2020

Okruhy otázek IOSYS 2020

FAKULTA ELEKTROTECHNIKY A INFROMATIKY UNIVERZITY PARDUBICE

1) Definice OS: typy OS; design OS: abstrakce a operace nad nimi – služby, systémová volání

a) Dvě základní funkce OS (s vysvětlením) 2

Rozšíření stroje (virtualizace) – pohled „shora“

- Zjednodušující interface, abstrakce
- Příklad: čtení/zápis na disk

Správce prostředků – pohled „zdola“

- Procesory, paměť, V/V zařízení
- Příklad: tisk na tiskárně
- Multiplexing – sdílení prostředků:
  - V čase (CPU)
  - V prostoru (RAM)

b) Vlastnosti moderního OS 4

- Preemptivní plánování procesů
- Izolace procesů
- Efektivní správa paměti
- Izolace uživatelů
- Podpora IPC

c) Typy OS (alespoň šest) 3

- Mainframe OS – sálové PC
- Serverové OS
- Vícepočítačové OS – clustery
- Osobní OS
- RealTime OS
- Vestavěné
- Smart Cart OS

d) Reprezentace abstrakcí (koncepcí) v OS a definice systémového volání 2

Systémové volání

- Volání služeb jádra OS
- Volání přes knihovnu

e) Průběh systémového volání 3

- Uložení parametrů na stack
- Volání systémové funkce v knihovně
- Knihovna nastaví registr na typ volání
- TRAP (skok do jádra)
- Jádro volá příslušný ovladač
- Návrat do knihovny a programu

f) Příklady systémových volání a co obstarvají (alespoň tři) 1

Pro procesy: vznik, nahrazení, čekání na uložení; pro soubory: otevření, zavření, čtení, zápis; pro adresáře: vytvoření, zrušení, připojení FS; ostatní (práva, signály): změna práv, signál, zjištění času.

2) Architektura jádra OS: monolitický systém, vrstvený systém, virtualizace na úrovni jádra OS, mikrojádro

a) Definice monolitického jádra OS 2

- Vše v jednom – vnitřně nečleněné jádro
  - Z hlediska hierarchie volání procedur
  - Každá procedura může volat libovolnou jinou
- Mohou mít i strukturu:
  - Hlavní program (obsahuje dispatcher)
  - Obslužné procedury
  - Užitkové procedury
- Procedury mají pevně definované rozhraní

b) Definice vrstveného jádra OS a čím se liší od monolitického s vnitřní strukturou vrstev 2

- Vrstva smí volat jen procedury stejné nebo nejbližší nižší vrstvy – zajišťuje HW (úrovně oprávnění)
- **Vrstvy:**
  - 5 – operátor
  - 4 – uživatelské programy
  - 3 – správa V/V zařízení
  - 2 – komunikace mezi procesy a konzolí operátora
  - 1 – správa paměti
  - 0 – alokace CPU a multiprogramming

c) Definice mikrojádra OS a které funkce zajišťuje 3

- Jedná se o trend moderních OS
  - Mikrojádro spravuje komunikaci mezi procesy
  - Klientské procesy spravují paměť, souborový systém a ovladače
- Náročné na implementaci
- **Zajišťuje:**
  - Mikroprogramování – alokace CPU
  - Ochrana paměti
  - Základní meziprocesová komunikace

d) Jak jsou v OS s mikrojádrem zajištěny ostatní funkcionality (které nejsou v mikrojádru) 2

e) Definice virtualizace na úrovni jádra OS s příkladem 3

- Jedná se o virtuální stroje
- Srdcem je VM monitor – multiprogramming
- Jedno jádro OS + kontejnery (izolace skupin procesů, uživatelů, sítě i souborových systémů)
- Solaris, FreeBSD, OpenVZ, Linux-Containers

f) Příklady OS různých architektur jádra (monolitické, vrstvené, mikrojádro) 3

Monolitické: Windows, MacOS, Linux
Vrstvené: Multics
Mikrojádro: QNX, Minix

3) *Návrh OS a jeho bezpečnost: důvody náročnosti implementace OS, principy jeho vývoje; zabezpečení systému, uživatelských dat a procesů (vyjma útoků na systém, to je jiný okruh)*

a) Hlavní obecné cíle návrhu OS 4

- Definování abstrakcí (definice procesů, souborů, abstrakce pro vlákna, signály, V/V zařízení, IPC,…)
- Poskytnutí základních operací (jednoduché operace nad strukturami (abstrakcemi), operace jsou přístupné skrze systémové volání)
- Zajištění izolace (Uživatelé si vzájemně nesmí zasahovat do svých procesů, prostředků a dat) – izolují se tedy procesy, poskytuje se řízení ochrany a nabízí se možnost sdílení
- Správa HW (čipy pro řadiče přerušení, systémová sběrnice, V/V zařízení, ovladače, abstrakce pro HW)

b) Důvody náročnosti návrhu OS 5

- Rozsáhlost, komplexnost, složitost
- Konkurence (uživatelů, procesů)
- Ochrana (nepřátelské prostředí)
- Sdílení dat a prostředků
- Životnost
- Obecnost použití
- Přenositelnost
- Zpětná kompatibilita

c) Principy vývoje OS 3

- **Jednoduchost**
  - Méně je více – KISS (Keep It Simple, Stupid)
- **Úplnost**
  - OS má dělat to, co má dělat, ale nic navíc (málo mechanismů s jasným chováním)
- **Jednoduchost a úplnost**
  - Každá část by měla dělat jen jednu věc a tu dělat dobře a správně
  - Pokud bude nová funkcionalita „možná potřeba“ je lepší ji zahrnout do knihovny, nikoliv do jádra

d) Způsoby zabezpečení procesů, dat na médiu a přenášených dat 3

- Domény ochrany
  - Stanovení domén a práv
- ACL – Access List Control
  - Stanovení seznamu práv na objekty pro uživatele

- Capability – C-list
  - Stanovení práv procesům
- Firewally a komunikační filtry
- Zabezpečení dat na médiu:
  - Zamezení přístupu (přístupová práva)
- Zabezpečení přenášených dat:
  - Šifrování, kódování

4) CPU: provádění instrukce, pipeline, atomicita a přerušitelnost procesu a průběhu zpracování instrukce, přerušovací systém, průběh zpracování přerušení, časovač, sdílení času

a) Popis fází provádění instrukce 4

FETCH – načtení instrukce

DECODE – dekódování instrukce

EXECUTE – provedení instrukce

WRITE – zápis výsledků

b) Definice pipeline a další možnosti zvyšování výkonu CPU 4

- Pipeline
  - Rozdělení zpracování instrukce na nezávislé kroky
  - Poté lze zahájit zpracování jedné instrukce ještě před dokončením předchozí
- Spekulativní provádění instrukce
- Sdílení subčástí CPU mezi vlákny
- Více jader

c) Přerušitelnost procesu, průběhu zpracování instrukce 2

d) Přerušovací systém: účel, průběh přerušení 5

- Umožňuje efektivní využití CPU
- Při výskytu důležité události procesor přeruší vykonávání hlavního programu a začne vykonávat obslužnou proceduru pro danou událost
- Po dokončení důležité události pokračuje vykonávání hlavního programu
- **Průběh přerušení:**
  - Dokončení rozpracované instrukce
  - Skok na obslužnou rutinu přerušení
  - Uložení kontextu
  - Obsluha přerušení
  - Návrat kontextu do procesoru

