1. Multiprogramovanie

2. Rozdiel medzi multiprogramovanim a multitaskingom

Multiprogramovanie – v pamati je viac procesov. Zakladna myslieka – uloha vyuziva procesor, dokym nepotrebuje I/O zariadenie, ked ho potrebuje, ide ho vyuzit, ale procesor strati (zoberie si ho ina uloha) a musi sa postavit na koniec fronty. Uzivatelsky neprijemne, dlho sa caka. Ide iba o efektivitu vyuzitia procesoru.

Multitasking – viac uzivatelov, viac procesov. Narozdiel od multiprogramovania, procesom sa pridelia casove okna – uzivatel ma dojem ze bezi vsetko naraz. Bezi to pomalsie, vyssia rezia, ale bezi vsetko. Je tam planovaci algoritmus, ktory rozhoduje, co pojde kedy na procesor. Umoznuje aby bezalo viac procesov a aby mal pocitac viac uzivatelov. Znizuje sa doba odozvy. Moznost synchronizacie procesov.

3. Co je DMA?

Direct Memory Access – Sposob ako rychlo prenasat data medzi I/O zariadenim a pamatou.

Prenos dat bez potreby vyuzivat CPU. CPU nemusi prenasat data, poziada radic o prenos celeho bloku dat. CPU poziada o prenos dat, po skonceni prenosu sa o tom dozvie pomocou prerusenia. Takze 1 prerusenie za cely blok dat namiesto 1 prerusenia po kazdom bajte.

V dobe prenosu superi I/O radic a CPU o pristup do pamati – oba pracuju s rovnakou pamatou.

4. OS s mikrojadrom

Male jadro OS, ktore plni iba niekolko malo potrebnych funkcii. Vacsina funkcii jadra sa presuva do uzivatelkej oblasti.

5. Stavy procesov

Proces sa moze nachadzat v jednom zo stavov:

Novy (new) – prave vytvoreny proces (ihned prechadza do "ready")

Beziaci (running) – procesor vykonava jeho instrukcie

Cakajuci (waiting) – caka na urcitu udalost (napr. I/O zariadenie)

Pripraveny (ready) – caka na pridelenie casu procesoru

Ukonceny (terminated) – ukoncil svoje provadeni

6. Prepnutie kontextu

OS prepojuje CPU z procesu X na proces Y. Musi uchovat stav procesu X (ulozit do PCB – Process Control Block), prepnut procesy a nacitat novy proces (zaviest stav do noveho proicesu nacitanim z PCB provcesu Y)

Prepnutie kontextu predstavuje rezijnu stratu. Behom prepinania kontextu system nic efektivne nerobi. Doba prepnutia zavisi na konkretnej HW platforme.

7. Kratkodoby planovac

Vybera sa proces, ktoremu bude pridelene CPU. Vybera jeden z procesov v operacnej pamati v stave “pripraveny”.

Planovacie rozhodnutie moze vydat v okamihu ked proces prechadza:

1 — zo stavu beziaci do cakajuci (nepreemptivne planovanie)

2 — beziaci do pripraveny (preemptivne planovanie)

3 — cakajuci do pripraveny (preemptivne planovanie)

4 — proces konci (nepreemptivne planovanie)

Dispecer – vystupny modul kratkodobeho planovaca, ktory predava procesor procesu vybranemu kratkodobym planovacom. Predanie zahrnuje – prepnutie kontextu, prepnutie rezimu procesoru na uzivatelky, skok na zodpovedajuce miesto v uzivatelkom programe pre opatovne pokracovanie v behu procesu.

8. Strednedoby planovac

Kazdy proces musime zaviest do RAM, ale ked ich tam je prilis vela, znizuje to vykonnost aj pri vyuziti virtualnej pamati. OS musi vykonavanie niektorych procesov odlozit – vratit na disk.

Strednedoby (takticky planovac) – vybera, ktory proces mozeme zaradit medzi odlozene procesy. Vybera, ktore odlozene procesy mozeme zaradit medzi pripravene procesy.

