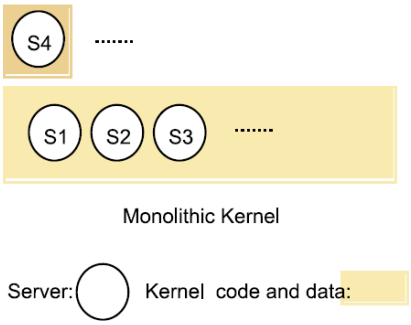
1. *Vyjmenuj genericke komponenty a seřaď je*

a) Správa procesorů
b) Správa procesů (proces – činnost řízená programem)
c) Správa paměti (hlavní, vnitřní)
d) Správa souborů
e) Správa V/V zařízení
f) Správa vnější paměti (sekundární)
g) Networking, distribuované systémy
h) Systém ochran
i) Interpret příznaků

2. *Nakresli architekturu monolitickeho jadra a vyjmenuj jeho nedostatky*

- mohutný objekt, obtížně diferencovatelný, obtížně manipulovatelný
- Veškerý kód běží ve stejném paměťovém prostoru, tím se liší od mikrojádra.
- Pokud se v jednom podsystému naskytne chyba, zablokuje to všechny ostatní z důvodu stejné paměti.



3. *Popis instrukci procesoru*

- Zavedení hodnoty z paměti do registru
- Operace nad hodnotami v registrech / paměti
- Větvení (skoky)
- Řízení (start V/V,…)

4. *Segmentovani a strankovani - vypsat rozdily a v cem se shoduji*

Používá se při dynamických výměnách částí mezi virtuální a reálnou pamětí. Stránkování a segmentaci máme prostou a na žádost (nepřítomnost adresovaného objektu v reálné paměti). Musí je podporovat HW správy paměti. OS musí být schopen organizovat tok stránek / segmentů mezi vnitřní a vnější pamětí.
Segmentování – stránky se uspořádávají do logických celků (OS vytvoří celek, o kterém si myslí, že pracuje společně), reálná paměť není implicitně dělena, použité segmenty virtuální paměti určuje programátor / kompilátor, žádná vnitřní fragmentace (rozdělení na díly), možnost externí fragmentace. OS udržuje tabulku segmentů s bází a délkou každého segmentu a udržuje seznam volných oblastí reálné paměti. Převod z virtuální paměti se řeší dynamicky při běhu mikroprogramem podle tabulky segmentů. Zavádějí se do reálné paměti podle potřeby, to může vyžádat výpis jednoho nebo několika segmentů na disk.
Stránkování – reálná paměť je implicitně dělena na rámce pevné délky, virtuální paměť se dělí na stránky implicitně, má vnitřní fragmentaci v rámcích, žádnou externí, OS udržuje tabulku stránek s určením rámce pro každou stránku a tabulku rámců definující stav každého rámce. Paměť je rozdělena na stejné úseky – stránky, nemohou vznikat různě velké díry při vymazávání, vymaže se nejdéle nepoužitá stránka, nevýhodou je dlouhý proces při hledání správné stránky, proto se používá segmentace. Stránky procesu se zavádějí do reálné paměti podle potřeby, to může vyžádat výpis některé stránky na disk.

5. *Preemptivni a nepreemptivni planovani - napsat priklady*

Preemptivní plánování – s předbíháním, jakmile se ve frontě ready objeví proces, s délkou dávky CPU kratší než je doba zbývající k dokončení dávky právě běžícího procesu, nový proces předběhne právě běžící proces. K předběhnutí dojde na základě vnější události (přerušení). Provádí se při změně stavu některého z procesů, proces může mít stavy vytvořený a připravený, běžící, blokován (čekající), ukončený, může přecházet ze stavu běžící na čekající, z čekající na připravený,…
Nepreemptivní plánování – bez předbíhání, jakmile se CPU, předá vybranému procesu, tento nemůže být předběhnutý žádným jiným procesem, dokud svoji dávku CPU, nedokončí.
Dojde k němu za situace, kdy proces přechází se stavů:

Běžící -> připravený – přerušený proces
Běžící -> čekající – čeká na nějakou událost, na V/V operaci

Čekající -> připravený – například při dokončení V/V operace
Ukončený (terminated) – proces ukončil svůj běh
Provádí se po dokončení připraveného plánu, plán pro vytvoření procesu až po jeho dokončení

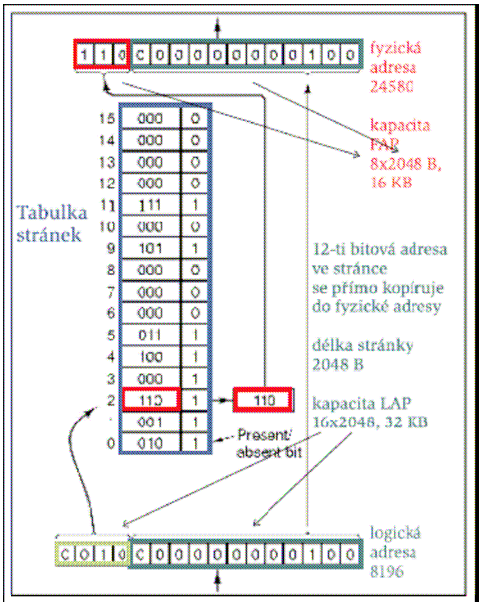
6. *Uvaznuti - co to je a napis moznosti jak mu predejit*