5) Vstupně-výstupní zařízení: ovladače a techniky programování vstupu a výstupu, DMA. Paměť cache, procesorová cache a střední přístupová doba do paměti

a) Metoda 1 komunikace se vstupně-výstupním zařízením (neefektivní) 2

Přístup pomocí instrukcí CPU

- Zápis na V/V zařízení, kontrola stavu, čekání

b) Metoda 2 komunikace se vstupně-výstupním zařízením (efektivnější) 2

Přístup pomocí instrukcí CPU s využitím přerušení

- Zápis na V/V zařízení, provádění jiných operací
- Po dokončení se generuje přerušení a pokračuje zápis

c) Metoda 3 komunikace se vstupně-výstupním zařízením (nejefektivnější) 3

Direct Memory Access (DMA) s využitím přerušení

- Zápis adresy a rozsahu dat v RAM do V/V zařízení, příkaz
- Provádění jiných operací
- Po dokončení: přerušení a obsluha
- Každé V/V zařízní má definovaný DMA kanál, který umožňuje přímý přístup do paměti RAM bez účasti procesoru

d) Definice paměti cache 2

- Rychlá paměť procesoru, využívá principu lokality odkazů v paměti
- Jedná se o rychlou mezipaměť (např. mezi procesorem a RAM)

e) Důvod existence konceptu paměti cache 1

Paměť RAM je pomalá a brzdí CPU, proto má každý CPU svou paměť cache.

f) Důvod efektivity používání cache 2

Do paměti cache se ukládá část aktuálně prováděného kódu procesu a část jeho dat. Procesor pak nemusí čekat na pomalou RAM.

g) Výpočet střední přístupové doby do paměti 3

$T_s = T_C + (1 - HR) \cdot T_{OP} (T_C \ll T_{OP})$

6) Požadavky OS na HW nutný pro jeho implementaci: zejména na procesor, správu a adresování paměti. Registry CPU

a) Nutné vlastnosti CPU a paměti pro implementaci OS a jejich účel 2 + 3

- **Požadavky na procesor**
  - Řadič nebo řídící jednotka (načítání, dekódování, ukládání)
  - Sada registrů (uchovávání operandů a mezivýsledků)
  - Aritmeticko-logická jednotka (provádění aritmetických a logických operací)
  - Paměť cache (zrychlení provádění procesů)
- **Požadavky na paměť**
  - Uvolňování paměti (ve chvíli, kdy proces paměť nepotřebuje je uvolněna)
  - Ochrana paměti (proces nesmí odkazovat bez povolení na paměťová místa přidělená jiným procesům)
  - Správa adresace paměti (odkazy na paměť programu se musí dynamicky překládat na fyzické adresy)
  - Logická organizace paměti (OS a HW musí podporovat práci se sekcemi – ochrana a sdílení)
  - Sdílení paměti (více procesů může sdílet dané úseky paměti bez narušení ochrany)

b) Nutné subsystémy HW pro implementaci OS (kromě CPU a paměti) a jejich účel 2 + 2

- **Přerušovací systém**
  - Efektivní využití CPU, nutné pro DMA
- **Časovač**
  - Pravidelně generuje přerušení, umožňuje preempci
- **Režimy kernel a user**
  - Ochrana paměti
  - Ochrana před neprivilegovanými operacemi v režimu user
- **Virtualizace paměti**
  - Logické adresování – relokace

c) Definice registrů CPU, jejich druhy 1 + 2

- Registr je malé a užitečné uložiště dat, které využívá procesor při své činnosti (registry jsou součástí CPU)
- CPU přesouvá data z RAM do registrů, aby je mohl zpracovat
- Druhy: obecné, datové, adresní, privátní

d) Stavový registr CPU a zahrnuté příznaky (alespoň tři) 3

- Speciální registr procesoru (PSW – program status word)
- Příznaky: C (carry), N (negative), Z (zero), V (overflow)

7) Implementace procesu v OS: proces a program, tabulka procesů, přepínání kontextu, stav procesu, třístavový model, příčiny změn stavů

a) Proces a program: definice 2

- **Program**
  - Postup operací, který popisuje realizace dané úlohy
- **Proces**
  - Instance programu v paměti
  - Pro běh potřebuje: procesor, vnitřní paměť + další prostředky (V/V zařízení, soubory,…)

b) Metadata procesu (alespoň šest) 3

- Adresový prostor: kód, stack, data, heap
- Přidělené prostředky: otevřené soubory, semafory
- Kontext (stav): registry CPU, mapování paměti
- Atributy: id, údaje plánovače, práva, časy, účtování

c) Průběh přepnutí kontextu 3

- Uložení kontextu
- Obsluha ovladačem v jádře
- Plánovač rozhoduje, který proces poběží poté

d) Třístavový model: popis stavů 3

- **Běžící** – aktuálně používá CPU
- **Připravený** – pozastaven jádrem CPU
- **Blokovaný** – čekající na vnější událost

e) Třístavový model: příčiny přechodů mezi stavy 4

- Nedostatek operační paměti
- Preempce
- Aktivace plánovačem
- Nastala událost

8) Procesy v OS UNIX/Linux: vznik a zánik procesu, systémová volání fork(2), exec(3), exit(3), wait(2), kill(2); hierarchie procesů, stavy procesů v Linuxu (podle příkazu ps); posixové signály a jejich zpracování

a) Příčiny vzniku procesu a spuštění nového programu v posixových systémech, související systémová volání, hierarchie procesů 4

- Původcem je jádro
  - Při inicializace systému
    - První proces (init)
    - Služby jádra (mikrojádrový OS)
- Původcem je proces
  - Systémové volání
    - Uživatel zadá příkaz, který interpret zpracuje tak, že se systémovým voláním vytvoří další proces
    - Proces může být aktivován i stiskem tlačítka
- Systémová volání
  - fork() – vytváří z rodičovského procesu nový proces, tak že se aktuální proces rozdvojí na dva identické procesy a běh programu pokračuje ve dvou nezávislých větvích
  - exec() – nahrazuje kód běžícího procesu procesem, který je uveden v parametru, typicky se volá po volání fork()

b) Příčiny zániku procesu a čekání na potomka v posixových systémech, související systémová volání 4

- Dobrovolné ukončení
  - Normální ukončení – úloha byla dokončena
  - Detekována chyba – například chybný vstup
- Nedobrovolné ukončení
  - Fatální chyba (neošetřená)
    - Neplatná adresa, nepovolená instrukce, dělení nulou, …
    - Jádro proces ukončí
  - Zabití jiným procesem
- Systémová volání
  - wait – donutí vyčkat jeden proces na dokončení jiného procesu
  - exit – volání jádra OS, pomocí které se ukončí činnost procesu
  - kill – slouží k okamžitému ukončení procesu (zabití procesu)

c) Stavy procesů v Linuxu 4

- R – připravený, běžící (runnable, running)
- S – blokovaný (sleep)
- Z – ukončený (defunct, zombie)
- T – pozastavený, krokovaný (stopped, traced)
- D – nepřerušitelný spánek (unniteruptible sleep)

d) Posixové signály, možnosti jejich zpracování a příklady 3

- Jednoduché zprávy
- Pouze pro posixové systémy
- Implicitně je posílá OS při daných událostech
- Explicitně se posílají příkazem kill()
- Zpracování:
  - Ukončení, pozastavení, pokračování procesu, ignorování
- SIGINT – exit – ukončení (CTRL + C)

- SIGSTP – stop – pozastavení (CTRL + Z)
- SIGCONT – resume – pokračování pozastaveného procesu
- SIGQUIT – core - ukončení

9) Vlákna: motivace zavedení vláken, proces × vlákno, možné implementace vláken, obecné problémy při implementaci a používání vláken; knihovna posixových vláken: mutex, bariéra, podmínková proměnná a jejich použití

a) Motivace zavedení vláken 1

- Kvaziparalelismus
- Jednodušší a rychlejší správa
  - pthread_create() je 50x rychlejší než fork()
- výkon – záleží na aplikaci
- užitečné na SMP

b) Společné a samostatné položky metadat procesů/vláken 3

Samostatné:

- Program counter a registry (uložený kontext procesu) – nutné, aby se mohlo každé vlákno vykonávat zvlášť (jinou funkci)
- Stav vlákna – každé vlákno může být v jiném stavu
- Stack (zásobník) – nutné kvůli nezasahování si vláken do svých výpočtů. Stack je využíván pro ukládání lokálních proměnných a parametrů funkcí.