2 nove stavy – odlozeny cakajuci, odlozeny pripraveny

9. Vlakna ULT

ULT – User Level Threads

Sprava vlaken sa provadi iba pomocou vlaknovej knihovny (Thread library – obsahuje funkcie pre vytvaranie, rusenie vlakien, predavanie sprav, planovanie behu, uchovavanie kontextu vlakien…) na urovni uzivatelskeho programu.

Jadro o nich nevie. Prepinanie medzi vlaknami nepotrebuje provadenie funkcii jadra, neprepina sa ani kontext procesu, ani rezim procesoru (vyssia rychlost). Planovanie a prepinanie vlakien si riesi sama aplikacia.

Ak nejake vlakno zavola sluzbu jadra, je zablokovany cely proces dokym sa sluzba nesplni.

10. Vlakna KLT

KLT – Kernel Level Threads

Spravu vlakien podporuje jadro, udrzuje aj info o kontexte vlakien, aktivuje prepojenia medzi vlaknami. Moze sucasne planovat beh viacerych vlakien jedneho procesu. Blokovanie dochadza na urovni vlakna, nie cely proces.

Prepojenie medzi vlaknami riesi jadro – pomalsie. Pri prepnuti vlakna sa 2 krat prepina rezim procesoru – rezia naviac.

11. Co je FCFS

12. Algoritmus RR

13. Algoritmus SJF

14. SJF vs. RR

15. RR s viacurovnovymi frontami

FCFS – First Come, First Served – kto prvy pride, ten prvy mele – klasicka fronta, efektivita zalezi na tom, ci pridu do fronty ako prve kreatke, rychle procesy, alebo pomale, dlhe s vysokymi narokmi na CPU (Gantt)

SJF – Shortest Job First – musime poznat dlzku poziadavku na davku CPU pre kazdy proces. Vyberame proces s najkratsim poziadavkom na CPU. Dve varianty – NEPREEMPTIVNE (bez predbiehania – proces na CPU uz nemoze byt nahradeny inym, kratsim), PREEMPTIVNE (s predbiehanim – ak je na CPU proces a pride dalsi, ktory ma kratsiu poziadavku ako ta, ktora este ostava beziacemu procesu, moze ho nahradit). Dochadza k starnutiu – niektore velke procesy mozu byt neustale predbiehane a nedostanu sa na CPU nikdy.

RR – Round Robin – Kazdy proces dostava CPU na malu jednotku casu. Po uplynuti doby je vymeneny za najstarsi proces vo fronte propravenych procesov a zaradi sa na koniec. Jednotka casu = casove kvantum q. Ak je q pre 1 proces moc velke je to ekvivalentne FIFO, ak moc male, vznika velka rezia – pomale.

16. Kratkodobe planovanie v Linuxe

Planovaci algoritmus je sucastou jadra OS. 2 funkcie – schedule() – planovanie procesov, do_timer() – aktualizuje informacie o procesoch. Casove kvantum 1/100 sekundy. Planovaci algoritmus bol predmetom vyvoja.

4 kategorie procesov z hladiska planovania – SHRED_FIFO (metoda FIFO), SHRED_RR (metoda RR), SHRED_OTHER (procesy planovane na zaklade dynamickych priorit, priorita je 0 a pri starnuti je zvysovana), SHRED_BATCH

17. Kratkodobe planovanie vo WIN32

Planovaci algoritmu je riadeny hlavne prioritami. 32 front (FIFO zoznamov) vlakien, ktore su “pripravene”. Jedna fronta pre kazdu uroven priority. Fronty su spolocne pre vsetky procesory.

Ked je vlakno pripravene, bud bezi okamzite, alebo je umiestnene na koniec fronty “pripravenych procesov” vo svojej priorite.

V jednoprocesorovom stroji vzdy bezi vlakno s najvyssou prioritou.

V ramci jednej prioritnej skupiny sa planuje algoritmom RR pomocou casovych kvant.

