PB153 OPERČNÉ SYSTÉMY A ICH ROZHRANIE

ČO JE OPERAČNÝ SYSTÉM
- spracuváva systémové dáta a vstupy od používateľa a odpovedá alokovaním a spravovaním úloh
- vykonáva základné úlohy ako kontrola a alokovanie pamäte, pridelenie priority systémovým požiadavkám, kontrola vstupných a výstupných zariadení, umožnenie pripojenia do siete a správa súborov

MULTIPROGRAMOVANIE
- úloha má k dispozícii procesor, dokým nepožiada o I/O operáciu, ak ju potrebuje tak ju využije, ale stratí procesor (dostane ho iná úloha), úloha sa potom pozastaví a postaví sa na koniec fronty
- je to uživatel'sky nepríjemné, musí sa dlho čakať, ide iba o efektivitu využitia procesoru
- v pamäti je zároveň niekolko úloh súčasne
- pre výber úlohy, ktorá dotane procesor musíme mať procesorový plánovací algoritmus
- procesor je vyťažený, pokial' mám úlohy, ktoré nečakajú na dokončenie I/O operácii

MULTITASKING
- je to logické rozšírenie multiprogramovania, proces opustí procesor nie len keď potrebuje I/O operáciu ale aj keď mu vyprší pridelené časové kvantum
- procesor je multiplexovaný, v skutočnosti vždy beží len jedna úloha, medzi týmito úlohami sa však procesor prepína, takže uživatel' získava dojem, že úlohy sú spracovávané paralelne
- multitaskingový systém umožňuje rade uživatel'om počítačový systém zdieľať
- oproti multiprogramovaniu znižuje dobu odozvy (response time) interaktívnych procesov

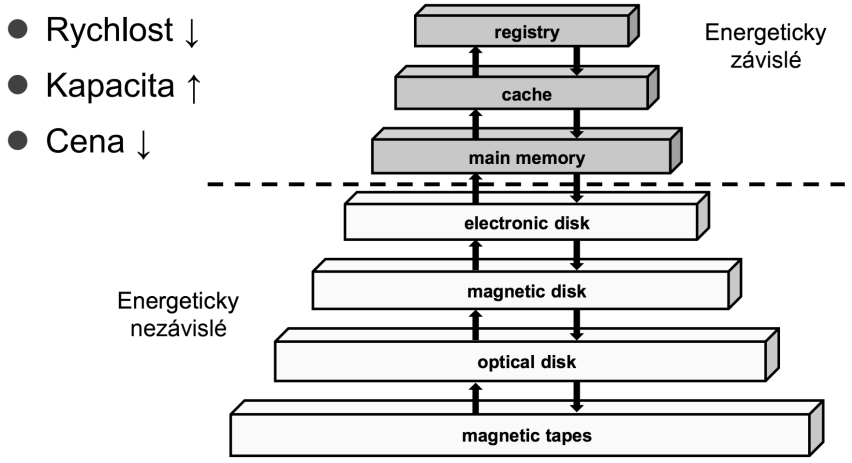
PRERUŠENIE
- prerušenie je signál od I/O zariadenia, že sa stalo niečo, čo by OS mal spracovať
- operačný systém je riadený prerušeniami
- prerušenie prichádza asynchronne
- aby nedochádzalo k “strate” prerušenia, je pri spracovávaní prerušenia ďalšie prerušenie zakázané (maskované)
- prerušenie nemusí byť generované len hardwarom, prerušenie je možné vyvolať aj softwérovými prostriedkami

I/O PRÍSTUPY
- sychronný prístup:
  - proces požiada o I/O operáciu, riadenie se uživatel'skému procesu vracia až po ukončení I/O operácie
  - výhody: jednoduchosť, vyhýbam sa súbežnému spracovaniu I/O požiadaviek
  - nevýhody: možná neefektivnosť
- asynchrony prístup:
  - proces požiada o I/O operáciu a riadenie se procesu vracia okamžite
  - výhody: je možné paralelne pracovat s niekol'kými I/O zariadeniami
  - nevýhody: komplexnejší systém, potrebujeme tabulku stavov I/O zariadení, príkazy pre počkanie na dokončenie I/O operácie apod.