Společné:

- Jsou sdílené v tabulce procesů (PCB)
- Adresní prostor, prostředky,…

c) Důvody vyhrazené části paměti pro každé vlákno 2

d) Implementace vláken s podporou OS a bez ní, výhody a nevýhody 4

- Bez podporou OS
  - Pomocí knihovních funkcí
  - Problémy:
    - Převedení blokujícícho volání na neblokující
    - Page-fault (stránka není v RAM)
    - Je třeba plánovač
  - Výhody:
    - Lepší režie – rychlejší vznik, přepnutí kontextu
    - Nevyžaduje se přechod do režimu jádra
    - Strategie plánovače se dá přizpůsobit aplikaci
  - Nevýhody:
    - Složitá implementace (neblokující volání, plánovač)
    - Page-fault zastaví všechna vlákna
- S podporou OS
  - Vzdálená volání – více režie
  - Není třeba neblokujících volání
  - Výhody:
    - Lze provádět i vlákno procesu, jehož jiné vlákno způsobilo page-fault
    - Není třeba neblokující volání
  - Nevýhody:
    - Horší režie – přechod do režimu jádra
    - Pevná strategie plánovače vláken

e) Komplikace při zavádění vláken 2

- Globální proměnné
- Nereentrantní volání knihovních funkcí

- Přístup ke sdíleným prostředkům (Kritická sekce)
- Znalost implementace signálů a jejich obsluhy
- Stack

f) Posixová knihovna vláken: nástroje pro řešení problémů souběhu a jejich účel 3

Mutex

Podmínková proměnná

Bariéra

10) Plánovač. Cíle plánování, režimy plánování, plánovací kriteria (cíle) pro plánovací algoritmy, plánovací algoritmy

a) Úloha plánovače, režimy plánování procesů 3

- Rozhoduje, který proces (vlákno) má CPU
- Řídí se plánovacím algoritmem
- Režimy plánování
- **Režimy**:
  - *Nepreemptivní* – proces se musí sám vzdát CPU
  - *Preemptivní* – plánovač rozhodne, kdy a který proces dostane CPU, předpokládá se přerušovací systém a časovač

b) Cíle plánování (obecně a dle určení OS) 3

Obecně:

- Spravedlnost
- Dodržování strategie
- Efektivní využití zdrojů

Dle určení OS:

- *Interaktivní systémy*
  - Minimalizace odezvy
  - Proporcionalita
- *Dávkové systémy*
  - Maximalizace propustnosti
  - Minimalizace obratu
  - Maximální využití CPU
- *RT systémy*
  - Respektování lhůt
  - předvídatelnost

c) Popis plánovacích algoritmů: historické 3

Fronta jednotlivých úloh

- Nepreemptivní
- Nové úlohy se řadí do fronty
- Po ukončení úlohy se přidělí CPU procesu, který čekal ve frontě nejdélě
- Krátké procesy dlouho čekají

Víceúlohová FIFO

- Nepreemptivní
- Spustí se proces, u kterého se očekává nejkratší doba provádění
- Krátké procesy mají přednost
- Hrozí vyhladovění dlouhodobých procesů

Podle odhadu doby běhu úlohy

- Preemptivní
- Spustí se vždy proces s nejkratší očekávanou dobou do dokončení
- Minimalizuje obrat

d) Popis plánovacích algoritmů: moderní a specifické 6

Round-Robin

- Preemptivní, založené na časovači
- Každý proces odstane časové kvantum na CPU
- Přepínání je provádění při vypršení časového kvanta
- Je nutné volit optimální délku časového kvanta

Prioritní

- Dává přednost procesu s vyšší prioritou
- Obvykle více fronta pro připravené procesy
- Nízká priorita může mít za následek vyhladovění procesu

Uživatelsky férové

- Zaručuje každému uživateli stejné podmínky
- Příklad:
    - Dva uživatelé (A – 9 procesů, B – 1 proces)
    - A dostane 50% času na CPU, B 50%

Loteriové plánování

- Každý proces dostane tiket a periodicky se losuje
- Výherní proces získává čas na CPU
- Proces může mít více tiketů
- Snadná implementace

11) Požadavky na plánování v systémech reálného času. Možnosti plánování vláken na víceprocesorových systémech (SMP). Časové a periodické plánování úloh uživatelem – příkazy `at` a `crontab`

a) Kritéria pro plánování v systémech reálného času, plánovatelnost 4

- Rychlý FS
- Rychlá komunikace procesů
- Rychlé přepínání kontextu
- Preemptivní plánování procesů

b) Možné optimalizace plánování vláken na systémech SMP 4× 2

Sdílení zátěže – zátěž se rozděluje mezi procesory náhodně, zajišťuje se, aby žádný procesor nezůstal nevyužitý

Skupinové plánování – související vlákna jsou plánována tak, aby běžela na různých procesorech současně, vzájemná synchronizace

Pevné přiřazení procesoru – při plánování jsou všem vláknům aplikace napevno přiřazeny procesory

Dynamické plánování – počet vláken se může během provádění měnit

c) Příkazy pro nastavení spouštění úloh v daném čase a démony, které to obstarávají. Orientačně možnosti konfigurace 3

- Příkaz at
  - Provede v unixových systémech daný příkaz pouze jednou v určitý čas
  - Pro naplánování spouštení programů
  - Umožňuje naplánovat více sekvenčních úloh
- Příkaz crontab
  - Provede příkaz opakovaně
  - Každý uživatel může mít svůj crontab

12) Požadavky na paměť, alokace, adresování, pevné (statické) a proměnné (dynamické) dělení paměti (fixed partitioning, variable partitioning), fragmentace paměti, typy fragmentace, umisťovací algoritmy

a) Požadavky na operační paměť (na její vlastnosti) 3

- Relokace paměti procesu
  - Přemístění paměti procesu na jiné místo
- Ochrana paměti
  - Procesy nesmí přímo přistupovat k paměti jádra a ostatních procesů
- Sdílení paměti mezi procesy
  - Nutné pro efektivní provádění procesů, soubory nebo jiné mechanismy výměny mohou být pomalé
- Logická a fyzická organizace
  - Modulární psaní programů
  - Dané moduly vyžadují různý stupeň ochrany paměti
  - Moduly lze sdílet

b) Pevné dělení paměti, nevýhody, umisťování procesů 4

- Paměť je rozdělena do oblastí s pevnými hranicemi
- Všechny oblasti mají stejnou velikost
  - Malý proces zabere celou oblast -> neefektivní využití paměti, vnitřní fragmentace
  - Proces je umístěn do libovolné volné oblasti
- Oblasti mají různé velikosti
  - Redukce vnitřní fragmentace, ale ne její odstranění
  - Procesy jsou řazeny do front (fronta pro každou velikost oblasti)
  - Pro každý proces se pak vybere co nejvhodnější oblast (aby se minimalizovala fragmentace – nejmenší oblast, kam se vejde)
  - Druhá možnost: jen jedna fronta, procesu je přiřazena nejmenší volná oblast do které se vejde, je rychlejší ale je větší vnitřní fragmentace

c) Dynamické dělení paměti, nevýhody 4

- Proměnný počet oblastí i jejich velikost
- Proces dostane tolik paměti, kolik potřebuje -> odstranění vnitřní fragmentace
- Po ukončení procesu však vznikají v paměti mezery
  - Do mezery je pak umístěn proces, který se tam ještě vejde, ale zůstává tam daleko menší mezera
  - Vzniká vnější fragmentace – řešení: defragmentace, „setřesením“

d) Umisťovací algoritmy pro dynamické dělení paměti (s popisem) 4

Best-fit

- Nejlépe padnoucí oblast
- Vybere stejnou nebo nejmenší volnou oblast do které se proces vejde
- Nejméně vykonná metoda
- Nejmenší možná fragmentace

First-fit

- První padnoucí
- Vybere první volnou oblast do které se proces vejde
- Rychlejší
- Prohledávání zpomaluje výskyt obsazených pamětí