18. Rozdiel medzi vnutornou a vonkajsou fragmentaciou

Vonkajsia – suhrn volnej pamati je dostatocny, ale nie v dostatocne suvislej oblasti (vela oblasti po malo volnej pamati dokopy dava dostatocne vela volnej pamati).

Vnutorna – Pridelena oblast pamati je vacsia ako pozadovana velkost. Cast pridelenej pamati sa nevyuziva.

Vonkajsiu fragmentaciu mozeme znizovat “sestřásáním”. Posuvaju sa obsahy pamati s cielom vytvorit jeden vacsi volny blok. Len ak je mozna dynamicka relokacia.

19. Strankovanie pamati

LAP (Logicky Adresovy Priestor) nemusi byt jedinou suvislou sekciou FAP (Fyzicky AP). LAP sa zobrazuje do sekcii FAP.

FAP sa deli na sekcie zvane ramce. Pevna dlzka ramcov (dlzka v nasobkoch mocnin 2).

LAP sa deli na sekcie zvane stranky. Pevna dlzka, zhodna s dlzkou ramcov.

Udrzujeme zoznam volnych ramcov. Preklad logickej adresy na fyzicku pomocou MMU – Memory Management Unit.

Vznika vnutorna fragmentacia – stranky maju pevnu velkost (nasobky velkosti ramca), ktoru nemusia zaplnit celu. Vonkajsia fragmentacia nie.

20. Vyznam tabulky stranok a princip hashovacej tabulky

Tabulka stranok je ulozena v operacnej pamati. Jej pociatok a koniec je odkazovany registrom. Spristupnenie udajov/instrukcii vyzaduje 2 pristupy do operacnej pamati (raz do tabulky stranok, raz pre udaj/instrukciu).

Problem zhorsenia efektivity dvojitym pristupom mozeme riesit specialnou rychlou HW cache pamatou.

21. Segmentacia

Logicka adresa je dvojica (segmet s, ofset d). Tabulka segmantov obsahuje base (pociatocna adresa umiestnenia segmentu), limit – dlzka segmantu.

Oproti strankovaniu, segmenty nie su rovnako velke, ale take velke ako potrebujeme. Dochadza k vonkajsej fragmentacii.

22. Zavadzanie stranky

KEDY?

Strankovanie na ziadost – stranka sa zobrazuje do FAP pri odkaze na nu, ak uz vo FAP nie je vytvorena. Pociatocne zhluky vypadku stranok. Ziadost = dynamicky a kontextovo generovana indikacia nedostatku.

Predstrankovanie – zavadza sa viac ako sa ziada, princip lokality. Umiestnenie na vonkajsej pamati susednych stranok byva blizke v LAP. Vhodne pri inicializacii procesu.

KDE?

Segmentacia – first-fit, best-fit, worst-fit

Strankovanie – nemusime riesit

Kombinacia Segmentacie a Strankovania – nemusime riesit

23. Ako sa postupuje pri vypade stranky

Ak sa stranka ktoru volame nenachadza vo FAP, musime ju tam zaviest.

Je aktivovany OS pomocou prerusenia “page fault”.

OS zisti bud NELEGALNU REFERENCIU – informuje o tom proces, alebo LEGALNU R. – zavedie stranku do pamati. Ziskame prazdny ramec a zavedieme do neho stranku (uprava Valid/Invalid bitu v tabulke na 1).

Potom sa opakuje instrukcia, ktora sposobila “page fault”.

24. Algoritmus FIFO

Algoritmus urcenia obeti. First In First Out. Obet = stranka, ktora je najdlhsie zobrazena vo FAP. Casto pouzivane stranky su v mnohych pripadoch prave tie najstarsie. Pravdepodobnost vypadku takej stranky je vysoka, takze algoritmus zbytocne odklada casto pouzivane stranky. Jednoducha implementacia.