DMA, DIRECT MEMORY ACCESS (priamy prístup do pamäte)
- spôsob ako rýchlo prenášať dáta medzi I/O zariadením a pamäťou
- výhoda je, že nezaťažujeme procesor
- procesor nemusí prenášať dáta po jednom bajte, požiada radič o prenos celého bloku dát

ŠTRUKTÚRA PAMÄTE
- primárna pamäť:
  - jediná väčšia pamäť, ktorú môže procesor priamo adresovať
  - typicky energeticky závislá
  - súčasná kapacita sú stovky MB až desiatky GB
  - súčasná rýchlosť v nanosekundách (je najrýchlejšia)
- sekundárna pamäť:
  - rozširuje pamäťovú kapacitu počítačového systému
  - energetický nezávislá
  - vysoká pamäťová kapacita (gigabajty až terabajty)
  - relatívne pomalá doba prístupu
  - v dnešnej dobe je najbežnejší magnetický disk

HIERARCHIA PAMÄTE
- rýchlosť a cena klesá
- kapacita stúpa

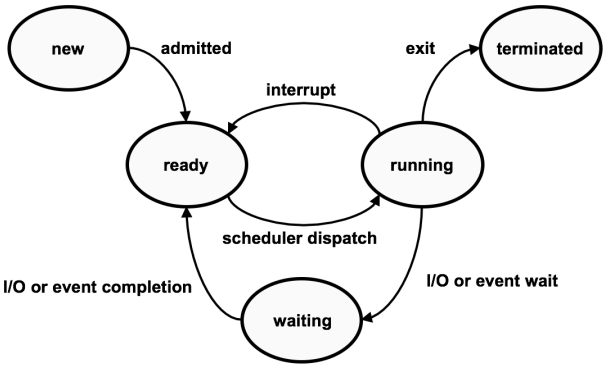


REŽIMY PROCESORU
- uživateľský a privilégovaný režim
- z privilegovaného režimu do uživatelského režimu sa procesor dostane špeciálnou inštrukciou, z uživatelského režimu do privilegovaného režimu sa procesor dostává pri spracovaní prerušenia
- niektoré inštrukcie je možné uskutočniť len v privilégovanom režime

POSIX
- Portable Operating System Interface
- je prenosné rozhranie pre operačné systémy
- vychádza zo systémov UNIX a určuje, ako majú POSIX-kompatibilné systémy vyzerať, čo majú vedieť, čo sa má ako robiť a podobne
- definuje rozhranie nie len pre programátorov, ale aj pre uživateľov

OS S MIKROJADROM
- Microkernel System Structure
- malé jadro OS plniace iba niekolko málo nezbytných funkcii (správa pamäte, podpora pre plánovanie procesov a komunikácia medzi procesmi)
- väčšina funkcii z jadra sa presúva do “uživateľskej” oblasti
- výhody:
  - ľahká prenositeľnosť OS, jadro je malé
  - vyššia spolahlivosť a bezpečnosť
  - flexibilita
- nevýhody:
  - zvýšená réžia (volanie služieb je nahradenévýmenou správ medzi procesmi)
- opakom je monolitické jadro, ktoré môže byť horšie udržovateľné, ale ľahšie sa navrhuje a implementuje

ČO JE TO PROCES
- spoločné pomenovanie pre spustený program (niekedy používame synonymum “task”)
- proces obsahuje:
  - čítač inštrukcii
  - zásobník
  - dátovú sekciu
  - program
- hierarchia procesov: rodič, potomok, súrodenci
- stavy procesu:
  - nový / new (práve vytvorený)
  - bežiaci / running (niektorý procesor práve vykonáva inštrukcie procesu)
  - čakajúci / waiting (čaká na určitú udalosť alebo I/O)
  - pripravený / ready (čaká na pridelenie času procesu)
  - ukončený / terminated (ukončil svoje prevedenie)



VYTVORENIE PROCESU
- rodič vytvára potomkov a potomkovia môže vytvárať ďalších potomkov
- vzniká strom procesov
- rodič a potomok zdieľajú zdroje pôvodné vlastnené rodičom

UKONČENIE PROCESU
- proces prevedie posledný príkaz a sám požiada OS o ukončenie (žiada jeho rodič)

PROCESS CONTROL BLOCK (PCB)
- tabuľka obsahujúca informácie potrebné pre definíciu a správu procesu
  - stav procesu (bežiaci, pripravený, …)
  - čítač inštrukcii
  - registry procesoru
  - informácie potrebné pre správu pamäte
  - informácie potrebné pre správu I/O
  - účtovacie informácie

PREPNUTIE KONTEXTU
- vyžiada sa služba, akceptuje se niekotré asynchronné prerušenie, obslúži sa a na novo se vyberie ako běžiaci proces
- keď OS prepája procesor z procesu X na proces Y, musí:
  - uchovať stav pôvodného procesu X (uložiť do PCB)
  - zaviesť stav nového procesu Y (z PCB)
- doba prepnutia závisí na konkrétnej hardwarovej platforme