Next-fit

- Následující padnoucí oblast
- Paměť je prohledávána vždy od stejného místa, kde se naposledy umisťovalo
- Vybere první volnou oblast, do které se proces vejde
- Nejrychlejší metoda

Worst-fit

- Největší padnoucí
- Vybírá oblast stejně velkou jako proces nebo největší volnou
- Nejhorší využití paměti

13) Problém nedostatku operační paměti, odkládání obsahu paměti na disk, virtuální paměť, motivace k zavedení virtuální paměti, výhody virtualizace paměti, princip lokality odkazů, thrashing

a) Možná řešení nedostatku RAM (s vysvětlením principu) 2 + 2

Swapping (odkládání):

- Odkládání procesu z operační paměti na sekundární pamět SWAP
- Využití levné sekundární paměti (disku)

Overlaying (překrývání):

- Různé moduly programu nejsou vyžadovány současně
- Proto využívají stejnou fyzickou oblast paměti

b) Virtualizace paměti: definice a HW podpora 3

Definice:

- Procesor je obvykle schopen adresovat větší množství paměti, než je skutečně instalováno
- Logický adresový prostor zahrnuje: skutečnou paměť RAM, část sekundární paměti SWAP

HW podpora:

- CPU musí umět pracovat s virtuální adresou
- MMU – převádí virtuální adresu na skutečnou
- Není-li adresová část v RAM, je třeba generovat přerušení a předat řízení OS

c) Virtualizace paměti: motivace a důsledky zavedení, princip fungování 2

Přistoupí-li proces k paměťovému místu (data, kód), které není ve fyzické operační paměti, generuje se přerušení. Operační systém proces zařadí mezi blokované, dokud nenahraje příslušnou část jeho adresového prostoru zpět do fyzické operační paměti. Po dokončení čtení je generováno přerušení a proces je následně zařazen opět mezi připravené.

d) Virtualizace paměti: pojmy (RS, S) 1

Resident-set = část adresového prostoru, která je v RAM

SWAP = sekundární paměť, kam lze dočasně odložit nepoužívané části adresového prostoru

e) Princip lokality odkazů 2

Proces má tendenci přistupovat k okolí svého adrsového prostoru, kam přistupoval nedávno, proto lze odvodit, kterou část paměti bude proces v nejbližší budoucnosti potřebovat. Umožňuje efektivněji využít virtuální paměť.

f) Thrashing – definice a příčiny 3

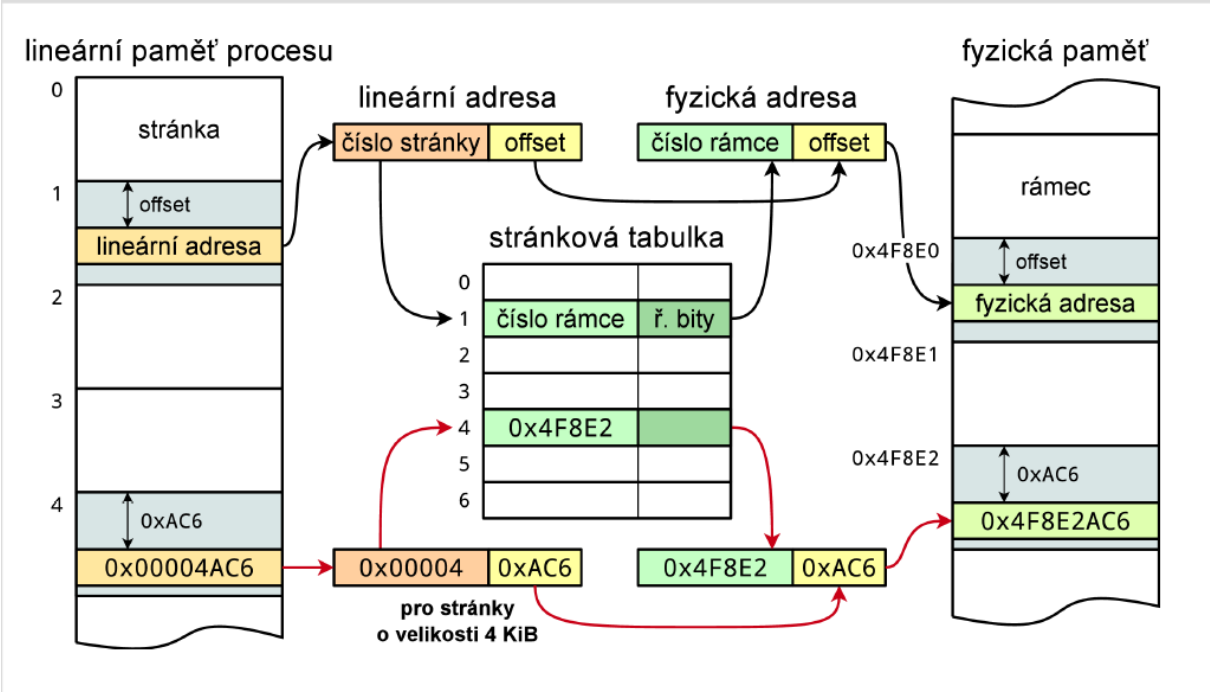
- Vzniká při špatné organizaci odkládání částí paměti na disk
- Část paměti je odložena těsně před tím, než bude potřeba
- Může být následkem nedostatečného dodržení principu lokality odkazů
    - Neoptimalizovaného překladu kompilátorem
    - Neefektivního programování

14) Stránkování paměti, převod adresy, vlastnosti stránkování, sdílení stránek, volba velikosti stránky, řešení problému rozsáhlých stránkových tabulek, TLB

a) Princip, vlastnosti (spojitost, fragmentace, sdílení) 5

- RAM je rozdělena na oblasti malé velikosti (rámce)
- Logický adresový prostor procesu se rozdělní na stejně velké části jako RAM – stránky
- OS pak udržuje pro každý proces tabulku přiřazení stránek k rámcům
- Logická adresa se skládá z čísla stránky a offsetu
- Vlastnosti:
  - Skutečné umístění stránek je nespojité
  - Logický adresový prostor je lineární
  -

b) Stránkové tabulky, převod adresy 4



c) Volba velikosti stránky a důsledky 3

Menší stránky

- Menší vnitřní fragmentace
- Více stránek na proces – větší stránková tabulka
- Více rámců v RAM = lepší hospodaření s RAM

Větší stránky

- Větší vnitřní fragmentace
- Menší stránkovací tabulka
- Méně rámců v RAM = horší hospodaření s RAM
- Sekundární paměť je efektivnější při přenosu dat po větších blocích

d) Řešení rozsáhlosti stránkových tabulek, TLB 3

Rozsah stránkových tabulek může být velký, ale proces většinu stránek nepoužije. Využívá se virtuální adresa o velikost 32 bitů, velikost stránky je pak 4 KiB. 64bitová adresa je nerealizovatelná! TLB (Translation Lookaside Buffer) = speciální cache pro položky stránkové tabulky, obsahuje několik posledních použitých položek stránkové tabulky.

