Státnicový okruh 4:
Informační technologie

26. května 2015

Obsah

1

John von Neumannova a harvardská architektura počítače, princip jeho činnosti. Binární logika, logické operace a funkce, logické obvody. Reprezentace čísel a znaků v paměti počítače. Osobní počítač (PC), základní deska, chipset a sběrnice (interní, externí). Procesor (CPU), vykonávání instrukcí, podprogramy a zásobník, přerušení. Paměti počítače (RAM, cache, disk, diskové pole). Přídavné karty PC, datové mechaniky a média (CD, DVD, paměťové karty), periferie.

2

Operační systém, architektura, poskytovaná rozhraní. Správa procesoru: procesy a vlákna, plánování jejich běhu, komunikace a synchronizace. Problém uváznutí, jeho detekce a metody předcházení. Správa operační pa- měti: segmentace, stránkování, virtuální paměť. Správa diskového prostoru: oddíly, souborové systémy, zajištění konzistence dat.

2.1 Operační systém, architektura, poskytovaná rozhraní

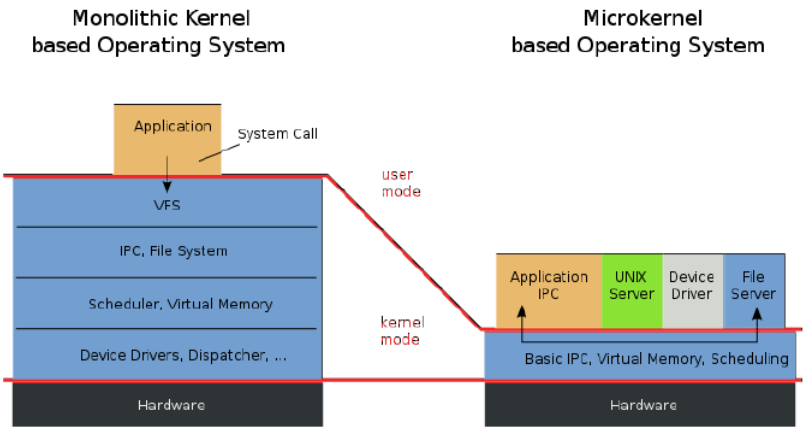
2.1.1 Operační systém

- Základní softwarové (programové) vybavení počítače
- Rozhraní mezi hardware a ostatním software (knihovny, systémové služby, aplikace)
- Rozhraní mezi uživatelem a aplikacemi
- Běží v privilegovaném režimu (jeho funkce jsou upřednostněné)

2.1.2 Architektura OS

Očekáváme od něj:

- správu a sdílení procesoru (možnost spouštět více procesů "současně")
- správu paměti (procesy jsou v paměti odděleny)
- komunikaci mezi procesy (IPC - Inter-Process Communication)
- obsluhu zařízení a organizaci dat (souborový systém, síťové rozhraní, uživatelské rozhraní)
- Monolitické jádro
  - všechny služby pohromadě - lepší výkon
  - problém s chybnými ovladači
  - Linux, *BSD
- Mikrojádro
  - poskytuje správu adresního prostoru, procesů, IPC
  - oddělení serverů => bezpečnost
  - Minix, Mach
- Hybridní jádro - Windows NT, MacOS X

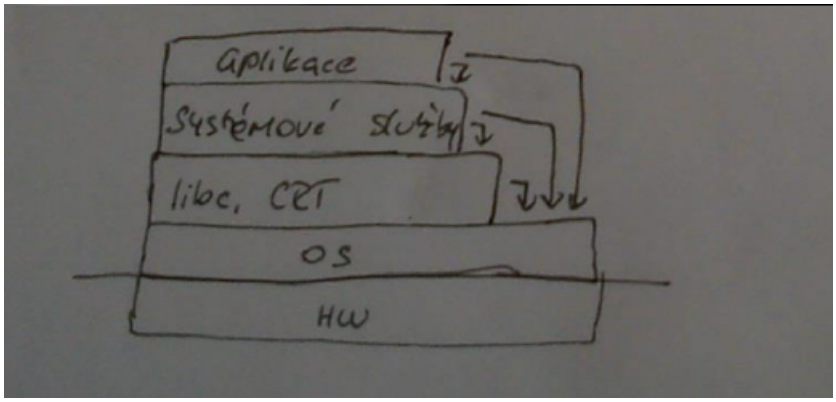


Obrázek 1: architektura jádra

2.1.3 Poskytovaná rozhraní

1. Hardware - přístup pouze k OS
2. Operační systém - privilegovaný režim, přístup z vyšších vrstev pouze skrze **SYSTÉMOVÉ VOLÁNÍ**
3. Standardní knihovna (libc, CRT) - alokace paměti, stará se o přidělování kousků paměti (malloc)
4. Systémové nástroje - logování, ls, dir atd
5. Aplikace

- Každá vrstva může volat nižší vrstvu, může přeskočit vrstvu
- Hranice mezi vrstvami nemusí být ostré



Obrázek 2: vrstvy HW/SW

2.2 Správa procesoru: procesy a vlákna, plánování jejich běhu, komunikace a synchronizace

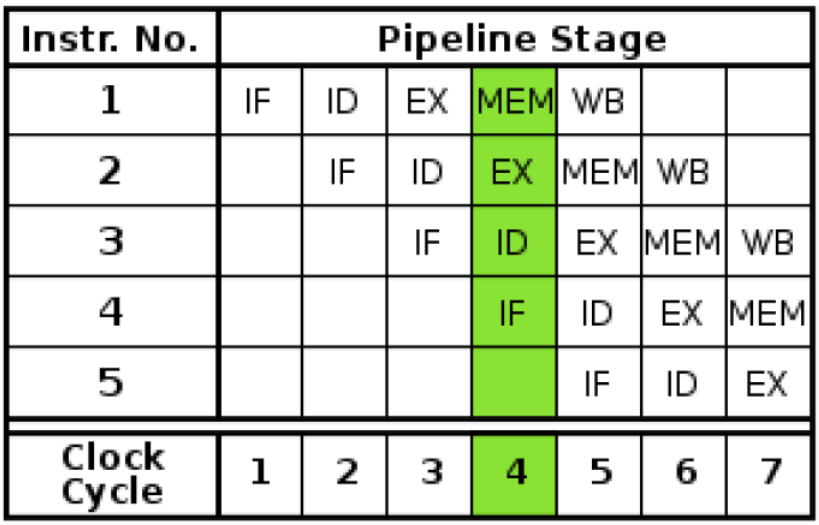
2.2.1 Procesor

OBECNÁ STRUKTURA CPU

- Je to jednotka, která zpracovává instrukce
- Aritmeticko-logická jednotka (ALU) - provádí výpočty
- Řídící jednotka - řídí chod CPU
- Registry - slouží k uchování právě zpracovávaných dat (násobně rychlejší přístup než do paměti)

INSTRUKČNÍ SADA

- Sada ovládající procesor (specifikovaná pro daný CPU/rodinu CPU)
- Instrukce a jejich operandy jsou reprezentovány jako čísla - strojový kód
- Každá instrukce má obvykle 0 až 3 operandy (může to být registr, konstantu nebo místo v paměti)
- Pro snazší porozumění, se instrukce CPU zapisují v jazyce symbolických adres (též vulgárně označován jako Assembler)
- Instrukce jsou zpracovávány (z důvodu větší efektivity) v několika krocích:
  1. načtení instrukce do CPU (Fetch)
  2. dekódování instrukce (decode)
  3. výpočet adres operandů
  4. přesun operandů do CPU
  5. provedení operace (Execute)
  6. uložení výsledku (Write-back)
- Pipelining - umožňuje zvýšit efektivitu CPU (díky rozdělení do jednotlivých kroků je možné provádět více instrukcí zároveň)
- Superskalární procesor - procesor může mít víc jednotek např. pro výpočty (FPU, ALU), musíme se postarat o správnou synchronizaci
- Problém s podmíněnými skoky (branch prediction) - jakmile se má provést nějaký podmíněný, nevíme, která instrukce bude následovat -> může se stát, že se musí vyprázdnit instrukci z pipeline -> zpomalení