Uváznutí = trvalé blokování postupu skupiny procesů
Uváznutí nastává při existenci blokovaných procesů, které vlastní nějaký prostředek (zdroj) a čekají na zdroj držený jiným procesem z této množiny. Nastane, když začnou současně platit 4 podmínky: (první 3 jsou nutné, další je postačující)
1. Vzájemné vyloučení – sdílený zdroj může v jednom okamžiku použít pouze jeden proces
2. Postupné uplatňování požadavků – proces vlastnící alespoň jeden zdroj čeká na získání dalšího zdroje, dosud vlastněného jiným procesem
3. Nepřipouští se předbíhání – zdroj lze uvolnit pouze procesem, který ho vlastní, dobrovolně poté, co daný proces zdroj dále nepotřebuje
4. Zacyklení požadavků – existuje množina čekající procesů, že poslední v řadě čeká na uvolnění zdroje procesu před ním, ale ten čeká na uvolnění zdroje procesu také před ním a první v řadě čeká na uvolnění zdroje od procesu posledního
Neexistuje efektivní řešení problému.
Ochrana před uváznutím:

1. prevence - systém se nikdy nedostane do stavu uváznutí, ruší se některá z nutných podmínek
    - uspořádání pořadí přidělování prostředků
    - odstranění sdílených zdrojů
    - kdykoliv proces něco požaduje, nesmí vlastnit žádný prostředek
    - uvolňování prostředků před žádostí dalšího, zařazení procesů do fronty, aktivace při dostupnosti všech potřebných prostředků
    - použití semaforu – nepustí další proces k datům, se kterými se pracuje
    - obcházení uváznutí – detekuje potencionální vznik uváznutí a takový stav nepřipustí, zamezuje se současné platnosti nutných podmínek, prostředek se nepřidělí, pokud by hrozilo uváznutí nebo zestárnutí
    - detekce uváznutí a obnovení stavu před uváznutím
    - ignorování hrozby uváznutí

7. *Tabulka stranek - popis a nakresli*

- Pomocí tabulky stránek se realizuje přechod z logické adresy na fyzickou
- V tabulce se udržují informace o mapování LA->FA v odpovídající datové struktuře
- Jako minimální jednotka se používá stránka, protože udržování informací pro jeden bajt by bylo paměťově neefektivní
- Stránka je tvořena posloupností n bajtů, její velikost je vždy mocnina 2, na 32 bitových architekturách je typicky 216 = 4096 bajtů
- Tabulka stránek je datová struktura, která umožní rychlé vyhledání záznamu podle zadaného čísla logické stránky
- Obsah tabulky nastavuje OS
- Je uložena v operační paměti a odkazována registrem PTBR



8. *Princip kontextu procesu*

Přepnutí procesoru (CPU) na jiný proces vyžaduje uložení stavu starého procesu, který dosud používal CPU, a nahrání uloženého stavu nového procesu. Toto se označuje jako přepnutí kontextu. Stavy procesů si OS uchovává v tabulce procesů – seznam PCB. Stav starého procesu se uloží do jeho PCB a stav nového procesu se načte z jeho PCB.

9. Konceptni rozdil mezi OS s jadrem a OS s mikrojadrem

Jádro

- je to základní složka OS
- odpovídá za přidělování a správu zdrojů, přímé ovládání HW a bezpečnost
- operuje jako samostatná entita v privilegovaném režimu
- po dobu řešení pro jeden proces je jádro ostatním procesům nedostupné
- programy a data OS jsou ve stejném sdíleném adresovém prostoru a sdílí je všechny uživatelské procesy
- složité testování, opravy,…

Mikrojádro

- funkcí jádra je procesy separovat a umožnit jejich kooperaci
- malé jádro, dělá jen nejnutnější procesy, ostatní převádí do uživatelského rozhraní
- mikrojádro plní pouze nezbytné funkce jako primitivní správu paměti a komunikace mezi procesory, většina se přesune do uživatelského rozhraní jako vizualizace paměti, drivery
- mezi uživatelskými systémy se komunikuje předáváním zpráv
- OS se snadněji přenáší na nové architektury procesorů
- má vyšší spolehlivost, protože se skládá z velmi malých modulů, které zodpovídají za konkrétní operace
- jsou snadněji testovatelné
- má vyšší bezpečnost díky tomu, že se méně programů řeší v režimu jádra
- lze snadno doplňovat nové služby a odstraňovat nepotřebné
- podpora distribuovatelnosti