15) Řídicí bity ve stránkových tabulkách, strategie zavádění, umisťování, nahrazování a uklízení (čištění) stránek paměti, vliv velikosti resident-set na běh procesů

a) Důležité řídicí bity ve stránkové tabulce a jejich význam 3

b) Strategie zavádění stránek: účel, algoritmy 3

- Určuje, která stránka se má zavést do RAM
- Algoritmy:
  - Demand pagging – zavedení stránek jen když jsou třeba
  - Lookahead pagging – zavedení stránek předem

c) Strategie umisťování dtránek: účel, algoritmy 1

- Určuje, do které části RAM se stránka zavede
- Typicky se tím OS nezabývá
  - Vybírá obvykle první volný rámec
  - Skutečnou adresu používá pouze HW

d) Strategie nahrazování stránek: účel, algoritmy 4

- Určuje, které stránky se z RAM odloží na disk, je-li potřeba načíst další stránky
- Ideální strategie:
  - Odstranění stránky, která bude nejdelší dobu nevyužitá
  - Nelze určit, jen se odhaduje
- Zamykání rámců
  - Používá se pro stránky, které nesmí být z RAM odloženy
  - Pro jádro OS, V/V buffery, klíčové řídící struktury OS

e) Strategie uklízení (čištění) stránek: účel, algoritmy 3

- Určuje, kdy se (modifikované stránky) uloží na disk
- Algoritmy:
  - Demand cleaning – ukládání stránky z rámce vybraného pro nahrazení
  - Precleaning – periodické ukládání stránek v dávce

f) Volba velikosti resident-set 1

- Pevná alokace
  - Procesu je při nahrávání vyhrazen pevný počet rámců RAM
- Proměnná alokace
  - Počet rámců procesu se může průběžně měnit

16) Požadavky na OS pro práci v reálném čase, rozdělení RT OS, pojmy latence a odezva (na úrovni přerušení); vestavěné systémy, OS pro ně a typické požadavky na ně

a) Definice systémů RT (pracujících v reálném čase) a jejich správná funkce 1

- Operační systém pracující v reálném čase
- Deterministické chování
- Krátká doba odezvy
- Vysoká spolehlivost

b) Rozdělení systémů RT dle dodržování termínů (s popisem) 3

- Hard real-time
  - Absolutní časové limity, při jejichž překročení je odezva zcela bezcenná a systém selhává
- Soft real-time
  - Časové limity jsou přibližné, jejichž překročení sníží pouze užitečnost systém
- Firm real-time
  - Odezva po časovém limitu je bezcenná, nicméně může snést několik málo zmeškání aniž by selhal

c) Charakteristické vlastnosti RTOS 4

- Rychlé přepínání kontextu
- Prioritní plánování založené na preepci procesů i jádra
- Malé rozměry
- Rychlý souborový systém
- Podpora speciálních systémových služeb
- Spolehlivost
- Multitasking

d) Příklady RTOS (alespoň dva) 1

- QNX, VxWorks, RTLinux

e) Definice pojmů latence a odezva (na úrovni přerušovacího systému) 2

Doba odezvy = čas, ve kterém musí systém přiměřeně reagovat na událost (za jak dloiho OS reaguje na požadavek přerušení)

Latence = doba mezi okamžikem příchodu požadavku na přerušení a dobou, kdy se začně provádět odpovídající obslužný program.

f) Definice vestavěných systémů 1

- Počítačové systémy, které jsou součástí jiných systémů
- Obvykle představují jejich řídící složku
- Nebo tvoří jejich podsystémy
- Obvykle jsou schopni pracovat v reálném čase

g) Typické vlastnosti vestavěných systémů (alespoň šest) 3

- Malá spotřeba
- Odolnost
- Malé rozměry a váha
- Reaktivita
- Funkce v reálném čase
- Spolehlivost a bezpečnost
- Cenová citlivost

17) Víceprocesorové systémy, rozdělení dle vazby a dle symetrie, granulovatelnost úlohy, souvislost s vazbou (víceprocesorových systémů) a stupně paralelismu, distribuované (rozptýlené, clusterové) OS

a) Kategorie počítačových systémů z hlediska paralelizace zpracování dat 4

- SISD (Single instruction single data) – jeden procesor zpracovává jednou sadou instrukcí jednu množinu dat
- SIMD (Single instruction multi data) – více procesorů zpracovává jednou sadou instrukcí více množin dat
- MISD (Multi instruction single data) – více procesorů zpracovává více sadami instrukcí jednu množinu dat
- MIMD (Multi instruction multi data) – více procesorů zpracovává více sadami instrukcí více množin dat

b) Rozdělení víceprocesorových systémů dle vazby 2

- S volnou vazbou
  - Každý procesor má vlastní RAM a V/V subsystém
- S pevnou vazbou
  - Procesory sdílejí RAM a jsou řízeny jedním OS

c) Rozdělení víceprocesorových systémů dle symetrie 2

- Symetrické
  - Shodné procesory
  - Jádro OS může provádět cokoliv
  - Procesy smí být prováděny jakýmkoliv procesorem
- Asymetrické
  - Procesory jsou různé – jsou funkčně specializované (V/V procesor, grafický procesor,…)
  - Systém je řízen centrálním procesorem

d) Granularita úlohy: účel, jak se pozná, že úloha je/není granulovatelná 4

- Každou úlohu můžeme rozčlenit na úseky, které lze provést samostatně, které pak lze na MP systému provádět paralelně na různých procesorech
- Pokud jeden úsek potřebuje pro svou činnost výsledky druhého, musí na něj čekat – nutnost komunikace synchronizace
- Různé typy úloh se liší velikostí a počtem takových úseků

e) Granularita a vhodné stupně vazby 3

- Hrubě granulovatelná úloha
  - Méně úseků
  - Vhodné pro kooperující procesy
- Jemněji granulovatelná úloha
  - Kratší úseků a více
  - Požadavky na komunikaci a synchronizace jsou častější
  - Vhodné pro vlákna, pokud běží na oddělených procesorech
- Čím volnější vazba, tím větší časové ztráty přináší komunikace a synchronizace

18) Soutěžení procesů (o prostředky), obecné problémy souběhu, vzájemné vylučování, kritická sekce, předpoklady pro řešení KS, požadované vlastnosti řešení KS, typy řešení

a) Definice kritické sekce v programu 4

- Jedná se o část kódu procesu, který manipuluje se sdílenými prostředky a současná manipulace by vedla k problému.
- Provádění KS musí být vzájemně výlučné – v každém okamžiku smí být v KS jen jediný proces a proces smí žádat o povolení vstoupit do KS
    - Vstupní sekce – implementace povolení vstupu do KS
    - Kritická sekce – manipulace se sdílenými prostředky
    - Výstupní sekce – implementace uvolnění KS pro jiné procesy
    - Zbytková sekce – zbytek kódu procesu

b) Předpoklady pro řešení přístupu do kritické sekce. (Co se předpokládá, že platí, a co se nesmí předpokládat) 4

- Nenulová rychlost procesů
- Žádné předpoklady o relativní rychlosti procesů
- Pro MP systémy: paměťové místo smí být přístupno v jeden okamžik pouze jedinému procesoru
- Žádné předpoklady o prokládaném provádění
- Specifikace vstupní a výstupní sekce

c) Požadované vlastnosti řešení přístupu do kritické sekce (s vysvětlením) 3

- Vzájemné vylučování
    - Vždy jen jeden proces v KS
- Pokrok v přidělování
    - Na rozhodování o vstupu do volné KS se mohou podílet pouze procesy, které nejsou v ZS kódu a rozhodnutí musí padnout v konečném čase
- Omezené čekání
    - Mezi požadavkem na vstup do KS a vyhověním do KS smí vstoupit pouze omezený počet jiných procesů

d) Typy řešení přístupu do kritické sekce (s příklady) 4

Softwarové řešení – algoritmy pro vstupní a výstupní sekci

Hardwarové řešení – instrukce procesoru

Řešení operačního systému – semafor

Řešení programovacího jazyka – konkurenční/souběžné programování

19) Řízení přístupu do kritické sekce pomocí SW metod, příklady nevhodných (obecně nefunkčních) SW řešení, SW algoritmy, vlastnosti (nedostatky) SW metod

a) SW algoritmus pro KS s proměnnou locked, důvod nefunkčnosti 1 + 1

- Sdílená proměnné locked (udává obsazenost KS), inicializována na 0
- Proces Pi čeká, dokud je KS obsazena (locked = 1)
- Jak je KS volná (locked = 0), tak se locked nastaví na 1
- Problém:
  - Dva procesy, či více, současně zjistí volnou KS, všechny procesy nastaví obsazeno a vstoupí do KS
  - Není splněn požadavek vzájemného vylučování

b) SW algoritmus pro KS s proměnnou turn, důvod nefunkčnosti 1 + 1

- Sdílená proměnná turn, inicializovaná na 0 nebo 1
- Udává, který proces smí vstoupit do KS
- Pokud proces s PID = i chce vstoupit do KS, musí být turn = i
- Proces aktivně čeká dokud turn nemá požadovanou hodnotu
- Není splněn požadavek pokroku

c) SW algoritmus pro KS s proměnnými flag[i], důvod nefunkčnosti 1 + 1

- Sdílená proměnná flag[i], pro každý proces Pi => flag[i]
- Při vstupu do KS se proces Pi ptá zda flag[i] = true
- Ve výstupní sekci se pak flag pro další proces nastaví na true => umožněn vstup do KS dalšímu procesu
- Pokud však chtějí procesy vstoupit do KS znovu po vykonání této sekvence již mají všechny procesy nastaveno true a hrozí deadlock
- Požadavek pokroku není splněn

d) Funkční SW algoritmy pro řízení přístupu do KS, krátká charakteristika 2 + 2

- Petersonův algoritmus
  -
- Leslie Lamport's bakery algorithm

e) Specifické nevýhody SW algoritmů pro řízení přístupu do KS (včetně vysvětlení, bez všeobecných nevýhod všech typů algoritmů) 5