Obrázek 3: pipelining

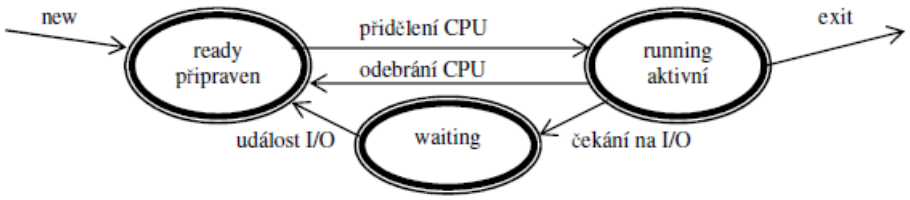
2.2.2 Procesy

- Neformálně: proces = běžící program (vykonává činnost)
- Proces charakterizuje:
  - Kód programu
  - Paměťový prostor
  - Data – statická a dynamická (halda – jsou tam uložená data, které se alokují při běhu procesu, typicky objekty alokované malloc nebo volané přes new)
  - Zásobník
  - Registry
- Oddělení jednotlivých úloh (abstrakce)
- Multiprogramování: (zdánlivě) souběžný běh více procesů
  - kooperativní - proces si sám odebere procesor
  - preemtivní - OS sám určuje, kdy odebere procesu procesor (velikost časového kvanta)
- Komunikace mezi procesy, sdílení zdrojů - synchronizace
- Informace o procesu má OS uloženy v Process Control Block (PCB)
  - identifikace procesu
  - informace o stavu
  - adresu instrukce, kterou bude program pokračovat
  - stav registrů

  - informace k plánování procesů
  - informace o přidělené paměti
  - informace o používaných I/O zařízeních, otevřených souborech, atd.
- potřeba plánování

OBECNÝ ŽIVOTNÍ CYKLUS PROCESU

- Nový (new) – proces byl vytvořen
- Připravený (ready)- proces čeká, až mu bude přidělen CPU
- Běžící (running) – CPU byl přidělen a právě provádí činnost
- Čekající (waiting/blocked) – proces čeká na vnější událost (např. na vyřízení I/O požadavku, synchronizační primitiva)
- Ukončený (terminated) – proces skončil svou činnost (dobrovolně x nedobrovolně)



Obrázek 4: stavy procesu

2.2.3 Plánování procesů

- Potřeba efektivně plánovat procesorový čas
- Časové kvantum: maximální čas přidělený procesu
- Samotné přepnutí procesu má režii (uložení kontextu, vyprázdnění cache) -> latence
- Symmetric Multi-Processing - přibývá problém, jak vybrat procesor
  - oddělené plánování pro každý procesor
  - maska affinity (definuje procesor, na kterém může proces běžet)

TYPY PLÁNOVÁNÍ:

- Dlouhodobé – rozhoduje, zda bude přijat k běhu (změna stavu z NEW na READY)
- Střednědobě – načtení/odložení procesu do sekundární paměti
- Krátkodobé – rozhoduje dostupné procesy ke spuštění na CPU

RŮZNÉ TYPY ÚLOH/SYSTÉMŮ:

- Iterativní
- Dávkové zpracování
- Pracují v reálném čase

OBECNÉ POŽADAVKY NA PLÁNOVÁNÍ PROCESŮ:

- Spravedlnost – každému procesu by v rozumné době měl být přidělen CPU
- Vyváženost – celý systém běží
- Efektivita – maximální využití CPU
- Maximalizace odvedené práce (throughput)
- Minimalizace doby odezvy
- Minimalizace doby průchodu systémem (turnaround) – od spuštění do konce procesu, aby byl co nejmenší časový úsek

ALGORITMY PRO PLÁNOVÁNÍ PROCESŮ

- FIRST-COME-FIRST-SERVED
  - První proces získává procesor
  - Jednoduchý, neefektivní
  - Nepreemptivní
- SHORTEST JOB FIRST
  - Vybere se takový proces, který poběží nejkratší dobu
  - Je potřeba znát (odhadnout) čas, který proces potřebuje
  - Nepreemptivní
- SHORTEST REMAINING TIME NEXT
  - Pokud aktivní proces potřebuje k dokončení činnosti méně času než aktuální, je spuštěn
  - Preemptivní
- ROUND ROBIN (CHODÍ PEŠEK OKOLO)
  - Každý proces má pevně stanovené kvantum
  - Připravené procesy jsou zařazeny ve frontě a postupně dostávají CPU
  - Nevýhoda: přiděluje stejný čas jak systémovým, tak uživatelským procesům
- VÍCEÚROŇOVÁ FRONTA
  - Každý proces má definovanou prioritu
  - Statické x dynamické nastavení priority (např. vyšší priorita pro I/O)
  - Systém eviduje pro každou frontu (čekající procesy)

  - Riziko vyhladovění procesů s nízkou prioritou (Rozšíření: nastavení různých velikostí kvant pro jednotlivé priority)
  - Používán Windows NT, staršími Linuxovými jádry
- SHORTEST PROCESS NEXT - Používá se odhad, podle předchozí aktivity procesu
- GUARANTEED SCHEDULING
  - Máme-li n procesů, každý proces má získat 1/n CPU
  - Volí se proces s nejmenším poměrem
  - Používají novější Linuxové jádra
- LOTTERY SCHEDULING - Proces dostane příděl "losů"
- FAIR-SHARE SCHEDULING - Plánování podle skupin procesů (např. podle uživatelů)

2.2.4 Vlákna

- OS umožňují rozdělit procesy na víc vláken (je základní entitou)
- Každé vlákno má svůj zásobník + registry (přepnutí vláken v rámci procesu je rychlejší)
- vlákna v rámci procesu sdílí paměťový prostor, data a prostředky -> nové problémy se synchronizací

IMPLEMENTACE VLÁKEN:

- Jako knihovna v uživatelském prostoru (první Linuxové systémy)
- Součást jádra operačního systému (Windows a další Linuxové systémy)
- Kombinované řešení

PROCESY A VLÁKNA V UNIXECH

- Původně základní entitou byl proces
- Procesy tvoří hierarchii rodič-potomek (na vrcholu je proces init)
- Vytvoření procesu pomocí volaní fork(), který vytvoří klon procesu
- Přepsání procesu pomocí exec
- Komunikace mezi procesy: zasílání signálů, roury, ...
- Možnost nastavit niceness procesu (prioritu od -20 do 20)
- Vlákna do Unixu přidána až později (implementace se v rámci různých OS liší)
- V Linuxu (pthreads) vnitřně implementovány stejně jako procesy (task), ale sdílí paměťový prostor

PROCESY A VLÁKNA VE WINDOWS:

- Windows NT navrženy s vlákny jako základní entitou pro běh aplikace
- Proces sdružuje vlákna, plánování se účastní vlákno
- Sofistikovaný systém priorit při plánování vláken
- Priorita vláken je odvozena od třídy priority procesu
- Proměnlivá velikost kvant
- Priority boost - dochází k dočasnému navýšení priority
  - po dokončení I/O operace
  - po dokončení čekání na semafor
  - vlákno již dlouho neběželo