10. Procesor s prerusenim, vysvetli princip mikroprocesorovych instrukci

Přerušení normálního běhu procesoru – přerušení znamená, že se potlačí provádění běžícího procesu v CPU takovým způsobem, aby ho bylo možné později obnovit, cílem je zlepšení účinnosti, při přerušení je potřeba provést jinou posloupnost příkazů
V době řešení V/V operace se umožní, aby CPU provádělo jiné instrukce, které nepotřebují čekat na V/V, činnost CPU se později přeruší díky V/V modulu
CPU testuje možnost přerušení alespoň po každém provedení instrukce

2. Přerušení předává řízení správci přerušení (interrupt handler) prostřednictvím vektoru přerušení
3. Vektor přerušení obsahuje adresy všech vstupních bodů správců přerušení
4. Mechanismus přerušení musí uchovat adresu přerušené instrukce
5. Aby se zabránilo ztrátě přerušení, jsou během obsluhy přerušení přicházející přerušení maskována (disabled)
6. Operační systém je systém řízený přerušeními

11. Zobrazeni logicke adresy na fyzickou metodou tabulky na bazi strankovani

Pro zpřístupnění fyzické paměti se provádí tzv. mapování logických adres na fyzické. Při mapování na bázi stránkování se mapuje po stránkách (stejně velké oblasti paměti pevné délky). Stránka je nejmenší pamětí přidělování procesům a její velikost musí být mocninou 2. Adresový prostor si můžeme představit jako pole stránek, jehož indexy odpovídají číslu stránky. LAP i FAP jsou rozděleny na stejně velké stránky. Stránce ve FAP se říká rámec. Výhodou je, že se proces nemusí o mapování starat (je pro něj neviditelné), je zajišťováno kombinací HW a jádra OS. Tento způsob vizualizace je dnes využíván u téměř všech OS.

12. Rozdil mezi programem, procesem, vlaknem

Program – soubor přesně definovaného formátu obsahující instrukce, data a služební údaje potřebné k provedení stanoveného úkolu.
Proces – jednotka realizace výpočtu podle programu, všechna data má privátní, sdílen je pouze program, prostor v reálné paměti je přidělován procesům, ne programu, může vlastnit soubory, V/V zařízení nebo komunikační kanál k jiným procesům,
Vlákno – podproces procesu, systémový objekt, který se vytváří v rámci procesu, viditelný pouze uvnitř procesu, většina dat je sdílena, má minimum vlastní paměti, může přistupovat k virtuální paměti a k ostatním zdrojům své paměti, jakmile jedno vlákno změní obsah některé buňky ve virtuální paměti, ostatní vlákna nový obsah vidí, skupina vláken jednoho procesu sdílí proměnné, otevřené soubory,…

13. Cyklicke planovani a dusledky pri prodluzovani a zkracovani casovych dilu /ROUND ROBIN/

Round Robin

- každý proces dostává CPU na malou jednotku času – časové kvantum, po uplynutí této doby je běžící proces předběhnut nejstarším procesem ve frontě připravených procesů a zařazuje se na konec této fronty
- je-li ve frontě n připravených procesů a časové kvantum je q, pak každý proces získává 1/n-tinu doby CPU, najednou nejvýše po q časových jednotkách.
- žádný proces nečeká na přidělení CPU déle než (n–1)q časových jednotek

Výkonnostní hodnocení

- q velké → podobné jako FIFO (dlouhá doba čekání)
- q malé → nevýhodné kvůli režii přepínání kontextu

14. semafor jako synchronni \ semafor jako reseni kriticke sekce - pocatecni stav

Semafor jako nástroj řešící problém KS: počátečně semafor=1; když proces „dojede“ ke KS zjistí jestli je semafor==1, pokud ano tak jeho hodnotu změní na 0 (pomocí fce. WAIT), provede KS a změní hodnotu semaforu opět na 1 (pomocí fce. SIGNAL). nastává při problému souběžnosti více procesů, kdy procesy používají stejnou paměť a stejné proměnné, musí se zajistit, aby nevyšel nedeterministický výsledek, kritická sekce je, že proces k datům nikoho jiného nepustí

Semafor jako synchronizátor: počátečně semafor=0; proces který má provést určitou sekci jako první, po jejím provedení změní hodnotu semaforu na 1(pomocí fce. SIGNAL). Proces který má sekci provést jako druhý čeká dokud nebude semafor==1 (pomocí fce. WAIT), poté sekci provede. používá proměnné typu Integer a dvě operace – čekej na událost (čekej na zvednutí semaforu a zvednutý shoď), oznam událost (zvednutí semaforu)

15. Petersonuv algoritmus, napis a vysvetli, zda splnuje kriteria pro kritickou sekci.

Procesy zároveň projeví zájem o určitá data a čekají na sebe navzájem, tím uváznou, toto se rozbíjí náhodností výsledku paralelně zapsané do společné proměnné více vlákny
Nesplňuje podmínku spravedlnosti, vlákna sice neuváznou, ale mohou stárnout, o vítězi rozhoduje náhodná hodnota proměnné -> nesplňuje kritéria pro kritickou sekci
- Petersonův algoritmus splňuje všechny tři podmínky (podm. vzájemného vyloučení, podm. trvalosti postupu, podm. konečnosti doby čekání) správnosti řešení problému kritické sekce.