- Neodolnost vůči chybám v KS
  - Pokud proces havaruje v KS, tak to stále vypadá že pracuje v KS a ostatní procesy se do KS nedostanou
- Aktivní čekání
  - Procesy čekající na vstup do KS spotřebovávají zbytečně a neproduktivně čas procesoru
  - Vhodné, pokud je KS krátká
-

20) Řízení přístupu do kritické sekce pomocí HW metod, výchozí předpoklady pro HW řešení, algoritmy využívající HW instrukce, vlastnosti (nedostatky) HW metod

a) Hardwarové předpoklady pro řízení přístupu do KS 3

- Procesy jsou prováděny v procesoru kontinuálně, dokud nevyvolají službu OS nebo nejsou přerušeny
- K přerušení procesu může dojít pouze na hranicích instrukcí
  - Mezi dokončením jedné instrukce a zahájením další
- Přístup k paměti je výlučný
  - Důležité pro SMP systémy

b) HW podpora pro řízení přístupu do KS, popis řešení 2 + 3

- Zákaz přerušení
  - Proces běží, dokud nezavolá službu OS nebo není přerušen
  - Plánovač využívá přerušení
  - Zákaz přerušení způsobí, že proces nemůže být přerušen, tudíž žádný jiný proces nemůže vstoupit do KS
  - Vzájemné vylučování zajištěno pouze na jednoprocesorových systémech
  - Nedostatky: Zvyšuje latenci systému, V KS se nesmí volat služba jádra, Obecně nevhodné řešení
- Speciální instrukce procesoru test-and-set
  - Jedna instrukce procesoru přečte příznak a zároveň ho nastaví
  - Byl-li příznak již nastaven – KS obsazena
  - Nebyl nastaven proces smí do KS
  - Lze využít pro více různých KS
  - Nevýhody: aktivní čekání, vyhladovění, deadlock
- Speciální instrukce procesoru xchg
  - Pro intel x86 architekturu
  - Využívá sdílenou proměnnou locked inicializovanou na 0
  - Funguje obdobně jako test-and-set

c) Vlastností (nedostatky) HW řešení 4 + 3

Nedostatky:

- Aktivní čekání

Vlastnosti: krátká vstupní a výstupní sekce

21) Nástroj OS: semafor, jeho popis včetně systémových volání, použití semaforu pro řízení přístupu do KS a pro synchronizaci, problém obědvajících filozofů

a) Popis nástroje OS: semafor 2

Semafor je prostředek OS, jedná se o synchronizační nástroj. Nevyžaduje aktivní čekání, dokud je KS obsazena čekající proces je blokován a zařazen do fronty čekajících na uvolnění KS. Obecně se jedná o datovou strukturu obsahující: celočíselný čítač, frontu procesů a operace init, wait a signal.

b) Popis systémových volání semaforu 1 + 2 + 2

- init – inicializuje čítač na požadovanou hodnotu
- wait – snižuje hodnotu čítače o 1 a pokud je hodnota čítače záporná, vkládá proces do fronty procesů
- signal – zvyšuje hodnotu čítače o 1 a pokud je ve frontě proces, tak je proces uvolněn a vykonán

c) Popis řešení přístupu do KS používajícího semafor 3

- Sdílený semafor se inicializuje na počet proměnných smějících naráz vstoupit do KS (typicky 1) – sem_init(&sem,1);
- Ve vstupní sekci se pak volá operace wait – sem_wait(&sem);
- Ve výstupní sekci operace signal – sem_post(&sem);

d) Popis řešení synchronizace vláken pomocí semaforu 3

- Sdílený semafor je inicializován na 0 – sem_init(&sem,0);
- Proces, který má čekat poté volá operaci wait – sem_wait(&sem) – čítač se nastaví na -1 a proces je blokován;
- Proces, na který se čeká poté volá operaci sem_post(&sem) – čítač se nastaví na 0 a blokovaný proces je aktivován;

e) Definice problému obědvajících filozofů a jeho řešení pomocí semaforů 2

22) Nástroj OS: předávání zpráv, popis systémových volání a možností blokování, použití fronty zpráv pro řízení přístupu do KS a pro synchronizaci, problém svázaných producentů a konzumentů

a) Popis nástroje OS: předávání zpráv 3

- Komunikační prostředek OS pro procesy
- Nutné vzájemné vylučování (dochází k výměně informací, zajišťuje OS)
- Systémová volání: send(cíl, zpráva), recieve(zdroj, zpráva)

b) Popis typické implementace (ne)blokování systémových volání pro předávání zpráv 3

- Typická implementace blokování send a recieve pro fronty zpráv s omezenou velikostí:
  - Neblokující send – blokuje pouze při zaplnění fronty zpráv
  - Blokující recieve – příjemce je blokován, není-li dostupná žádná zpráva
  - Neblokující varianty lze nastavit parametrem, volání pak místo blokování vrací chybu

c) Popis řešení KS používajícího frontu zpráv 3

- Sdílená fronta se inicializuje zasláním jedné zprávy
- Vstupní sekce pak volá blokující recieve
- Výstupní sekce neblokující send
- První proces, který provede recieve se dostane do KS, ostatní jsou blokovány, dokud jiný proces neprovede send.

d) Popis řešení synchronizace vláken pomocí fronty zpráv 3

- Fronta zpráv je vyprázdněna
- Proces, který čeká na vykonání jiného volá blokující recieve
- Proces, na který se čeká pak volá neblokující send
- Provede se nejdříve send, tak recieve neblokuje

e) Definice problému svázaných producentů a konzumentů a jeho řešení pomocí fronty zpráv 3

23) Nástroj programovacích jazyků: koncept monitoru, problém producentů a konzumentů a jeho řešení pomocí monitoru

a) Popis konceptu monitoru: jeho účel, struktura a základní vlastnosti 2 + 2

- SW modul (něco jako třída)
  - Obsahuje lokání proměnné – sdílená data, funkce zpřístupňující lokální data a inicializační část
- Ve funkci monitoru smí být v jednu chvíli pouze jedno jediné vlákno
- Zajišťuje vzájemné vylučování
- Synchronizace lze zajistit podmínkovými proměnnými

b) Popis řešení KS pomocí monitoru 2

- Kód kritické sekce je umístěn ve funkci monitoru
- Pro vykonání kritické sekce se pak volá tato funkce
- Monitor obsahuje sdílené prostředky, se kterými se manipuluje v KS
- Tím, že ve funkci monitoru smí být pouze jedno vlákno je zaručeno vzájemné vylučování

c) Popis řešení synchronizace vláken pomocí monitoru 3

- Řešeno pomocí podmínkových proměnných – synchronizační nástroj monitoru
- Podmínkové proměnné jsou lokální v monitoru a dostupné pouze pomocí funkce monitoru
- Lze je měnit dvěma funkcemi:
  - cwait(cv) – blokuje vlákno, dokud není zavoláno csignal(cv)
  - csignal(cv) – obnovuje provádění vlákna blokovaného proměnnou cv
    - je-li takových vláken více, vybere se jedno z nich, není-li žádné nestane se nic

d) Definice problému svázaných producentů a konzumentů a jeho řešení pomocí monitoru, podmínky použité u synchronizace a zdůvodnění jejich použití 6