DLHODOBÝ / STRATEGICKÝ PLÁNOVAČ
- vyberá, ktorý proces sa dá zaradiť medzi pripravené procesy
- plánovač je spúšťaný málo často
- nemusí byť super rýchli

KRÁTKODOBÝ PLÁNOVAČ
- prideluje procesor pripraveným procesom
- je spúšťaný často, preto musí byť rýchli
- dispečer: výstupný modul, ktorý predáva procesor procesu vybranému krátkodobým plánovačom
- plánovacie rozhodnutie môže vydať v okamihu, keď proces:
  - prechádza zo stavu bežiaceho do stavu čakajúceho
  - prechádza zo stavu bežiaceho do stavu pripravený
  - prechádza zo stavu čakajúci do stavu pripravený
  - končí

STREDNODOBÝ / TAKTICKÝ PLÁNOVAČ
- vyberá, ktorý proces sa dá zaradiť medzi odložené procesy
- vyberá, ktorý odložený proces sa dá zaradiť medzi pripravené procesy
- náleží čiastočne aj do správy operačnej pamäte

KRÁTKODOBÉ PLÁNOVANIE V LINUXE
- plánovací algorithms je súčasťou jadra
- 2 funkcie: schedule() – plánovanie procesov, do_timer() – aktualizuje informácie o procesoch

KRÁTKODOBÉ PLÁNOVANIE VO WIN32
- plánovací algoritmus je riadený hlavne prioritami
- 32 front (FIFO zoznamov) vlákien, ktoré sú “pripravené”
- jedna fronta pre každú úroveň priority
- fronty su spoločné pre všetky procesory

VLÁKNO
- objekt, ktorý vzniká v rámci procesu, je viditeľný iba vnútri procesu a je charakterizovaný svojím stavom (procesory sa pridelujú vláknam)
- každé vlákno si udržuje svôj vlastný zásobník, program counter (PC), registry, TCB (thread context block)
- vlákno môže pristupovať k pamäti a k ostatným zdrojom svojho procesu
- výhody: vlákno sa vytvorí aj ukončí rýchlejšie ako proces, jednoduchšie programovanie
- stay vlákien: beží, pripravený, čakajúci

VLÁKNA ULT
- User Level Threads
- správa vlákien sa vykonáva iba pomocou vláknovej knižnice
- jadro o nich nevie
- plánovanie a prepínanie vlákien si rieši sama aplikácia.

VLÁKNA KLT
- Kernel Level Threads
- správu vlákien podporuje jadro, udržuje aj informáciu o kontexte vlákien, aktivuje prepojenia medzi vlaknami
- pri prepnutí vlákna sa 2x prepína režim procesoru

ALGORITMUS FCFS
- First Come, First Served ("Kto skôr príde, ten skôr melie")
- procesy sú obslúžené v poradí, v akom prišli
- efektivity závisí na dĺžke a poradí procesov (ak sú na začiatku fronty krátke procesy je to efektívne, ak sú dlhé tak to nie je efektívne)

ALGORITMUS SJF
- Shortest Job First
- vyberá proces s najkratším požiadavkom na procesor
- je to druh prioritného plánovanie, rolu priority tvorí predpokladaná dĺžka nasledujúcej dávky procesora
- preemtívna varianta:
  - pokiaľ v priebehu jeho vykonávania príde ďalší proces kratší než je zbytkový čas prvého procesu, je prvý proces prerušený a obslúžený predchádzajúci proces
- nepreemtívna varianta:
  - nedovoluje sa predbiehanie

URČENIE DĹŽKY NASLEDUJÚCEJ DÁVKY PROCESORU
- dĺžku ďalšej dávky procesoru nepoznáme, môžeme ju iba odhadovať
- použijeme exponenciálne spriemerovanie

1. t_n = skutecna delka n - te davky
2. τ_{n+1} = predpokladana hodnota pro dalsi CPU davku
3. $\alpha, 0 \le \alpha \le 1$, koeficient historie
4. pak definujeme odhad jako :

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n .$$

PRIORITNÉ PLÁNOVANIE
- s každým procesom je spojené prioritné číslo (najvyššia priorita - najnižšie číslo)
- procesor sa prideluje procesu s najvyššou prioritou
- preemptivná(s predbiehaním) aj nepreemtivná(bez predbiehania) varianta
- aby sa zamedzilo stárnutiu, tak práve prevedenému procesu klesá priorita a čakajúcim procesom sa priorita zvýši