16. 4 nutne pozadavky k uvaznuti procesu

1. Vzájemné vyloučení – sdílený zdroj může v jednom okamžiku použít pouze jeden proces
2. Postupné uplatňování požadavků – proces vlastnící alespoň jeden zdroj čeká na získání dalšího zdroje, dosud vlastněného jiným procesem
3. Nepřipouští se předbíhání – zdroj lze uvolnit pouze procesem, který ho vlastní, dobrovolně poté, co daný proces zdroj dále nepotřebuje
4. Zacyklení požadavků – existuje množina čekající procesů, že poslední v řadě čeká na uvolnění zdroje procesu před ním, ale ten čeká na uvolnění zdroje procesu také před ním a první v řadě čeká na uvolnění zdroje od procesu posledního

17. Charakterizujte vlastnosti cyklického plánování a uveďte důsledky zkracování a prodluľování časových dílů přidělovaných procesům

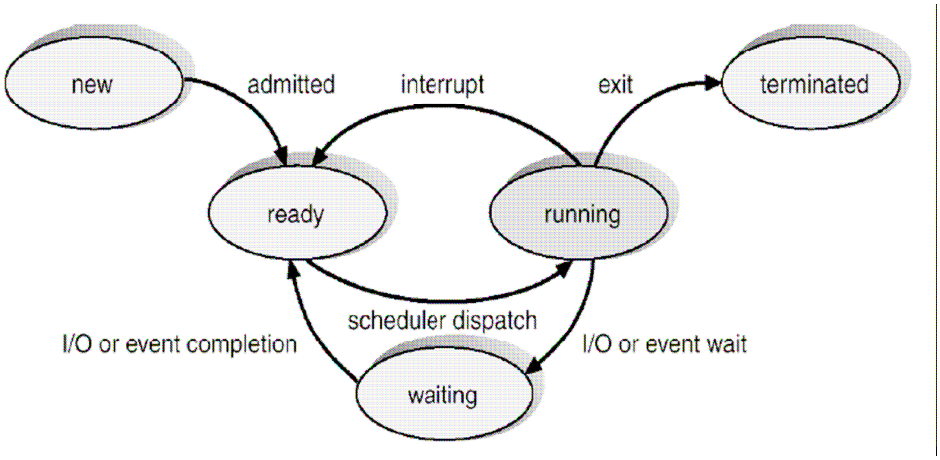
Každý proces dostává CPU cyklicky na malou jednotku času – časové kvantum (desítky až stovky milisekund), po uplynutí této doby je proces předběhnut nejstarším procesem ve frontě připravených procesů a dosud běžící proces se zařazuje nakonec fronty
Pokud máme ve frontě n procesů, každý proces dostane 1/n-tinu doby CPU, žádný proces nečeká na přidělení CPU déle než (n-1)q časových jednotek
Fronta připravených procesů je implementována jako kruhová fronta. Plánovač postupně prochází frontu a každému procesu přidělí procesor nejvýše po dobu jednoho kvanta. Nově příchozí procesy jsou zařazeny na konec a procesor se přiřazuje od začátku fronty. Po přidělení procesoru se nastaví časovač na přerušení po uběhnutí jednoho kvanta.
Při běhu procesu mohou nastat tyto dva případy:
Doba obsazení procesoru je menší než časové kvantum – proces se sám vzdává procesoru (např. při V/V operaci), plánovač vybere další proces ve frontě a přiřadí mu procesor.

Doba obsazení procesoru je větší než časové kvantum – po uplynutí časového kvanta časovač generuje přerušení, dojde k přerušení kontextu a zařadí se do fronty připravených procesů
q velké – dlouhá doba čekání
q malé – režijní ztráty při přepínání kontextů

18. Vyjádřete strukturální schéma propojeni procesoru, vnitřní paměti a periférie řízené principem DMA a stručně vyloľte princip operací DMA

DMA se používá u velmi rychlých V/V zařízení pro přenos dat z/ do pamětí rychlostí blízkou rychlosti vnitřní paměti
Řadič periferie přenáší bloky dat mezi vyrovnávací pamětí a periferií bez zásahů ze strany CPU – kradení cyklů
Přerušení se generuje po přenesení celého bloku, ne po každé dílčí jednotce (byte)

19. Nakreslete stavový diagram procesu bez odkládání procesů a stručně charakterizujte jednotlivé stavy a přechody mezi stavy