24) Nástroje knihovny posixových vláken: mutex, bariéra, podmínková proměnná

a) Popis mutexu, účel a vlastnosti, popis funkcí 3

- Binární semafor
- Místo čítače obsahuje logickou proměnnou inicializovanou na false
- **Mut**ual **ex**clusion – Mutex
- Využití pro vzájemné vylučování
- Operace wait a signal se označují jako lock a unlock
- Lock – nastaví logickou proměnnou na false – blokující volání
  - Zamkne mutex, neblokující pouze pro jeden proces
  - Umístí proces do fronty procesů
- Unlock – nastaví logickou proměnnou na true – neblokující volání
  - Odemyká mutex, je-li fronta procesů prázdná

b) Způsob použití mutexu 2

- Mutex zaručuje vzájemné vylučování, využívá neaktivní čekání
- Vlákno je při pokusu zamknout již uzamčený mutex blokováno
- Využití hlavně pro řešení přístupu do KS
- Ve vstupní sekci: lock
- Ve výstupní sekci: unlock

c) Bariéra: účel, použití a popis funkcí (včetně návratové hodnoty funkce čekání) 1 + 4

- Objekt sloužící pro synchronizaci vláken
- Umožňuje vláknu/ům neaktivně čekat na ostatní vlákna
- Bariéra se inicializuje na počet vláken, kolik je potřeba synchronizovat
- Jakmile k bariéře dospěje daný počet vláken bariéra propustí všechny vlákna – paralelní běh
- Operace wait:
  - Operaci wait je volána tolikrát, na kolik je inicializován čítač
  - Vrací 0 při úspěšném volání, nenulovou hodnotu pro chybu
  - Vrací PTHREAD_BARRIER_SERIAL_THREAD (speciální hodnota), kterou to vrátí jen pro jedno nespecifikované vlákno – bariéra je pak připravena pro použitá na daný počet vláken

d) Podmínková proměnná: účel, způsob použití a vlastnosti 1 + 4

- Slouží pro synchronizaci vláken
- Umožňuje vláknu neaktivně čekat na událost
- Nezaručuje exkluzivitu přístupu (je třeba k ní přistupovat pomocí mutexu)
- Na událost může čekat i několik vláken
- Událost, na kterou se čeká je oznámena některým vláknem pomocí signálu
  - Signál probouzí jedno vlákno
  - Lze však poslat i vícesměrový signál a probudit všechna vlákna
  - Nečeká-li žádné vlákno je signál ztracen

25) Stav uváznutí (deadlock) a vyhladovění (definice a rozdíl), nutné podmínky pro vznik stavu uváznutí, předcházení a řešení problému stavu uváznutí, algoritmus bankéře

a) Definice stavu uváznutí (deadlock) 2

Skupina procesů je ve stavu uváznutí, když každý proces ve skupině čeká na událost, kterou může vyvolat je proces, který je v dané skupině. Procesy budou nekonečně dlouho čekat, protože událost lze vyvolat jen procesem, který taktéž čeká na danou událost.

b) Definice stavu vyhladovění (starvation), příklad 1

Nekonečné čekání procesu na přidělení prostředků, proces nemá žádný pokrok. Proces v soupeření o prostředek neustále prohrává – např. nízká priorita při soutěži o čas na CPU.

c) Vysvětlení podmínek pro vznik stavu uváznutí 4

- Vzájemné vylučování
  - Prostředky lze vlastnit pouze jedním procesem
- Alokace a čekání
  - Proces vlastnící prostředky může požadovat další
- Neodnímatelné prostředky
  - OS je nemůže odejmout, musí být explicitně uvolněny vlastnícím prostředkem
- Cyklické čekání
  - Řetěz vzájemně čekajících procesů uzavírá cyklus

d) Přístupy k řešení odstranění vzniklého stavu uváznutí 3

- Ignorování problému
- Detekce a obnovení
  - Obnova do stavu bez uváznutí
  - Násilné odebrání prostředků
  - Zabití procesu
- Vyloučení možnosti vzniku (prevence)
  - Negování alespoň jedné z předešlých podmínek

e) Způsoby prevence vzniku stavu uváznutí 3

- Negace vzájemného vylučování – spooling
- Negace alokace a čekání – zajištění alokace všech potřebných prostředků naráz
- Negace neodnímatelných prostředků – možnost nenásilného odebrání prostředku
- Negace cyklického čekání – nedovolit alokaci prostředku, pokud by vznikl cyklus
- Dovolení pouze bezpečných stavů

f) Popis bankéřova algoritmu aplikovaného na procesy (s popisem stavů) 2

- Řešení situace bankéře při jednání s klienty, kterým poskytuje půjčku – pokud požadavek klienta vede k nebezpečnému stavu je tento požadavek odmítnut.
- Výchozí předpoklady: pevný počet prostředků, každý proces deklaruje své maximální požadavky, postupná alokace prostředků
- Prostředek je na požadavek přidělen jen tehdy vede-li situace do bezpečného stavu

26) IPC: komunikace procesů a vláken, možné prostředky komunikace

a) Možné prostředky pro komunikaci procesů / vláken 5

- **Soubor, databáze** – pomalé, náhodný přístup, současný přístup je třeba řídit
- **Roura** – proudový přístup (FIFO), jednosměrná komunikace
- **Socket** – proudový přístup (FIFO), obousměrná komunikace
- **Fronta zpráv** – exkluzivní přístup, jednosměrná komunikace
- **Sdílená paměť** – nejrychlejší, náhodný přístup, současný přístup nutno řídit

b) Krátká charakteristika prostředků pro komunikaci procesů / vláken 5

c) Popis funkcí pro sokety (alespoň pěti typů) 5

27) Dělení disku na oddíly, zavaděč OS, důvody dělení, MBR, GPT, swap

a) Popis důvodů pro rozdělení disku na oddíly 4

- Více operačních systémů
- Oddělení dat od OS
- Možnost vytvoření oddílu SWAP
- Při poruše HW se chyba může týkat jen jednoho oddílu, ostatní tak může zůstat neporušen

b) Popis MBR a způsob dělení disku na oddíly 3

- **MBR** (Master Boot Record)
  - První sektor disku
  - Tabulka rozdělení disku na oddíly
    - Primární – lze z něj zavést OS sekundárním zavaděčem
    - Rozšířený – dělí se na logické oddíly
  - Zavaděč systému může být univerzální – skok na sekundární zavaděč na aktivním primárním oddíle

c) Popis typů oddílů (pro dělení MBR) 3

- **Primární** – oddíl, ze kterého lze zavést OS
- **Rozšířený** – oddíl, který obsahuje seznam logické oddíly
- **Logický** – v podstatě primární oddíly vytvořené v rozšířeném oddílu, můžeme na ně ukládat data, jen neslouží pro zavedení OS

d) Charakterizace GPT 3

- **GUID Partition Table**
  - Standard UEFI
  - Náhrada z MBR
  - Má zálohu na konci disku
  - Až 128 primárních oddílů
  - Možno využít disk větší než 2 TiB

e) Vysvětlení rozdílů mezi odkládacím prostorem (swap) na samostatném diskovém oddíle a v souboru 2

- **Swap na diskovém oddíle**
  - Je rychlejší
  - Hůř se mění velikost oddílu
- **Swap v souboru**
  - Pomalejší
  - Velikost souboru se může měnit dle potřeby