2.2.5 Komunikace

- IPC – Inter-process communication
- Procesy oddělené, potřeba kooperace
- Základní kategorie
  - Sdílená paměť
    - Procesy sdílejí kousek paměti - nutná spolupráce se správou paměti
    - Čtení i zápis, náhodný přístup
    - Deklarace, že paměť je sdílená + namapování do adresního prostoru
    - SIGNÁLY - Mechanizmus podobný přerušení (asynchronní volání)
    - ROURY - Mechanizmus umožňující jednosměrnou komunikace mezi procesy
  - Zasílání zpráv
    - obecný mechanizmus komunikace mezi procesy
    - vhodný pro počítače se sdílenou paměti i pro distribuované systémy
    - lze jeho pomocí implementovat vzájemné vyloužení (mutual exclusion)
  - Synchronizace
  - Vzdálené volání procedur

2.2.6 Synchronizace

- procesy a vlákna přistupují ke sdíleným zdrojům (paměť, souborový systém)
- příklad: současné zvýšení hodnoty proměnné o 1 (díky preemtivnímu plánování tohle opravdu může nastat)
  1. A: načte hodnotu proměnné X z paměti do registru (X = 1)
  2. A: zvýšení hodnotu v registru o jedna
  3. B: načte hodnotu proměnné X z paměti do registru (X = 1)
  4. A: uloží hodnotu zpět do paměti (X = 2)
  5. B: zvýší hodnotu v registru o jedna
  6. B: uloží hodnotu zpět do paměti (X = 2)
- řešení - atomické operace nebo synchronizace
- je potřeba zajistit, aby v průběhu manipulace s určitými zdroji nemohl manipulovat někdo jiný (vzájemné vyloužení)

ATOMICKÝ PŘÍSTUP DO PAMĚTI

- Obecné přístupy do paměti nemusí být atomické (záležitosti CPU, překladače)
- Lze vynutit určité chování -> klíčové slovo volatile – často záleží na překladači
- Memory barriers umožňují vynutit si synchronizaci (záležitost CPU)

ATOMICKÉ OPERACE:

- Test-and-Set (TAS): nastav proměnnou a vrať její původní hodnotu
- Compare-and-Swap (CAS): ověří, jestli se daná hodnota rovná požadované a pokud ano, přiřadí ji novou (CMPXCHG)
- Fetch-and-Add: vrátí hodnotu místa v paměti a zvýší jeho hodnotu o jedna (XADD)
- Load-link/Store-Conditional (LL/SS): načte hodnotu a pokud během čtení nebyla změněna, uloží do ní novou hodnotu

SYNCHRONIZAČNÍ PRIMITIVA

- slouží k řízení přístupu ke sdíleným zdrojům
- Petersonův algoritmus - umožňuje implementovat vzájemné vyloučení dvou procesů pomocí běžných operací

KRITICKÁ SEKCE

- Obecně třeba zajistit, aby se sdílenými zdroji pracoval jen jeden proces
- Část kódu, kdy program pracuje se sdílenými zdroji (např. pamětí)

- Pokud jeden proces je v kritické sekci, další proces nesmí vstoupit do své kritické sekce
- Každý proces před vstupem žádá o povolení vstoupit do kritické sekce
- Poadavky na kritickou sekci:
  - Vzájemné vyloučení – maximálně jeden proces je v daný okamžik v KS
  - Absence zbytečného čekání – není-li žádný proces v kritické sekci a proces do ní chce vstoupit, není mu bráněno
  - Zaručený vstup – proces snažící se vstoupit do KS, do ní v konečném čase vstoupí
- Řešení:
  - Zablokování přerušení (použitelné v rámci jádra OS); více CPU -> neefektivní
  - Aktivní čekání - Spinlocks – testujeme pořád dokola jednu proměnou

```
Př. 1) ŠPATNÉ - Vstup do kritické sekce a její uzamknutí
                není provedeno atomicky
while (lock); // čekej
lock = 1;
// kritická sekce
lock = 0;


Př. 2)
Uvažujeme následující atomickou operaci
bool test_and_set(bool * target) {
bool rv = *target;
*target = true;
return rv;
}

A kód:
while (test_and_set(&lock) == true);
// kritická sekce
lock = false;


Př. 3)
Uvažujeme následující atomickou
operaci, která prohodí dvě hodnoty
void swap(bool *a, bool *b) {
bool tmp = *a;
*a = *b;
*b = tmp;
}
```

```
A kód
key = true;
while(key == true)
swap(&lock, &key);
// kritická sekce
lock = false;
```

SEMAFOR

- Chráněná proměnná obsahující počítadlo s nezápornými celými čísly
- Operace P (proberem – zkusit): pokud je hodnota čísla nenulová sníží hodnotu o jedna, jinak čeká, až bude hodnota zvýšena (operace někdy označena i jako wait)

```
void P (Semaphore s) {
while (s <= 0) { }
s--;
}
```

- Operace V (verhogen – zvýšit): zvýší hodnotu o jedna (operace označování jako signal, post)

```
void V (Semaphore s) {
s++;
}
```

- perace P a V se provádí atomicky - operace jsou na začátku a na konci zamykané pomocí spinlocku, protože je kód relativně krátký, tak se nečeká moc dlouho na čekání
- Binární semafor – může nabývat hodnot 0, 1 (mutex, implementace kritické sekce)
- Obecný semafor – slouží k řízení přístupu ke zdrojům, kterých je končené množství
- Implementace s pomocí aktivního čekání nebo OS (pasivní čekání)

```
struct sem {
int value;
struct process * list;
};

void P (struct sem * s) {
s -> value--;
if (s -> value < 0) {
// přidej proces s->list;
block(); // uspi aktuální proces
}
}
```

```
void V(struct sem * s) {
s-> value++;
if (s -> value <= 0) {
// odeber proces P z s s-> list
wakeup(P);
}
}
```

- Operace musí být provedeny atomicky

DALŠÍ SYNCHRONIZAČNÍ NÁSTROJE:

- Bariéry - Synchronizačních metod vyžadujících, aby se proces zastavil v daném bodě, dokud všechny procesy nedosáhnout daného bodu
- Read-Write zámky
  - Vhodné pro situace, které čtou i zapisují do sdíleného prostředku
  - Čtecí a zapisovací režim zámku
  - Vhodný pokud jde rozlišit čtenáře a písaře (písařů je víc)
- Monitor
  - Modul nebo objekt
  - V jeden okamžik může kteroukoliv metodu používat pouze jeden proces/vlákno
  - Nutná podpora programovacího jazyka
  - Java (synchronized), .NET (lock)

2.3 Problém uváznutí, jeho detekce a metody předcházení

2.3.1 Deadlock

- Uváznutí – systém se dostal do stavu, kdy nemůže dál pokračovat
- U množiny procesů došlo k uváznutí (deadlocku), pokud každý proces z této množiny čeká na událost, kterou pouze proces z této množiny může vyvolat.