Stavy:
Nový (new) – právě vytvořený proces
Běžící (running) – proces je přiřazen procesoru a běží jeho instrukce
Čekající (waiting) – proces čeká na výskyt nějaké událost (např. dokončení V/V operace)
Připravený (ready) – proces čeká na své přiřazení rpocesoru
Ukončený (terminated) – proces ukončil svůj běh
V každém stavu může být současně více procesů, ale v každém okamžiku může být procesoru přidělen pouze jeden proces
Přechody mezi stavy:
Nový -> připravený
Připravený -> běžící – přiřazen procesoru
Běžící -> připravený – přerušený proces
Běžící -> čekající – čeká na nějakou událost, na V/V operaci
Běžící -> ukončený – dokončený proces
Čekající -> připravený – například při dokončení V/V operace

20. Vymezte oblast působení dlouhodobého, střednědobého a krátkodobého plánovače

Dlouhodobý plánovač (strategický plánovač) – vybírá, který požadavek na výpočet lze zařadit mezi procesy, je vyvoláván řídce (sekundy až minuty), nemusí být rychlý
Střednědobý plánovač (taktický plánovač) – taktika využívá omezené kapacity reálné paměti při multitaskingu, vybírá, který proces je nutné zařadit mezi odsunuté procesy (odebírá mu prostor v reálné paměti), vybírá, kterému odsunutému procesu lze opět přidělit prostor v reálné paměti
Krátkodobý plánovač (operační plánovač, dispečer) – reprezentace správy procesorů, vybírá proces, který poběží na uvolněném procesoru, přiděluje procesu procesor, vyvoláván velmi často, musí být rychlý

21. *Vysvětlete rozdíl mezi pojmy vlákno typu User-Level Threads (ULT) a vlákno typu Kernel-Level Threads (KLT)*

ULT – jádro OS podporuje procesy, vlákna podporuje vláknová knihovna, která obsahuje funkce pro vytváření a rušení vláken, předávání zpráv a dat mezi vlákny, plánování běhů vláken, uchovávání a obnova kontextu vláken. Jádro neví o aktivitě vláken, manipuluje pouze s celými procesy, pokud některé vlákno zavolá službu jádra, dokud se služba nesplní, je blokován celý proces, stavy vláken jsou na stavech procesu nezávislé.
KLT – podporuje je přímo jádro OS. Celou správu vláken podporuje jádro, nepoužívá se vláknová knihovna, používá se API na vláknové služby jádra, informace o kontextu, procesu a vláknech udržuje jádro, přepojování mezi procesy aktivuje jádro, plánování se řeší na bázi vláken. Jádro může současně plánovat běh více vláken stejného procesu na více procesorech, k blokování dochází na úrovni vláken. Pomalejší, snížení dostupného aplikačního výkonu.

22. *Uveďte a charakterizujte tři základní podmínky správného řeąení problému kritické sekce.*

Podmínka bezpečnosti – dosažení vzájemného vyloučení – jestliže jeden proces provádí svoji kritickou sekci, nemůže ji jiný proces provádět se stejnými zdroji
Podmínka životnosti – trvalost postupu – jestliže žádný proces neprovádí svoji kritickou sekci se sdruženým zdrojem, pak se výběr procesu, který chce vstoupit do kritické sekce sdružené s tímto zdrojem, nesmí odkládat do nekonečna
Podmínka spravedlnosti – konečnost doby čekání – musí existovat horní mez počtu vstupu do kritické sekce sdružené s určitým zdrojem jiným procesům, než procesu, který vydal žádost o vstup do kritické sekce sdružené s tímto zdrojem, po vydání takové žádosti a předtím, než je takový požadavek uspokojen

23. *Definujte uváznutí procesů*

- **Definice:** množina procesů P uvázla <=> každý proces P_i z P čeká na událost (uvolnění prostředku, zaslání zprávy), kterou vyvolá pouze některý z procesů v P (prostředek - paměťový prostor, IO zařízení, soubor, …)
- K uváznutí dojde, když začnou současně platit 4 následující podmínky: vzájemné vyloučení, postupné uplatňování požadavků, nepřipouští se předbíhání, zacyklení pořadí (první tři jsou nutné podmínky poslední je postačující)

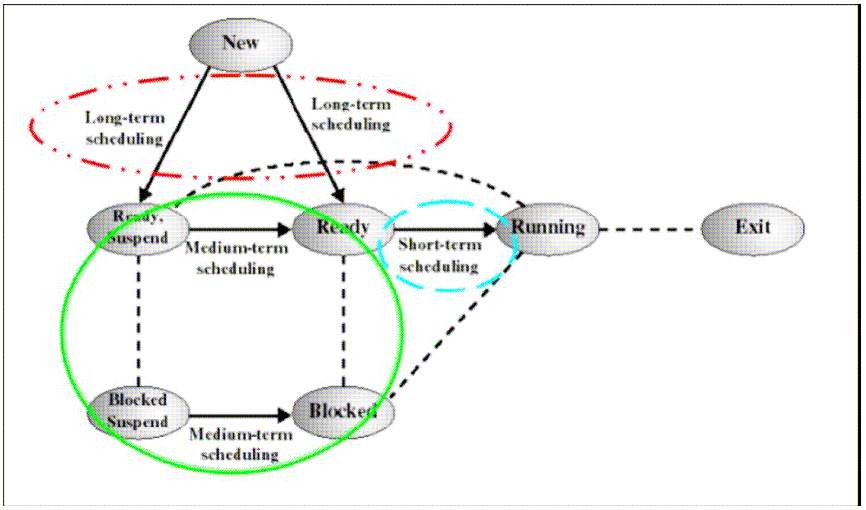
24. *Charakterizujte princip preventivních metod ochrany proti uváznutí a uveďte alespoň dva příklady jejich použití*