28) Souborový systém, metadata, speciální soubory

a) Popis obecné struktury souborových systémů a jejich metadat 3

- Umožňuje uživatelům a procesům přístup k souborům na paměťovém médiu
- Vytváří logickou strukturu souborů
- Eviduje metadata souborů
- Organizuje uspořádání souborů na médiu
- Sjednocuje přístup k datům různého typu

b) Metadata souborů vyjma oprávnění (alespoň šest) 3

Typ, velikost, čas změny, umístění, jméno, vlastník, oprávnění

c) Možná oprávnění na soubor (alespoň šest) 3

Provádění, čtení, zápis, mazání, vytváření, změna práv, znalost existence, přidávání

d) Popis zvláštních typů souborů (alespoň čtyř) 2

- **Odkazy** – obsahuje jméno cíle
- **Adresáře** – jména souborů s čísly i-uzlů
- **Pojmenovaná roura** – jednosměrný komunikační nástroj pro dva procesy, co bylo přečteno se odstraní
- **Pojmenovaný socket** – adresa = jméno souboru
- **Zařízení** – jednotný přístup k HW (blokový – náhodný přístup – disky; znakový – proudový přístup – terminál, tiskárna, skener,…)

e) Charakteristika typů odkazů a jejich reprezentace na souborovém systému 2 + 2

- **Pevné**
  - Více jmen pro jeden soubor na stejném FS
  - Data i metadata jsou na médiu pouze jednou
  - Ke smazání dojde při smazání posledního odkazu
- **Symbolické**
  - Textová záměna za jiné jméno (absolutní či relativní)
  - Změna jména je pro procesy transparentní

29) Konzistence metadat souborových systémů: příčiny vzniku nekonzistencí, metody zachování konzistence, vlastnosti metod, příklady souborových systémů

a) Popis příčiny vzniku nekonzistence metadat souborového systému 3

Zápis dat do souboru znamená provedení několik operací na různých místech na médiu:

- Metadata souborového systému – alokace nových bloků, aktualizace seznamu volných bloků
- Datová část souborového systému – zápis nových dat souborů
- Metadata souboru – aktualizace přidělených bloků, velikosti, času, změny apod.

Tyto změny probíhají postupně, čímž nutně vznikají stavy nekonzistence. Při pádu systému je třeba provést kontrolu stavu souborových systémů a případné nekonzistence odstranit.

b) Popis principu žurnálování a činnost obnovy konzistence po pádu 5

Pokud dojde k pádu systému, je potřeba při startu OS zkontrolovat integritu FS (prověřit zda jsou metadata konzistentní). Na velký FS tahle operace může trvat dost dlouho. Z tohoto důvodu se zavedla podpora žurnálování.
Pro zachování konzistence lze zavést transakční log, zvaný žurnál. Jedná se o kruhový buffer, do kterého se zapisují prováděné změny metadat.
Až po potvrzení zápisu do žurnálu se provede patřičná změna FS. Při startu systému po pádu pak není třeba kontrolovat konzistenci celého celku, ale stačí zkontrolovat místa, kde došlo ke změnám těsně před pádem systému. Tyto míst se zjistí právě z žurnálu.

c) Popis principu metody copy-on-write a činnost obnovy konzistence po pádu 4

- Kopírování bloku při změně
- Nekonzistence vzniká při změně informací – zápisu
- Souborový systém nikdy nemodifikuje bloky na místě
  - Při přepisu se modifikuje kopie bloku
  - Stejně jako metadata
  - Po dokončení zápisu se atomicky provede zneplatnění původních dat/metadat a potvrdí se platnost nových
  - FS je tedy vždy konzistentní
  - Nevýhoda: větší datová fragmentace

d) Příklady souborových systémů ve vztahu k metodě zachování konzistence, alespoň dva různé pro každou metodu (nikoliv různé verze téhož), pro žurnálování zvlášť po dvou příkladech pro dvojí způsob 3

- Souborové systémy bez konzistence po pádu
  - FAT, exFAT, ext2, HFS, HPFS
- Žurnálovací FS
  - Úplné žurnálování: ext3, ext4, ReiserFS, UDF
  - Jen metadata: NTFS, JFS, XFS, VMFS
- Souborové systémy s copy-on-write
  - ZFS, BtrFS, Nilfs, ReFS

30) Typy úložišť, RAID, způsob alokace dat souborů

a) Popis typů úložišť (DAS, NAS, SAN) 3

- DAS (Directly Attached Storage)
  - Lokální paměťové úložiště
  - SCSI, SATA, IDE
- NAS (Network Attached Storage)
  - Souborový systém zpřístupněný síťovým protokolem
  - NFS, CIFS, AFS
- SAN (Storage Area Network)
  - Blokové zařízení zpřístupněné síťovým protokolem
  - iSCSI, Fibre Channel

b) Charakteristika RAID 0 (způsob zapojení a ukládání dat, odolnost, rychlost R/W) 3

- Stripping – prokládané ukládání
- Rychlejší
- Výpadek disku znamená ztrátu všeho

c) Charakteristika RAID 1 (způsob zapojení a ukládání dat, odolnost, rychlost R/W) 3

- Mirroring – zrcadlené ukládání
- Rychlejší, redundantní, kapacita jen jednoho disku

d) Charakteristika RAID 5 (způsob zapojení a ukládání dat, odolnost, rychlost R/W) 3

- Stripping + parity – prokládané s paritou
- Rychlejší, redundantní, kapacita snížená o jeden disk

e) Charakteristika možných způsobů alokace dat pro soubory 3

- Souvislá alokace
  - Souboru jsou přiděleny po sobě jdoucí bloky
  - V metadatech: adresa prvního bloku a velikost souboru
  - Dochází k vnější fragmentaci
- Řetězená alokace
  - Alokují se jednotlivé bloky
  - Alokační tabulka obsahuje seznam bloků
  - Odstraněna vnější fragmentace
  - Sousedící bloky nemusí být u sebe
  - FS FAT
- Indexová alokace
  - Index obsahuje seznam přidělených bloků
  - Index se může dotazovat na blok s indexem
  - Bloky mohou mít různou velikost
  - Unixové FS

31) Vzdálený přístup k OS, telnet, SSH, autentizace, autorizace, zásady tvorby hesla, typy útoků na systém a prevence

a) Definice pojmů autentizace a autorizace 2

Autentizace = proces ověření identity (biometrie, průkaz, čip,…)

Autorizace = povolení provádět příslušné operace (například číst, zapisovat do souboru, spustit ho)

b) Možné metody autentizace (alespoň čtyři) 2

- Login + heslo
- Biometrie – otisk prstu, hlas, obličej
- Rfid karty nebo čipy
- Certifikáty

c) Popis principu zabezpečení přihlašování pomocí protokolu SSH 3

První připojení z daného PC:

- Dochází k ověření veřejného klíče serveru uživatelem (pomocí otisku klíče)
- Po potvrzení tohoto klíče se klíč uloží a při dalším přihlášení je kontrolováno, zda je klíč stejný
- Klient ssh se poté pokusí uživatele ověřit k tomuto účelu existuje několik metod, z nichž je nejběžnější ověření hesla nebo klíče uživatele

d) Riziko přihlašování pomocí protokolu SSH a jak mu předcházet 2 + 2

Riziko nastává při prvním přihlašování. Server s PC si musí předat klíče, které nejsou šifrované. Lze jednoduše tyto klíče padělat (útočník je vymění za své) a pak komunikace běží přes počítač útočníka a ten poslouchá (sleduje) komunikaci.

Tomuto riziku se můžeme bránit kontrolou klíče, který server má vystaven na své webové stránce.

e) Zásady tvorby hesla (čtyři nejdůležitější) 2

- Minimálně 8 znaků
- Jiné heslo pro každou službu
- Kombinace různých typů znaků (velká písmena, malá písmena, čísla, speciální symboly)
- Nemělo by obsahovat slova nebo posloupnosti znaků (neslovníková slova)

f) Popis alespoň čtyř typů útoků na systém 2

- **Login spoofing** – podvržení přihlašovací obrazovky
- **Trojský kůň** – nevinně vypadající program, který má škodlivou funkci navíc (malware)
- **Zadní vrátka** – vložená metoda přístupu navíc, například při speciálních vstupních datech je zaručen přístup
- **Logické bomby** (časované) – po určité době nebo za určitých okolností je spuštěn škodlivý kód