UŽÍVÁNÍ PROSTŘEDKŮ:

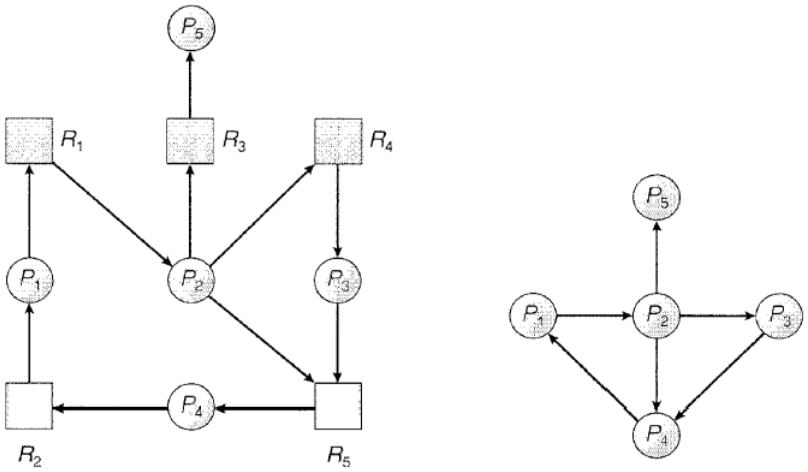
- Request – požadavek na prostředek, není-li k dispozici, proces čeká
- Use – vlákno s prostředkem pracuje
- Release – uvolnění prostředku pro další použití

UŽÍVÁNÍ PROSTŘEDKŮ

- Mutual exclusion – alespoň jeden prostředek je výlučně užíván jedním procesem
- Hold and wait – proces vlastní alespoň jeden prostředek a čeká na další
- No preemption – prostředek nelze násilně odebrat
- Circular wait – cyklické čekání (proces A vlastní prostředek 1, chce prostředek 2, který drží B a současně žádá o 1)

2.3.2 Řešení deadlocku

- IGNORACE - "neřešení", v praxi často používané tzv. pštrosí algoritmus – strčí se hlava do písku a dělá se, že žádný problém není
- DETEKCE
  - Pokud vznikne deadlock, je detekován a některý proces odstraněn
  - K detekci se používá alokační graf prostředků a graf čekání
  - Alokační graf:
    - Orientovaný graf
    - Dva typy uzlů – prostředek, proces
    - Hrana proces-prostředek – proces čeká na prostředek
    - Hrana prostředek-proces – prostředek je vlastněn procesem
  - Graf čekání vznikne vynecháním uzlů prostředků a přidáním hran Pn -> Pm pokud existovaly hrany Pn -> R a R ->Pm, kde Pn a Pm jsou procesy a R je prostředek
  - Ukázka alokačního grafu a grafu čekání:



Obrázek 5: alokacni graf a graf cekani

  - Deadlock vznikne, pokud je v grafu čekání na cyklus
- ZAMEZENÍ VZNIKU (PREVENCE)
  - Snažíme se zajistit, že některá z podmínek není splněna
  - Zamezení výlučnému vlastnění prostředků (často nelze z povahy zařízení)
  - Zamezení držení a čekání
    - Proces zažádá o všechny prostředky hned na začátku
    - Problém s odhadem
    - Plýtvání a hladovění

    - Množství prostředků nemusí být známé předem
    - Jde použít i v průběhu procesu (ale proces se musí vzdát všech prostředků)
  - Zavedení možnosti odejmout prostředek – vhodné tam, kde nelze odejmout prostředky tak, aby nešlo poznat, že byly odebrány
  - Zamezení cyklickému čekání – zavedení globálního číslování prostředků a možnost žádat prostředky jen v daném pořadí
- VYHÝBÁNÍ SE UVÁZNUTÍ
  - Procesy žádají prostředky libovolně
  - Systém se snaží vyhovět těm požadavkům, které nemohou vést k uváznutí
  - Je potřeba znát předem, kolik prostředků bude vyžádáno
  - Tomu je přizpůsobeno plánování procesů
  - Bezpečný stav – existuje pořadí procesů, ve kterém jejich požadavky budou vyřízeny bez vzniku deadlocku
  - Systém, který není v bezpečném stavu, nemusí být v deadlocku
  - Systém odmítne přidělení prostředků, pokud by to znamenalo přechod do bezpečného stavu (proces musí čekat)
  - ALGORITMUS NA BÁZI ALOKAČNÍHO GRAFU
    - Vhodný, pokud existuje jen jedna instance každého prostředku
    - Do alokačního grafu přidáme hrany (proces-prostředek) označující potenciální žádosti procesu a prostředky
    - Žádosti a prostředek se vyhoví pouze tehdy, pokud konverze hrany na hranu typu (prostředek-je vlastněn-procesem) nepovede ke vzniku cyklu
  - BANKÉŘŮV ALGORITMUS
    - Vhodný tam, kde je větší počet prostředků daného typu
    - Na začátku každý proces oznámí, kolik prostředků jakého typu bude maximálně potřebovat
    - Při žádosti o prostředky systém ověří, jestli se nedostane do nebezpečného stavu
    - Pokud nelze vyhovět, je proces pozdržen
    - Porovnávají se volné prostředky, s aktuálně přidělenými a maximálními
    - Uvažujme m prostředků a n procesů
    - Matice m x n
      - Max – počet prostředků, které bude každý proces žádat
      - Assigned – počet přiřazených prostředků jednotlivým procesům
      - Needed – počet prostředků, které bude proces ještě potřebovat (evidentně needed = max – assigned)
    - Vektory velikosti m

Assigned

	K	L	M	N
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Needed

	K	L	M	N
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

E = <6, 3, 4, 2>
P = <5, 3, 2, 2>
A = <1, 0, 2, 0>

- Podmínku splňuje proces D → odebrán a A <- <2, 1, 2, 1>
- Podmínku splňuje proces A → odebrán a A <- <5, 1, 3, 2>
- Podmínku splňuje proces B → odebrán a A <- <5; 2; 3; 2>
- Podmínku splňuje proces C → odebrán a A <- <6, 3, 4, 2>
- Podmínku splňuje proces E → odebrán a A <- <6, 3, 4, 2>

Obrázek 6: bankéřův algoritmus

· E – počet existujících prostředků
· P – počet aktuálně držených prostředků
· A – počet zdrojů

1. Najdi řádek i v needed takový, že needed[i] <= A, pokud takový není, systém není v bezpečném stavu
2. Předpokládej, že proces skončil a uvolnil své zdroje; A <- A + assigned[i] a odstraň řádný i ze všech matic
3. Opakuj body 1 a 2 dokud nejsou odstraněny všechny procesy nebo není jasné, že systém není v bezpečném stavu

2.4 Správa operační paměti: segmentace, stránkování, virtuální paměť.

2.4.1 Operační paměť

- zásadní část počítače
- uložení kódu a dat běžících v OS
- přístup k zařízením (DMA) - Direct Memory Access - přístup zařízení se namapuje do operační paměti a dá se poté přistupovat k zařízení přes operační paměť
- virtualní paměť umožňuje například přístup k souborovému systému
- z HW pohledu může byt operační paměť realizovaná různými způsoby (DRAM, SRAM, (EEP)ROM, ale i HDD, SSD, flash paměti)
- přístup k CPU k paměti nemusí být přímočarý (L1, L2 a L3 cache)
- pro jednoduchost budeme HW stránku věci zanedbávat

2.4.2 Stránkování

- adresní (logický) prostor procesu je rozdělen na menši úseky - stránky (pages)
- fyzická paměť je rozdělena na úseky stejné délky - rámce (frame, page frames)
- provádí se mapování logický adres -> na fyzické
- procesy už nemusí být umístěny v souvislých blocích paměti
- výpočet fyzické adresy musí byt implementovaný HW (efektivita) - pokud by implementoval OS, nemohl by ho používat
- CPU si udržuje stránkovací tabulku
- logická adresa má dvě části: pd, kde p je číslo stránky a d je offset
- fyzická adresa vznikne jako fd, kde f je číslo rámce v tabulce stránek příslušného stránce p

