselor la procesoare
un proces rulează pe un procesor (sau mai multe) şi apoi lasă locul altui proces: multi-tasking
sistemul de operare planifică accesul unui proces la procesoare: scheduling
planificatorul de procese este o componentă din sistemul de operare care decide când un proces părăseşte procesorul şi ce proces îi ia locul

Schimbarea de context

un proces este un program căruia i se ataşează un context de execuţie
când planificatorul decide că un proces părăseşte procesorul, are loc o schimbare de context
schimbarea de context înseamnă salvarea informaţiilor procesului anterior într-o zonă din sistemul de operare şi restaurarea informaţiilor noului proces
schimbarea de context înseamnă overhead
schimbarea de context se poate produce:
* voluntar: un proces decide sau cauzează schimbarea de context
** un proces cedează de bună voie procesorul: yielding
** un proces execută o operaţie blocantă (de exemplu citire de la un dispozitiv care nu are încă date)
** un proces îşi încheie execuţia
* nevoluntar: planificatorul sistemului de operare decide forţarea unui proces de pe procesor
** un proces a stat prea mult timp pe procesor

** apare în sistem un proces mai important
+ demo cu schimbări voluntare şi nevoluntare la rularea comezii find

Reminder: Stările proceselor

un proces care rulează pe procesor este în starea running
un proces care execută o operaţie I/O se blochează în aşteptarea încheierii operaţiei, intră în starea blocking/waiting
atunci când un proces poate rula, dar nu are alocat un procesor, este în starea ready
stările running, blocking şi ready sunt cele trei stări principale ale unui proces
+ diagramă cu stările proceselor
tranziţia din running în ready se întâmplă când unui proces îi expiră cuanta
tranziţia din running în blocking este când un proces realizează o operaţie I/O blocantă
tranziţia din blocking în ready are loc când operaţia I/O blocantă s-a definitivat
tranziţia din ready în running este când se eliberează un procesor

Stările proceselor şi schimbările de context

tranziţiile din starea running sunt tranziţii de schimbare de context: procesul cedează procesorul în faţa altui proces
când un proces cedează procesorul, se alege un proces din starea READY
ce se întâmplă când nu există nici un proces în starea READY? procesul idle
ce tranziţii au loc la diferitele tipuri de schimbări de context:
RUNNING -> READY: yielding, prea mult timp pe procesor, proces prioritar
RUNNING -> WAITING: un proces execută o operaţie blocantă
RUNNING -> TERMINATED: un proces şi-a încheiat execuţia
noul proces execută tranziţia READY -> RUNNING

Planificatoare cooperative şi preemptive

planificatorul este o componentă a sistemului de operare; este o funcţie care este apelată pentru:
* a selecta un proces aflat în READY
* a înlocui procesul curent (din RUNNING) cu noul proces selectat (din READY)
planificatorul este apelat pentru a efectua schimbarea de context
planificatoarele pot fi cooperative şi preemptive
planificatoarele cooperative permit doar schimbări de context voluntare
cele preemptive permit şi schimbări de context nevoluntare: a preempta = a forţa părăsirea procesorului
planificarea cooperativă are dezavantajul starvation: un proces poate acapara complet procesorul şi să nu lase alte procese să ruleze
în general planificarea preemptivă presupune existenţa unei cuante de timp asociate unui proces; expirarea cuantei de timp forţează înlocuirea procesului (context switch)

Criterii de evaluare a unui planificator

sunt esenţiale două metrici: productivitate (throughput) şi echitate (fairness)
un planificator este cu atât mai productiv cu cât se consumă cât mai puţin timp schimbând contextul şi mai mult timp rulând procese; procesele să îşi termine cât mai repede treaba
un planificator este cu atât mai echitabil cu cât fiecare proces are acces la procesor; adică procesele stau cât mai puţin timp în coada READY până să intre în RUNNING
un sistem este inechitabil când un proces stă foarte mult în coada READY şi nu este planificat pe procesor: starvation
un sistem este neproductiv când sunt schimbări de context foarte dese şi procesoarele fac foarte puţină treabă
turnaround time: timpul din care un proces intră în sistem până când îşi încheie execuţia
average turnaround time: media turnaround time pentru toate procesele din sistem
waiting time: suma timpilor în care un proces aşteaptă în starea READY
average waiting time: media waiting time pentru toate procesele din sistem
un sistem productiv are average turnaround time mic
un sistem echitabil are average waiting time mic
Atenţie: waiting time se referă la timpul petrecut în starea READY (nu în starea WAITING)
+ demo cu timpul de aşteptare (în coada READY)