- snaha o zneplatnění některé nutné (postačující) podmínky
- omezují způsob provedení požadavku
- **ne vzájemné vyloučení**
  - řeší se virtualizací prostředků, pokud to jde (tiskárny, …)
  - nepožaduje se pro zdroje, které lze sdílet
  - musí se dodržet při používání opakovaně dostupných zdrojů
- **ne postupná alokace**
  - musí se zaručit, aby proces žádající nějaký zdroj žádný jiný zdroj nevlastnil
  - proces musí požádat o zdroje a obdržet je dříve než se spustí jeho běh nebo o ně smí žádat pouze tehdy, když žádné zdroje nevlastní
  - důsledek: nízká efektivita využití zdrojů, možnost stárnutí
- **ne předbíhání**
  - jestliže proces držící nějaké zdroje a požaduje přidělení dalšího zdroje, která mu nelze přidělit okamžitě, pak se uvolní všechny, tímto procesem držené zdroje v tomto okamžiku
  - „odebrané“ zdroje se zapíší do seznamu zdrojů, na které proces čeká
  - proces bude obnoven pouze když může získat jak jím původně držené zdroje, tak jím požadované zdroje
- **ne zacyklení pořadí -** zavede se úplné uspořádání typů zdrojů a každý proces požaduje prostředky v pořadí daném vzrůstajícím pořadí vyčtu

25. Charakterizujte privilegovaný a uživatelský režim činnosti procesu a zdůvodněte používaní dvou režimů

Uživatelský režim – omezený repertoár povolených instrukcí
Privilegovaný režim – připouští se provádění plného repertoáru instrukcí
Důvod: Sdílení systémových zdrojů požaduje, aby OS měl záruku, že nesprávný program negativně neovlivní běh ostatních programů, z uživatelského režimu jsou proto vyjímány V/V operace a operace ovlivňující stav systémových zdrojů (registry ochrany, …)

26. Nakreslete stavový diagram procesu s odkládáním procesů a stručně charakterizujte jednotlivé stavy a přechody mezi stavy



Stavy:
Nový (new) – právě vytvořený proces
Běžící (running) – proces je přiřazen procesoru a běží jeho instrukce
Čekající (waiting) – proces čeká na výskyt nějaké událost (např. dokončení V/V operace)
Připravený (ready) – proces čeká na své přiřazení rpocesoru
Ukončený (terminated) – proces ukončil svůj běh
+ další dva stavy
Odložený připravený
Odložený čekající
Přechody mezi stavy:
Čekající -> odložený čekající – uvolní jinak nepoužívané části FAP
Odložený čekající -> odložený připravený – stala se očekávaná událost, proces zůstává odložený
Odložený připravený -> připravený – uvolnil se prostor ve FAP
Připravený -> odložený připravený – uvolnil se prostor ve FAP, ale pro procesy s vyšší prioritou

27. Zdůvodněte používání odkládání (swapping) procesů

Běžící proces musí mít alespoň aktuální části virtuální paměti přidělené reálné, ta ale není dost velká. Velké množství procesů v RAM snižuje výkonnost, proto musí RAM některé procesy odložit, když je plná. OS je odloží na disk.
Odložení – swap-out – plácnutí na disk
Obnovení – swap-in – plácnutí do RAM
Tímto přibývají další dva stavy procesů, odložený připravený a odložený čekající.

28. Vysvětlete heuristiku odhadování časových dílů přidělovaných plánovačem procesů na bázi exponenciálního průměrování

Odhad pravděpodobné délky příští dávky CPU se odvodí z historie chování procesu. Musí se znát předchozí odhady délek dávek CPU, musí se znát, jak proces využíval přidělená kvanta CPU, použije se exponenciální průměrování. Je to exponenciální průměr odhadnutých dob předchozích obsazení procesoru.
Exponenciální průměrování:

tn – skutečná délka n-té dávky CPU
tn+1 – odhad délky příští dávky CPU
a, $0 \leq a \leq 1$ – parametr vlivu historie
$t_{n+1} = a t_n + (1 - a) t_n$

29. Ilustrujte princip použití semaforu jako synchronizačního nástroje pro řízení běhů procesů

Nejčastěji využívaný prostředek pro synchronizaci, který nevyžaduje aktivní čekání. Semafor může vpouštět procesy do kritické sekce po jednom. Při obsazování kritické sekce se hodnota semaforu snižuje, při uvolňování zase zvyšuje. Je-li hodnota semaforu kladná, udává počet procesů, které je ještě možné vpustit do kritické sekce. Je-li nulový, znamená to, že kritická sekce je neobsazená a nikdo na ni nečeká. Pokud je hodnota záporná, že kritická sekce je obsazená a že na ni čekají další procesy. Proces, který pomocí operace unlock uvolňuje kritickou sekci, zjistí zda někdo nečeká ve frontě spojené se semaforem. Pokud ano, odstraní první čekající proces z fronty a aktivuje ho.