2.4.3 Segmentace

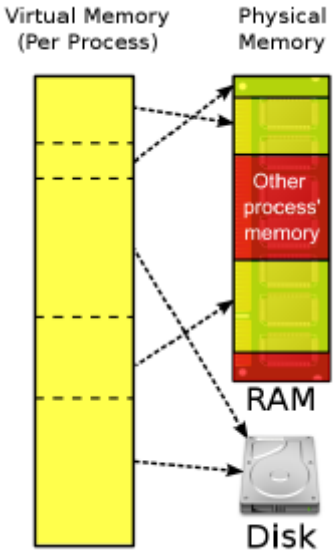
- paměť je rozdělena na několik segmentů (například kód, data, zásobník)
- speciální případ přidělování souvislých bloků paměti
- stránkování je obvykle pro programátora nerozeznatelné
- naproti tomu segmentace umožňuje rozdělit program do logických celků (kód, data)
- při použití segmentace a stránkování programy nepracují přímo s lineární adresou
- používají adresu ve tvaru segment + offset a ta se až převádí na fyzickou adresu pomocí stránkování
- ochrana paměti přes začátek a délku segmentu + oprávnění

2.4.4 Virtuální paměť

MOTIVACE

- paměť RAM je relativně drahá -> nemusí vždy dostačovat
- aktualně používaná data (například instrukce) musí byt v RAM, nepoužívaná data nemusí (velké programy -> nepoužívané funkce)
- je vhodné rozšířit primární paměť (RAM) o sekundární (např.: HDD)
- zvětšením dostupné paměti je možné zjednodušit vývoj aplikací (není potřeba se omezovat v množství použité paměti)
- sekundární paměť bývá řádově pomalejší

- k efektivní implementaci je potřeba spolupráce HW (MMU) a OS
- z pohledu aplikace musí být přístup k paměti transparentní
- virtuální paměť (VM) je součástí soudobých OS (swapování)
  - Windows NT - stránkovací soubor (pagefile.sys)
  - Linux - swap partition (ale může být i soubor)
- bezpečnost dat v sekundární paměti? (např.: po vypnuti počítače, řeší se např. kódováním)



Obrázek 7: virtuální paměť

VIRTUÁLNÍ PAMĚŤ

- způsob správy operační paměti počítače
- umožňuje předložit běžícímu procesu adresní prostor paměti, který je uspořádán jinak nebo je dokonce větší, než je fyzicky připojená operační paměť RAM
- virtuální adresy - pracují s nimi strojové instrukce, resp. běžící proces
- fyzické adresy - odkazují na konkrétní adresové buňky paměti RAM
- převod mezi virtuální a fyzickou adresou je zajišťován samotným procesorem
- virtuální paměť je implementována pomocí stránkování paměti spolu se stránkováním na disk, které rozšiřuje operační paměť o prostor na pevném disku

2.5 Správa diskového prostoru: oddíly, souborové systémy, zajištění konzistence dat

2.5.1 Souborové systémy

- způsob organizace dat ve formě souborů (a většinou i adresářů) tak, aby k nim bylo možné snadno přistupovat
- souborové systémy jsou uloženy na vhodném typu elektronické paměti, která je umístěna přímo v počítači (pevný disk nebo CD,...)

MOTIVACE

- potřeba uchovávat větší množství dat (primární paměť nemusí dostačovat)
- data musí být perzistentní (musí přežít ukončení procesu)
- k souborům musí být umožněn souběžný přístup
- řešení v podobě ukládání dat na vnější paměť (např.: disk)
- data ukládána do souborů tvořících souborový systém (File System/FS)
- soubor jako proud bytů (doprovázen doplňujícími informacemi)
- souborový systém jako abstrakce (odstínění od implementačních detailů)
- zajímavý problém: pojmenování objektů (souborů) a jejich organizace

OPERACE SE SOUBORY

- create - vytvoření souboru
- write/append - zápis do souboru (na konec, popř. přepis); souvislý blok vs. postupný zápis
- read - čtení ze souboru (do přichystaného bufferu)
- seek - změna pozice
- erase - odstranění souboru (uvolnění místa); link a unlink
- truncate - zkrátí daný soubor na požadovanou velikost
- open - otevře soubor, aby s nim šlo manipulovat přes popisovač (file descriptor, file handle)
- close - uzavře soubor
- "soubory"nemusí mít jméno
- jeden soubor může byt otevřen vícekrát (vice ukazatelů na pozici v souboru)

ORGANIZACE SOUBORŮ

- soubory jsou rozlišované podle názvů (často specifikované pro daný OS nebo FS)

- rozlišování velkých a malých písmen (Unix vs. MS-DOS a Windows)
- MS-DOS: požadavek na jméno souboru ve tvaru 8+3: jméno + přípona (zůstalo zachováno ve FAT)
- typicky se soubory organizuji do adresáře (složek)
- každý adresář může obsahovat běžné (popř. speciální) soubory i další adresáře -> stromová struktura
- k přístupu k souboru se používají:
  - absolutní cesty: /foo/bar/baz.dat
  - relativní cesty: foo/bar.dat -> každý proces má aktuální adresář
- operace s adresáři: Create, Delete, OpenDir, CloseDir, ReadDir, Rename
- struktura nemusí být hierarchická -> obecný graf (acyklický, cyklický)
  - hardlink - ukazatel na soubor (jeho tělo/obsah) (výhoda - transparentnos, nevýhoda - odkaz pouze v jednom FS)
  - symlink - soubor jako ukazatel na jiný soubor (specifikovaný cestou) (lze používat odkaz na jiný FS, ale nejsou plně transparentní - může vzniknout odkaz na neexistující soubor)

2.5.2 Oddíly

DĚLENÍ DISKU

- každý fyzický disk se skládá z jedné nebo více logických částí (partition, oddíl); popsané pomocí partition table daného disku
- v každém partition může existovat souborový system (označovaný jako svazek)
- v Unixech je každý svazek připojen (mounted) jako adresář (samostatný svaze pro /, /home, /usr)
- ve Windows jednotlivé svazky označeny (a:, b:, c:, . . .); ale funguje i mountovani (preferovaný jeden svazek pro vše)
- Virtual File System (VFS)
  - využití abstrakce =) umožňující kombinovat různé FS do jednoho VFS
  - možnost připojit běžný soubor jako svazek (i svazek jako soubor)
  - síťové disky (NFS, CIFS)

STRUKTURA SOUBORŮ

- často je soubor chápán jako proud bytů
- sekvenční vs. náhodný přístup
- společně s daty jsou k souboru připojena metadata (atributy)

- vlastník souboru
- přístupová práva
- velikost souboru
- příznaky (skrytý, archivace, spustitelný, systémový)

2.5.3 Souborové systémy - implementace

OČEKÁVÁME:

- (budeme předpokládat souborové systémy pro práci s disky s rotujícími částmi)
- schopnost pracovat se soubory a disky adekvátní velikosti
- efektivní práce s místem (evidence volného místa, nízká fragmentace)
- rychlý přístup k datům
- eliminace roztroušení dat na disku
- odolnost proti poškození při pádu systému (výpadku napájení) -> rychlé zotavení
- snapshoty - obraz disku v jednom okamžiku ke kterému jsme schopni se vrátit
- komprese dat
- možnost zvětšovat/zmenšovat FS za běhu
- kontrolní součty
- defragmentace za běhu
- správa oprávnění

STRUKTURA DISKU

- pro jednoduchost předpokládáme, že struktura disku je lineární
- MBR - master boot record: informace o rozdělení disku na svazky + zavaděč
- sektor disku - obvykle velikost 512 B
- pracuje se s většími bloky 1 - 32 kB (často 4 kB nebo 8 kB)
- jednotlivé svazky obsahují souborový systém (vlastní organizace dat)

FAT

- souborový system pro MS-DOS (přežil až do Windows ME)
- jednoduchý design
- soubory se jmény ve tvaru 8+3, nepodporuje oprávnění