ROUND ROBIN (RR)
- každý proces dostáva na procesore mal jednotku času - kvantum (desiatky až stovky milisekund)
- efektivita záleží na dĺžke kvanta(príliš krátke spomaluje beh)
- s viacúrovňovými frontami (Win32) máme prioritné fronty (skupiny), najskôr sa vykoná skupina s najväčšími prioritami, vnútri sa plánuje pomocou RR

STRÁNKOVANIE
- LAP(logický adresový priestor) procesu nemusí byť jedinou súvislou sekciou FAP(fyzický adresový priestor), LAP sa zobrazuje do sekcii FAP
- FAP sa delí na sekcie zvané rámce, LAP sa delí na sekcie zvané stránky
- udržujeme zoznam voľných rámcov
- program dĺžky n stránok sa umiestni do n rámcov
- preklad logic adresa -> fyzická adresa, pomocou prekladovej tabulky nastavovanej OS a interpretovanej MMU (memory managment unit)

TABULKA STRÁNOK
- je uložená v operačnej pamäti
- jej počiatok a koniec je odkazovaný registrom
- sprístupnenie údajov / inštrukcii v operačnej pamäti vyžaduje 2 prístupy do operačnej pamäti (jeden do tabulky strýnok a jeden do údajov/inštrukcii)

SEGMENTÁCIA
- je to metóda správy pamäte, kedy sa procesu vytvorí virtuálny adresný priestor začínajúci od nuly
- pre získanie fyzickej adresy používa 2 špeciálne registry: segment a offset
- logic adresa je dvojica segment a offset
- nepoužívané segmenty môžu byť odsunuté na pevný disk

FRAGMENTÁCIA
- vonkajšia fragmentácia:
  - nie je možné alokovať súvislý blok pamäte, aj keď v súčte je dostupná pamäť dostatočná
- vnútorná fragmentácia:
  - pridelená oblasť je v násobkoch velikosti rámca (často 4Kb), v prípade, že velikosť procesu nie je násobkom, zostane časť pamäte nevyužitá

VIRTUÁLNA PAMÄŤ
- virtualizácia pamäte je spôsob správy operačnej pamäte, ktorý umožňuje procesu predĺžiť adresný priestor, ktorý je usporiadaný inak alebo dokonca väčší ako pamäť RAM
- separácia LAP a FAP
- dajú sa efektívnejšie vytvárať procesy
- vo FAP sa môžu nachádzať iba časti programov nutné pre bezprostedné riadenie procesu
- vo FAP sa dá udržovať viac procesov naraz
- stránkovanie / segmentáciu musí podporovať hardware správy pamäte
- OS must byť schopný organizovať tok stránok / segmentov medzi vnútornov a vonkajšou pamäťou

ZAVÁDZANIE STRÁNKY
- KEDY ?
  - stránkovanie na žiadosť: stránka sa zobrazuje do FAP pri odkaze na ňu
  - predstránkovanie: zavádza sa viacej stránok než sa žiada; princíp lokality
- KDE ?
  - segmentácia: best fit (pridelujú sa najmenšie dostatočne dlhé volné miesta), first fit (pridelujú sa prvé dostatočne dlhé volné miesta), worst fit (pridelujú sa najväčšie dostatočne dlhé volné miesta)
  - stránkovanie: nemusíme riešiť
  - segmentácia + stránkovanie: nemusíme riešiť

VÝPADOK STRÁNKY
- ak sa stránka nenachádza vo FAP, musíme je tam zaviesť
- je aktivovaný OS pomocou prerušenia “page fault” (výpadok stránky)
- OS zistí:
  - nelegálnu referenciu: informuje o tom proces
  - legálnu referenciu, ale stránka chýba v pamäti: zavedie stránku do pamäti
- potom sa opakuje inštrukcia, ktorá spôsobila “page fault”

ALGORITMY PLÁNOVANIA DISKU
- OS je zodpovedný za efektívne používanie hardwaru - čítanie z pevného disku
- najznámejšie algoritmy: FCFS, SSTF, SCAN, C-SCAN, C-LOOK

ALGORITMY URČENIA OBETE
- proces vyžaduje stránku, ktorú sme najskôr odložili na pevný disk, pamäť je ale plná, musíme teda vybraž stránku v RAM, ktorá bude obeťou a presunie sa na disk
- čím viac rámcov máme, tým menej často dôjde k výpadku stránky

ALGORITMUS FIRST -IN FIRST -OUT (FIFO)
- najjednoduchší stránkovací algoritmus
- vyhodí prvú stránku vo fronte
- neefektívny