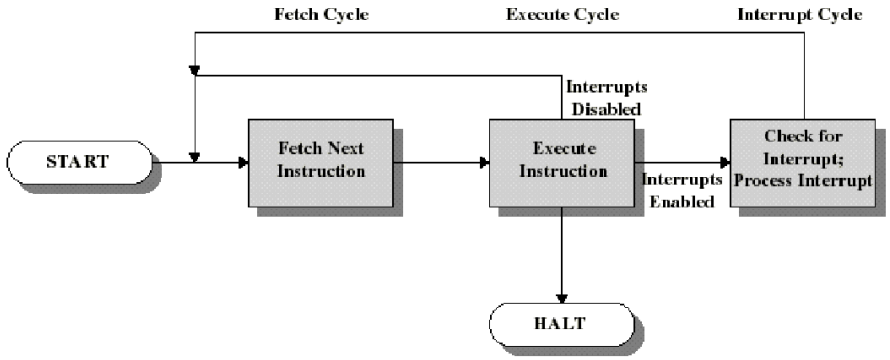
30. Definujte stárnutí procesů

Požadavky jednoho nebo více procesů z množiny procesů nebudou splněny v konečném čase, např. z důvodů priorit procesů

31. Charakterizujte princip a vlastnosti metod ochrany proti uváznutí obcházením podmínek vedoucích k uváznutí

Základním předpokladem je, aby každý proces udal maximální množství prostředků, které může potřebovat. Algoritmus řešící obcházení uváznutí dynamicky zkouší, zda se procesy nedostanou do zacyklení. Pro každý proces, který žádá o prostředek, musí systém rozhodnout, jestli přidělení prostředku zanechá systém v bezpečném stavu, ten je když existuje bezpečná posloupnost procesů. Jestliže je systém v bezpečném stavu, k uváznutí nedojde. Výkon správy zajišťuje, že systém nikdy nepřejde do stavu, který není bezpečný, nespouští potencionálně nebezpečný proces, neprovádí potencionálně nebezpečné přidělení.

32. Instrukční cyklus - zobrazit popsat správně jednotlivé součásti



33. Co znamená stránkování a segmentace ve správě paměti? Vypsat v čem se shodují a v čem jsou rozdílné.

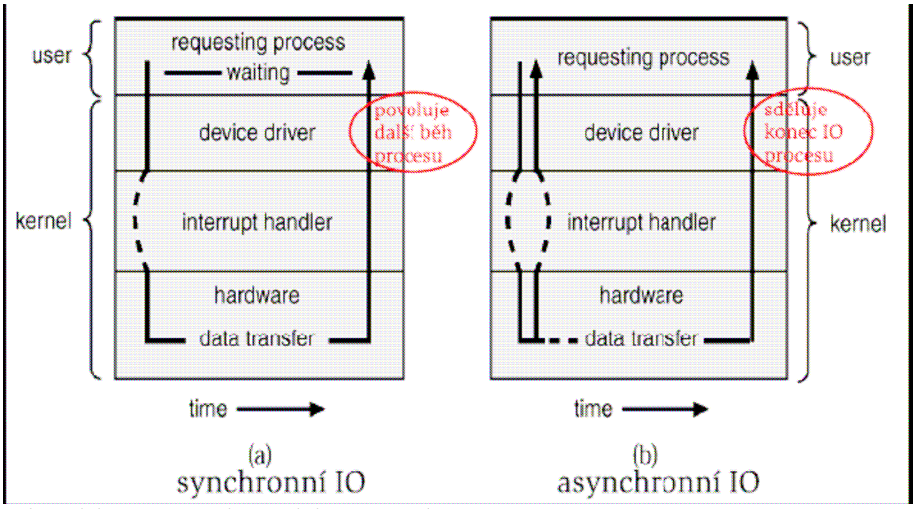
Používá se při dynamických výměnách částí mezi virtuální a reálnou pamětí. Stránkování a segmentaci máme prostou a na žádost (nepřítomnost adresovaného objektu v reálné paměti). Musí je podporovat HW správy paměti. OS musí být schopen organizovat tok stránek / segmentů mezi vnitřní a vnější pamětí.
Segmentování – stránky se uspořádávají do logických celků (OS vytvoří celek, o kterém si myslí, že pracuje společně), reálná paměť není implicitně dělena, použité segmenty virtuální paměti určuje programátor / kompilátor, žádná vnitřní fragmentace (rozdělení na díly), možnost externí fragmentace. OS udržuje tabulku segmentů s bází a délkou každého segmentu a udržuje seznam volných oblastí reálné paměti. Převod z virtuální paměti se řeší dynamicky při běhu mikroprogramem podle tabulky segmentů. Zavádějí se do reálné paměti podle potřeby, to může vyžádat výpis jednoho nebo několika segmentů na disk.
Stránkování – reálná paměť je implicitně dělena na rámce pevné délky, virtuální paměť se dělí na stránky implicitně, má vnitřní fragmentaci v rámcích, žádnou externí, OS udržuje tabulku stránek s určením rámce pro každou stránku a tabulku rámců definující stav každého rámce. Paměť je rozdělena na stejné úseky – stránky, nemohou vznikat různě velké díry při vymazávání, vymaže se nejdéle nepoužitá stránka, nevýhodou je dlouhý proces