- nemá metody proti poškození dat
- disk rozdělený na bloky (clustery)
- soubory popsány pomocí File Allocation Table (FAT) - spojový seznam
- disk (oddíl) rozdělen na 4 části:
  - bootsector (rezervovaná oblast) + informace o svazku
  - 2x FAT
  - kořenový adresář
  - data
- adresáře jako soubory, kořenový svazek je vytvořen hned na začátku
- původní FAT nepodporoval adresáře
- FAT12, 16, 32: podle velikosti clusteru (max. kapacity - 32 MB, 2 GB, 8 TB)
- Virtual FAT
  - s Windows 95
  - podpora dlouhých jmen (LFN)
  - až 256 znaků
  - soubor má dvě jména - dlouhé a ve tvaru 8+3
- exFAT
  - určen pro flash paměti
  - podpora větších disků (512 TB, 64 ZB)
  - podpora v novějších Windows (původně Windows CE 6)

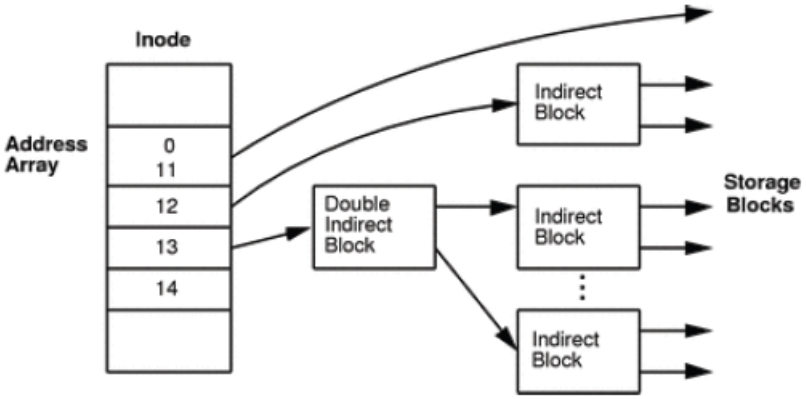
UFS: UNIX FILE SYSTEM

- v různých variantách přítomných v unixových OS - BSD, Solaris, System V, Linux (ext[2,3,4])
- disk se skládá:
  - bootblock - místo pro zavaděč OS
  - superblock - informace o souborovém systému
  - místo pro inody
  - místo pro data

  INODY

  - struktura popisující soubor
  - informace o souboru
    - typ souboru, vlastníka (UID, GID), oprávnění (rwx)
    - časy (vytvoření, přístup)

    - počet ukazatelů, počet otevřených popisovačů
  - informace o uložení dat
  - struktura inody umožňuje mít řídké soubory
  - adresář je soubor obsahující sekvenci dvojic (jméno souboru, číslo inody)
  - k evidenci volného místa a inod se používají bitmapy



Obrázek 8: inode

FS V LINUXU

- Linux nemá jeden hlavní FS
- nejčastěji se používá: Ext2/3/4
- název souboru může mít až 256 znaků
- vychází z UFS
- ext2: maximální velikost souboru 16 GB - 2 TB, disku: 2 TB - 16 TB
- ext3: přidává žurnál (3. úrovně - journal, ordered, unordered), binárně kompatibilní s ext2
- ext4: maximální velikost souboru 16 GB - 2 TB, disku 1 EB; optimalizace alokací

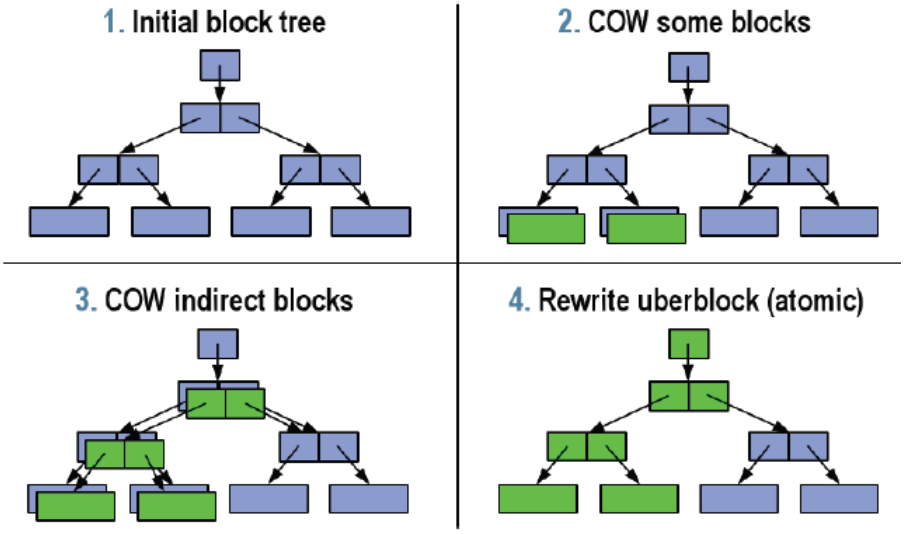
NTFS

- hlavní souborový systém Windows NT
- kořeny v OS/2 a jeho HPFS (vyvíjen od roku 1993)
- velikost clusteru podle velikosti svazku (512 B - 4 KB) <- max. velikost disku 256 TB, max. velikost souboru 16 TB
- oproti FAT (souborovému systému W9x) ochrana před poškozením + práva

- žurnálování a transakce
- dlouhé názvy (255 znaků) + unicode
- podpora standartu POSIX; hardlinky, symlinky
- adresáře
  - opět technicky soubory
  - jména v B+ stromech
  - části metadat jsou součástí adresáře, část metadat je součástí souborů
- **STRUKTURA DISKU**
  - na začátku disku: boot sector
  - 12% MFT (Maste File Table), 88% data souborů
  - MFT je soubor popisující všechny soubory na FS (MFT je taky soubor)
  - MFT se skládá ze záznamů o velikosti 1 KB
  - každý soubor je popsán tímto záznamem
  - informace o souborech včetně jména, časů, atd. uloženy jako záznam v MFT jako dvojice atribut-hodnota
  - tělo souboru je taky atribut (uniformní přístup, možnost uložit malé soubory přímo do MFT
  - v případě potřeby může jeden soubor zabrat vice záznamů v MFT
  - případně lze použít místo mimo MFT (rezidentní a nerezidentní atributy)

2.5.4 Zajištění konzistence dat

- používání blokového zařízení: RAID (Redundant Array of Independent Disk)
  - RAID-0 (stripping) - nepomůže konzistenci
  - RAID-1 (mirroring)
  - RAID-2 Hammingův kód - dělí data po bitech, disk pro paritu
  - RAID-3 dělí data po bytech, XOR, disk pro paritu
  - RAID-4 podobné jako RAID-3, používá paritní bloky
  - RAID-5 podobné jako RAID-4, paritní bloky jsou distribuovány
  - RAID-6 podobné jako RAID-5, Reed-Solomon kód, dva paritní bloky
- u dat jsou evidovány kontrolní součty (ochrana proti tichému poškození (chyba HW i SW)
- konzistence založena na metodě Copy-on-Write
- používaná data nikdy nejsou přepsána (nejdřív jsou zapsána data a pak jsou (automaticky) změněna metadata



1. alokujeme bloky pro data
2. pro kazda data vytvorime novy blok
3. vytvorime metadata, ktere predstavuji bloky dat
4. po zapsani provedeme prohozeni těch částí, které jsou měněné

Obrázek 9: zachování konzistence

- infrastruktura pro vytváření snapshotů/klonů souborového systému
- výhodné slučovat operace do transakcí

Copy-on-Write (COW)

- může být užitečné sdílet paměť (komunikace, úspora místa)
- dva stejné programy/knihovny v paměti
- stránky více procesů jsou navázány na jeden rámec
- daná stránka má nastavený příznak CoW
- dojde-li k pokusu o zápis, vznikne výjimka a procesu je vytvořena kopie rámce
- bude-li na rámec s příznakem CoW odkazovat jenom jedna stránka, příznak se odstraní
- fork() - vytvoří identickou kopii procesu; data jsou sice izolovaná, ale je možné sdílet kód
- úspora místa; úspora strojového času (není potřeba vytvářet kopie stránek/rámců); nízká penalizace pokud stránky přepíšeme
- virtuální paměť umožňuje další optimalizace (mapování souborů do paměti, atd.)