OPTIMÁLNY ALGORITMUS
- obeťou je najneskoršia zo všetkých nasledujúcich odkazov na stránky
- potrebuje vedieť budúcu postupnosť referencii
- algoritmus je iba teoretický

ALGORITMUS LEAST RECENTLY USED (LRU)
- obeťou je najdlhšie neodkazovaná stránka
- využíva princíp lokality (pravdepodobnosť jej skorého využitia je malá)
- výkov je blízky optimálne stratégii, ale algoritmus je nákladný v kontexte systémových prostriedkov
- musíme stále posúvať zoznam stránok, podl'a toho ako sa používajú

DRUHÁ ŠANCA
- modifikácia FIFO: výber obete je cyklickým prechádzaním
- obeť, ktorá nemá nastavný príznak a je na rade pri kruhovom prechádzaní je nahradzovaná
- odkazom na stránku sa nastaví príznak
- z výberu obetí sa vynecháva aspoň raz odkázaná stránka od posledného výberu

ČO JE SCAN
- hlavička disk začína na jednej strane disku a presúva sa pri splňovaní požiadaviek ku druhej strane disku, potom sa vracia späť a opäť plní požiadavky
- niekedy sa nazýva ako algoritmus typu výťah (elevator)
- vhodnejšie pre väčšiu záťaž disku

ČO JE C-SCAN
- spôsob plánovanie disku, resp. plánovanie čítania z disku
- snažime sa minimalizovať dobu prístupu
- od počiatočného miesta postupuje na koniec disku, behom tohoto číta, potom sa bez čítania vracia na začiatok a znovu postupuje a číta smerom na koniec
- vhodnejšie pre ťažkú táťaž disku

TECHNOLÓGIA RAID
- redundant arrays of independent disks
- spojenie viacej diskov, ktoré navonok vyzerajú ako jeden, za účelom navýšenia kapacity a rýchlosti alebo spolahlivosti
- namiesto na jeden disk sa ukladajú dáta redundantne na dva, pri výpadku ho druhý zastúpi
- úrovne RAID:
    - level 0: žiadna redundiancia, len väčšia kapacita
    - level 1: zrkadlenie diskov
    - level 2: delenie bitov, kontrolné disky vykonávajú kontrolu pomocou hamminga
    - level 3: delenie bitov, jeden kontrolný disk na skupinu
    - level 4: delenie blokov, nezávislé operácie read/write
    - level 5: viacej súbežných prístupov, dáta/parity cez vetky disky
    - level 6: odolnosť pri viacej než jednej poruche disku

RACE CONDITION
- viacej procesorov súčasne pristupuje ku zdielaným zdrojom a manipulujú s nimi

SEMAFÓRY
- synchronizačný nástroj, ktorý ide implementovať aj bez “busy waiting”
- proces je operačným systémom “uspaný” a “prebudený”
- semafór je buď binárny alebo nadobúda celočíselné hodnoty z neobmedzeného intervalu

SPRÁVNE RIEŠENIE KRITICKEJ SEKCIE
- u kritickej sekcie musíme zaručiť vzájomné vylúčenie a dostupnosť kritickej sekcie, teda trvalosť postupu a podmienka spravodlivosti
- v kritickej sekcii nie sú nikdy 2 procesy súčasne, vždycky sa vyberie nejaký proces a každý proces sa niekedy dostane na radu

PETERSONOV ALGORITMUS
- rieši vzájomné vylúčenie a dostupnosť kritickej sekcie
- proces A nie je na rade a dá prednosť procesu B
- pokiaľ B nechce vstúpiť do kritickej sekcie alebo dal prednosť, vsúpi A
- pokiaľ B chce vstúpiť, A loopuje a aktívne čaká, až B odíde
- nastavujú sa 3 bity, flag A, flag B a turn; jedná sa o atomické operácie

PROBLÉM UVIAZNUTIA
- existuje množina blokovaných procesov, každý proces vlastní nejaký prostriedok (zdroj) a čaká na zdroj držaný iným procesom z tejto množiny
- k uviaznutiu dôjde(deadlock), keď súčasne začnú platiť 4 podmienky:
    - vzájomné vylúčenie
    - ponechanie si zdroja a čakanie na ďalší
    - bez predbiehania
    - kruhové čakanie
- 2 metódy riešenia:
    - prevencia uviaznutia: zneplatnenie jednej z podmienok
    - obchádzané uvaznutie: než je pridelený zdroj, kontroluje sa, či nemôže vzniknúť deadlock(ak vznikne, treba reštart do predhádzajúceho bezpečného stavu)