při hledání správné stránky, proto se používá segmentace. Stránky procesu se zavádějí do reálné paměti podle potřeby, to může vyžádat výpis některé stránky na disk.

34. Jakými způsoby se předávají parametry volání služeb systému?

V registrech – registry jsou dostupné procesu i OS
V tabulce uložené v hlavní paměti – adresy tabulky se umístí v registru
V zásobníku – je dostupný procesu i OS

35. znázorněte princip synchronního a asynchronního řešení V/V operací na úrovni procesů v počítači řízeném operačním systémem



Synchronní – uživatelský proces čeká na dokončení V/V operace
Asynchronní - uživatelský proces nečeká na dokončení V/V operace

36. vysvětlete SJR

Plánování Shortest-Job-First – k definici procesu se doplní délka jeho příští dávky CPU, vybírá se proces s nejkratší dobou dávky CPU, používají se dvě varianty, preemptivní (s předbíháním) a nepreemptivní (bez předbíhání). S hlediska průměrné doby čekání je SJF optimální algoritmus. Pokud při vybírání jsou délky následného obsazení procesoru stejné, vybírá podle algoritmu FCFS.
Bere nejrychlejší požadavky nejdříve, vybere z fronty vždy požadavek, který je nejblíže předchozímu vyřizovanému. Nevýhoda spočívá v tom, že když bude do fronty neustále přicházet dostatek požadavků nacházejících se v jedné oblasti disku, všechny ostatní požadavky zůstanou nevyřízeny. To je v praxi nepřijatelné.

37. Popiš činnost procesoru

- Získává instrukce z paměti, dekóduje je a provádí.
- Množina instrukcí je specifická pro jistý typ procesoru
- Procesor je vybaven rychlými pamětmi – registry, které obsahují klíčové proměnné, dočasné výsledky, data nutná pro řízení běhů výpočtu,…

Činnost:

- Interpretuje instrukce uvedené v programu
- Instrukce se získávají z operační paměti (FAP) sekvenčně
- Ukazatelem na příští získávanou instrukci z FAP je PC

38. Striktní a tolerantní RT systém

Přísné Hard real-time systems – omezená/žádná vnější paměť, data se pamatují krátkodobě v RAM paměti nebo v ROM, protipól interaktivních univerzálních systémů, univerzální OS provozování přísných RT systémů nepodporují, plánování musí respektovat požadavek ukončení kritického úkolu v rámci garantovaného časového intervalu

Tolerantní Soft real-time systems – omezená použitelnost, např. v robotickém průmyslovém řízení, použitelné v aplikacích požadujících dosputnost rysů propracovaných OS (multimedia, virtual reality), krtické procesy mají přednost před „méně šťastnými" procesy

39. Fetch Policy

Kdy stránku zavádět? Při virtualizaci paměti, dvě politiky

Stránkování na žádost

- stránka se zobrazuje do FAP při odkazu na ni, pokud ve FAP není již zobrazena
- způsobuje počáteční shluky výpadků stránek

předstránkování

- umístění sousedních stránek v LAP v obrazu LAP na vnější paměti bývá rovněž blízké
- platí princip časo-prostorové lokality, sousední stránky v LAP se s velkou pravděpodobnosti používají v blízkém časoprostoru
- zavádí se více sousedních stránek než se žádá
- předstránkování je vhodná politika zvláště při inicializaci procesu

40. Placement policy

Kam stránku / segment umístit v FAP?
Stránkování – nemá význam, pro umístění stránky se volí libovolný volný rámec
Segmentace – politiky first-fit, best-fit, worst-fit, next-fit, …

41. Replacement policy

Politika nahrazování – uplatňuje se v okamžiku, kdy je reálná paměť (FAP) plná, není dostupný neobsazený rámec
Kterou stránku „obětovat" a vyhodit z FAP? – také Politika výběru oběti řízená algoritmem pro výběr oběti