3

Klasifikace (LAN/MAN/WAN) a služby počítačových sítí. Síťová architektura TCP/IP a referenční model ISO OSI. Strukturovaná kabeláž, přepínaný Ethernet a WLAN. Protokol IP, adresa a maska, směrování, IP multicast. Protokoly TCP a UDP, správa spojení a řízení toku dat. Systém DNS, překlad jména (na IP adresu, reverzní), protokol. Služby WWW, elektronické pošty, přenosu souborů a vzdáleného přihlášení.

3.1 Klasifikace (LAN/MAN/WAN)

Klasifikace sítí podle různých kritérií: rozlehlost, rychlost přenosu (klasické, vysokorychlostní), forma aplikace, dělení podle postavení uzlů (perr-to-peer, klient-server), podle druhu přenášení signálů (analog vs. digital) aj.

3.1.1 Dělení podle rozlehlosti

- **PAN (Personal Area Network)**: Sítě s nejmenší rozlehlostí. Propojení mobilů, PDA, atd. Požadavky na odolnost vůči rušení, nízká spotřeba energie, snadná implementace.Ppřenosová rychlost není prioritou (zpravidla jen několik Mb/s). Dosah pouze několik metrů. Typické technologie Bluetooth, IrDA, Wifi.
- **LAN (Local Area Network)**: Sítě propojující koncové uzly typu počítač, tiskárna, server. V soukromé správě, dosah jednotky km (v rámci budovy/komplexu budov). Přenosové rychlosti 10Mb/s až 1Gb/s. Sdílené využití přenosového média. Ethernet, Wifi.
- **MAN (Metropolitan Area Network)**: Propojení několika LAN (účelem přenosové sítě, charakterem lokální). V rámci města (desítky km). Přenosové rychlosti jak vyšší (několik Gb/s), tak i nižší (<1Gb/s) ve srovnání s LAN.
- **WAN (Wide Area Network)**: Rozsáhlé sítě spojující LAN/MAN (páteřní sítě, telekomunikační - broadband). Velké vzdálenosti (prakticky neomezené). Mohou být soukromé i veřejné. Vysoké přenosové rychlosti (až stovky Gb/s). Příklad GPRS, xDSL, aj. Pronájem kapacity sítě = vyhrazené nesdílené využití přenosového média.

3.1.2 Dělení podle topologie

Topologie počítačové sítě říká, jak jsou vlastně prvky v této síti uspořádány.

Kruhová topologie (RING) Každý počítač je propojen přímo s předchozím a následujícím počítačem v kruhu. V LAN je používána velmi málo, používá se v průmyslových sítí nebo sítích MAN.
Výhody:

- lehce rozšiřitelná struktura
- malý počet spojů

- snadné vysílání, zprávy v kruhu od stanice ke stanici

Nevýhody:

- výpadek libovolné stanice zapříčiní výpadek celé sítě, úplný výpadek sítě při přerušení kabelu v libovolném místě
- poměrně veliké nebezpeční odposlechu síťové komunikace, která prochází přes spojovací počítače

Problém výpadku se řešil tzv. dvojitým kruhem, ve kterém byly stanice propojeny dvěma kruhy, každý v opačné směru.

Sběrnicová topologie (BUS) Tato topologie patří k nejstarším, všechny stanice jsou připojeny na pasivní společné médium, které sdílejí. Dnes už se tato topologie příliš nepoužívá, ale na začátku devadesátých let byla dominantní. Tím společným médiem byl koaxiální kabel, pomocí kterého se jednotlivé počítače připojily do sítě.
Výhody:

- nezávislost stanic na výpadku libovolné jiné stanice
- levné náklady takového řešení
- neexistence aktivních prvků
- snadné všesměrové vysílání

Nevýhody:

- úplný výpadek sítě při přerušení kabelu v libovolném místě
- nutnost vyřešení přístupu stanic k médiu (kdo bude vysílat)

Výhody a nevýhody jsou relativní a poplatné době. To, že se v dobách používání této topologie v sítích LAN, považovala absence aktivních prvků za výhodu, bychom v dnešní době řadili spíše k nevýhodě.

Hvězdicová topologie Tato topologie je dnes jednoznačně nejpoužívanější topologií v sítích LAN. Myšlenka spočívá v tom, že existuje centrální prvek, který spojuje všechny prvky. Dříve tím centrálním prvkem býval počítač, dnes je aktivní prvek (HUB nebo SWITCH).
Výhody:

- lehce rozšiřitelná struktura
- výpadek libovolné stanice neznamená výpadek celé sítě
- větší možnosti zabezpečení, při použití aktivních prvků typu SWITCH je většina síťové komunikace skryta před ostatními účastníky sítě

Nevýhody:

- nutnost použití hubu nebo switche
- vyžaduje veliké množství kabelů a je tak náročná na montáž

Páteřní topologie Páteřní topologií rozumíme situaci, kdy pomocí určité topologie propojujeme celé sítě LAN. Páteřní topologie může být zapojena jako sběrnice, hvězda i kruh, často se používá zapojení typu kruh. Jejím základem je vytvoření nezávislé hlavní části, která propojuje důležité celky. Na ni se naopak připojují různé subsítě nebo segmenty. V případě výpadku libovolného segmentu zůstává provoz na páteři neohrožen. Páteř může mít vyšší přenosovou rychlost.

3.1.3 Služby počítačových sítí

- připojení k síti
- vzdálený přístup, sdílení výpočetních prostředků a přenos dat (sdílené databáze, perr-to-peer sítě, sdílené soubory)
- sdílení technických prostředků (tiskárny, faxy, disky, apod.)
- adresářové služby (jednotný přístup do informačního systému a k informacím z centrální databáze, např. LDAP, Active Directory)
- elektronická pošta a výměna dokumentů (objednávky, faktury, atd.)
- online komunikace/multimedia (např. IRC, VoIP, straming, hry) - vysoké nároky na síť
- informační služby, internetové aplikace (WWW, business a desktopové aplikace)
- monitorování a vzdálená administrace sítě

Komunikace uzlů a propojovacích prvků sítě na různých úrovních:

- nižší - přenos bloků dat, většinou nespolehlivý (bez potvrzení a opakování přenosu), nespojová komunikace
  - **unicast**: dvoubodová, základní
  - **multicast**: bod-skupina, např. straming, virtuální sítě
  - **broadcast**: bod-všichni, např. konfigurace a zapojení do sítě
- vyšší - komunikace aplikací, většinou spolehlivá (s potvrzením doručení, popř. opakování přenosu), spojově orientovaná (vytvořeno spojení mezi aplikacemi)
  - **peer-to-peer**: zpravidla rovnocenná výměna dat
  - **klient-server**: hiearchická, forma požadavek-odpověď

Typy koncových uzlů:

- **pracovní stanice** (work station, klient): převážně využívá služeb sítě
  - **tenký klient**: znakový/grafický HW terminál, pouze zprostředkování vstupu a výstupu pro vzdálený server, nemůže pracovat samostatně
  - **tlustý klient**: osobní počítač - klientské části síťových služeb i lokální úlohy, může (do určité míry) fungovat samostatně

- **server**: převážně poskytuje služby v síti
  souborový (FTP, NFS, SMB), databázový/adresářový (SŘBD, LDAP), poštovní (IMAP, POP3, SMTP), terminálový (telnet, SSH), informační/WWW (HTTP), komunikační (IM, VoIP), tiskový, aj.