25. Algoritmus LRU

Algoritmus urcenia obeti. Last Recently Used. Obet = Najdlhsie neodkazovana stranka. Princip lokality hovori, ze prst jej skoreho pouzitia je velmi mala. Vykon je blizky optimalnej strategii.

Implementacia – moznost 1 – V polozkach tabulky stranok sa udrzuje HW citac – sekvencne cislo, ma konecnu kapacitu, problem jeho precitania, zvysuje reziu.

Implementacia – moznost 2 – Zasobnik cisel stranok, pri pristupe ku stranke sa odstrani zo zasobnika a vlozi na vrchol. Na spodku je obet.

Aproximacia – ci bola stranka poucita do nejakeho casu v minulosti. Iba jeden bit 1/0 bola/nebola.

26. Spravne riesenie kritickej sekcie + Petersonov algoritmus

Problem kritickej sekcie – viac procesov superi o pravo pouzivat iste zdielane data. V kazdom procese sa nachadza segment kodu nazyvany kriticka sekcia, v ktorom proces pristupuje k zdielanym zdrojom. Je potrebne zaistit, aby sa v kritickej sekcii, zdruzenej s istym zdrojom, nachadzal najviac 1 proces.

Podmienky riesienia problemu:

1 — Podmienka vzajomneho vylucenia / bezpecnosti (Ak 1 proces provadi svoju kriticku sekciu, nemoze uz ziadny iny proces provadit svoju KS zdruzenu s danym zdrojom)

2 — Podmienka trvalosti postupu / zivotnosti (Ak ziadny proces na danom zdroji neprovadi svoju KS, potom vyber procesu cakjuceho na zdroj sa nemoze odkladat nekonecne dlho)

3 — Podmienka konecnosti doby cakania / spravodlivosti (Musi byt nejaka horna mez, ktora udava kolko procesov moze iny proces predbehnut. Kazdy sa musi dostat v konecnom case k zdroju)

Riesenie problemu: SW riesenia (algo, ktorych spravnost neposlieha na ziadne dalsie predpoklady, s aktivnym cakanim “busy waiting"), HW riesenia (vyzaduju specialne instrukcie procesoru, s “busy waiting"), riesenia sprostredkovane OS (potrebne funkcie a dat. struktury poskytuje OS, s pasivnym cakanim)

27. Semafor

Synchronizacny nastroj na riesenie kritickej sekcie, ktory mozeme implementovat aj bez “busy waiting”. Riesenie na urovni OS. Proces je operacnym systemom uspany a potom znovu

prebudeny. Hodnota, ktora drzi informaciu o tom, v akom stave je dany zdielany prostriedok. Ak je hodnota > 0, nemozeme pouzit, ak 0, mozeme

28. Co je POSIX

29. Kedy dojde k deadlocku

30. Podmienky uviaznutia + 2 preventivne metody

K uviaznutiu dojde ked sucasne platia 4 podmienky:

1 — vzajomne vylucenie (zdielany zdroj moze v 1 okamihu pouzivat len 1 proces)

2 — ponechanie si zdroja a cakanie na dalsi (proces vlastniaci zdroj caka na zdroji na iny zdroj obsadeny inym procesom)

3 — bez predbiehania (zdroj moze byt uvolneny iba dobrovolne procesom, ktory ho vlastni, ked ho uz nepotrebuje)

4 — kruhove cakanie (existuje taky zoznam cakajucich procesov {P1, P2, P3 … Pn}, ze P1 caka na zdroj drzany P2, P2 na P3 … a Pn na P1)

31. Hierarchie pamati

Primarna pamat – operacna, najrychlejsia, energeticky zavisla

Sekundarna pamat – pomalsia, vacsia, energeticky nezavisla, flash-disky, magneticke disky

Tercialna pamat – najpomalsia, lacna, externe vymenitene media, energeticky nezavisle, tzv. offline storage, floppy/magneticke/opticke disky

32. Algoritmy planovania pristupu na disk

FCFS, SSTF (Shortest Seek Time First), SCAN, C-SCAN, C-LOOK

33. RAID level 1, RAID level 0