Cuanta de timp. Planificatorul round robin

cuanta de timp se asociază unui proces în planificatoare preemptive
când expiră cuanta unui proces acesta este scos de pe procesor, i se calculează o nouă cuantă şi este trecut în starea READY
timer-ul procesorului (uzual o dată 1 ms sau 10ms) generează întrerupere de ceas; în rutina de tratare a întreruperii de ceas se verifică dacă un proces în RUNNING are cuanta expirată
cuanta este dinamică, se actualizează la fiecare expirare
productivate vs echitate: cuantă mare vs cuantă mică
cuantă mare -> schimbări de context mai rare, productivitate sporită
cuantă mică -> schimbări de context mai dese, echitate sporită
cuanta variază între procese; procesele sunt CPU-intensive vs I/O-intensive
procesele I/O-intensive primesc în general o cuantă de timp mai mare pentru că se vor bloca şi vor trece din RUNNING în WAITING şi vor lăsa alt proces în loc
procesele CPU-intensive primesc în general o cuantă de timp mai mică; este posibil să ruleze pe procesor până la expirarea cuantei; cu o cuantă mare ar dura mai mult până ar lăsa alt proces în locul său pe procesor

Coadă/cozi de procese READY. Priorităţile proceselor

când are loc o schimbare de context, planificatorul alege un proces aflat în starea READY şi îl trece în RUNNING; ce procese alege

planificatorul round-robin are o coadă de procese READY; ia pe rând procesele READY şi le trece în RUNNIG; când un proces ajunge în READY e adăugat la sfârşitul cozii
unele procese sunt mai importante; au prioritate mai mare
planificatoarele cu priorităţi au mai multe cozi pentru procesele READY: câte o coadă pe o prioritate; când se planifică un proces se ia primul proces din coada cu prioritatea cea mai mare
prioritate statică este o prioritate care nu poate fi modificată pe parcursul execuţiei procesului; un sistem care foloseşte doar priorităţi statice poate duce la starvation pentru procesele cu prioritate mai mică; vor rula mereu procesele cu prioritate mai mare dacă dintre acestea sunt mereu în starea READY
priorităţile statice în Linux sunt numite "nice"
+ demo cu procese cu valoare nice diferită
pentru a preveni starvation, priorităţile sunt dinamice, se modifică pe parcursul execuţiei; un proces care stă destul de mult într-o coadă READY cu prioritate mai mică va fi promovat într-o coadă READY cu prioritate mai mare
modificarea priorităţii ţine cont de natura procesului, similar alegerii cuantei: procesele CPU-intensive primesc, în general, o prioritate mai mică decât procesele I/O intensive

Comunicarea inter-proces

procesele comunică pentru a transfera informaţii, pentru notificare sau pentru a asigura integritatea datelor (sincronizare)
transferul de informaţii se face prin
* comunicare de tip ţeavă, sau transfer de mesaje: pipe-uri (anonime), pipe-uri cu nume (FIFO), socketi locali (UNIX), socketi de reţea (Berkeley), cozi de mesaje
** există API-uri şi biblioteci de message passing: MPI, ZeroMQ, RabbitMQ
* comunicare cu date partajate: memorie partajată (shared memory)
** API de date partajate: OpenMP
la comunicarea prin transfer de mesaje există un sender, un receiver, un canal virtual de comunicare şi un protocol înţeles de parteneri; în general nu e nevoie de sincronizarea accesului la date pentru că fiecare partener are o instanţă
la comunicarea prin memorie partajată partenerii pot fi cititori sau scriitori; datele fiind comune, este nevoie de sincronizare pentru a asigura integritatea acestora
notificarea se face prin semnale (numite şi excepţii pe Windows); un proces îşi defineşte o rutină de tratare a semnalului/excepţiei, rutină apelată la apariţia semnalului
semnalele pot fi livrate de
* sistemul de operare ca urmare a întâmpinării unei situaţii neprevăzute în execuţia procesului: de exemplu acces nevalid la memorie (segmentation fault)
* de un alt proces pentru a notifica procesul curent de o condiţie sau eveniment
sincronizarea se face prin mutex-uri, semafoare, variabile condiţie; mai multe au fost prezentate la APD, vom insista la cursul 9: Sincronizare

Sumar

mai multe procese concurează pe procesoarele sistemului; planificatorul (parte a sistemului de operare) gestionează accesul proceselor la procesoare (trecerea în starea RUNNING)
planificarea unui proces, adică înlocuirea unui proces cu altul se numeşte schimbare de context
schimbarea de context se poate produce:
* voluntar: un proces decide sau cauzează schimbarea de context
** un proces cedează de bună voie procesorul: yielding
** un proces execută o operaţie blocantă (de exemplu citire de la un dispozitiv care nu are încă date)
** un proces îşi încheie execuţia
* nevoluntar: planificatorul sistemului de operare decide forţarea unui proces de pe procesor
** un proces a stat prea mult timp pe procesor
** apare în sistem un proces mai important
planificatoarele pot fi cooperative şi preemptive; cele cooperative nu pot preveni apariţia situaţiei de starvation
planificatorul urmăreşte două obiective conflictuale: productivitate şi echitate
planificatoarele preemptive folosesc cuantă de timp şi prioritate pentru fiecare proces
comunicarea intreprocese presupune transfer de mesaje, memorie partajată, notificare sau sincronizare