3.2 Síťová architektura TCP/IP a referenční model ISO OSI.

Snaha o vytvoření univerzálního konceptu sítě (topologie, formy a pravidlakomunikace, poskytování služeb atd.). Požadavky: decentralizace služeb, rozumná adresace uzlů, data zasílána v nezávislých blocích, směrování bloků, zabezpečení, kontrola a řízení přenosu aj. Dříve proprietární uzavřená řešení, následně standatdizace s koncepcí komunikace nezávislé na implementaci.
Komunikace ve vrstvách

- definované službami poskytované vyšším vrstvám a využívající služby nižších vrstev, implementace skryté okolním vrstvám
- samostatné vrstvy s funkcemi podobnými v rámci vrstvy a odlišnými v různých vrstvách, nezávislé na implementaci

Komunikace mezi vrstvami pomocí **mezivrstvových protokolů** - na každé komunikující straně zvlášť; skrze **programová rozhraní**; prostřednictvím přístupových bodů; využívající tzv. služební primitiva.

Obecná služební primitiva

- žádost o službu (request)
- oznámení poskytovatele o přijetí žádosti (indication) - nepovinné
- odezva poskytovatele (response), příp. vytvoření spojení
- potvrzení odezvy žadatelem (confirmation) - nepovinné

Komunikace ve stejných vrstvách mezi entitami (zařízeními) pomocí **vrstvových protokolů**.

3.2.1 Protokol

= souhrn pravidel (norem a doporučení) a procedur pro komunikaci (výměnu dat). Obsahuje syntaktická a sémantická pravidla výměny protokolových datových jednotek.
Protokolové datové jednotky = režijní informace a data (rámce, pakety, segmenty). Komunikace zprostředkována sousední nižší vrstvou. Na straně odesílatele **zapouzdřování** od nejvyšší po nejnižší vrstvu. Na straně příjemce **rozbalování** dat v opačném směru.

Síťová (protokolová) architektura = definice vrstev, služeb funkcí, protokolů a forem komunikace. Normalizované (OSI, TCP/IP) a firemní proprietární (Novell NetWare, SMB, Apple Appletalk aj.)

Abstraktní referenční síťový model Abstrakce konkrétních síťových architektur - nemusí podporovat všechny funkce modelu (např. průmyslové sítě nepodporují směrování, propojení pomocí mostů a bran).

3.2.2 Referenční model ISO/OSI

Propojení otevřených systémů = zařízení podporující příslušné normy. Deinovány koncové uzly (koncová datová zařízení) a mezilehlé uzly (propojovací prvky).
Každá ze sedmi vrstev vykonává skupinu jasně definovaných funkcí potřebných pro komunikaci. Pro svou činnost využívá služeb své sousední nižší vrstvy. Své služby pak poskytuje sousední vyšší vrstvě. Podle referenčního modelu není dovoleno vynechávat vrstvy, ale některá vrstva nemusí být aktivní. Takové vrstvě se říká nulová, nebo transparentní.
Na počátku vznikne požadavek některého procesu v aplikační vrstvě. Příslušný podsystém požádá o vytvoření spojení prezentační vrstvu. V rámci aplikační vrstvy je komunikace s protějším systémem řízena aplikačním protokolem. Podsystémy v prezentační vrstvě se dorozumívají prezentačním protokolem. Takto se postupuje stále níže až k fyzické vrstvě, kde se použije pro spojení přenosové prostředí. Současně se při přechodu z vyšší vrstvy k nižší přidávají k uživatelským (aplikačním) datům záhlaví jednotlivých vrstev. Tak dochází k postupnému zapouzdřování původní informace. U příjemce se postupně zpracovávají řídící informace jednotlivých vrstev a vykonávají jejich funkce.

Fyzická vrstva Specifikuje fyzickou komunikaci. Aktivuje, udržuje a deaktivuje fyzické spoje mezi koncovými systémy. Definuje všechny elektrické a fyzikální vlastnosti zařízení (tvary konektorů; typy médií = kroucená dvojlinka, optické vlákno, mikrovlny; přenosové rychlosti). Obsahuje rozložení pinů, napěťové úrovně a specifikuje vlastnosti kabelů.
Funkce poskytované fyzickou vrstvou:

- Navazování a ukončování spojení s komunikačním médiem.
- Spolupráce na efektivním rozložení všech zdrojů mezi všechny uživatele.
- Modulace neboli konverze digitálních dat na signály používané přenosovým médiem (a zpět) (A/D, D/A převodníky).

HW zařízení: HUB, opakovač.

Linková vrstva Poskytuje spojení mezi dvěma sousedními systémy. Uspořádává data z fyzické vrstvy do logických celků = **rámce (frames)**: záhlaví s linkovou adresou příjemce a odesílatele (např. MAC) + data + zápatí s kontrolním součtem (CRC) celého rámce. Stará se o nastavení parametrů přenosu linky, oznamuje neopravitelné chyby. Formátuje fyzické rámce, opatřuje je fyzickou adresou a poskytuje synchronizaci pro fyzickou vrstvu. Příkladem je MAC u Ethernetu. Poskytuje propojení pouze mezi místně připojenými zařízeními a tak vytváří doménu na druhé vrstvě pro směrové a všesměrové vysílání.
HW zařízení: most, přepínač, síťová karta, přístupový bod.

Síťová vrstva Stará se o směrování v síti a síťové adresování. Poskytuje spojení mezi systémy, které spolu přímo nesousedí. Poskytuje funkce k zajištění přenosu dat různé délky od zdroje k příjemci skrze jednu případně několik vzájemně propojených sítí při zachování kvality služby, kterou požaduje přenosová vrstva. Síťová vrstva poskytuje směrovací funkce a také reportuje o problémech při doručování dat. Jednotkou přenosu je **síťový packet**: záhlaví se síťovou adresou příjemce a odesílatele (např. IP) + data + zápatí jen výjimečně.
Funkce poskytované síťovou vrstvou:

- abstrakce různých linkových technologií
- správa linkových spojení, multiplexování síťových spojení do linkových (více datových toků kombinováno do jednoho)
- formátování dat do packetů
- směrování packetů
- zjišťování a oprava chyb
- vytváření podsítí

HW zařízení: směrovač, brána.

Transportní vrstva Zajišťuje přenos dat mezi koncovými uzly. Jejím účelem je poskytnout takovou kvalitu přenosu, jakou požadují vyšší vrstvy. Vrstva nabízí spojově (TCP) a nespojově orientované (UDP) protokoly. Jednotkou přenosu je **transportní packet**: záhlaví s transportní adresou příjemce a odesílatele + data.
TCP: Protokol garantuje spolehlivé doručování a doručování ve správném pořadí. TCP také umožňuje rozlišovat a rozdělovat data pro více aplikací (například webový server a emailový server) běžících na stejném počítači. TCP využívá mnoho populárních aplikačních protokolů a aplikací na internetu, včetně WWW, e-mailu a SSH.
UDP: Na rozdíl od protokolu TCP nezaručuje, zda se přenášený datagram neztratí, zda se nezmění pořadí doručených datagramů, nebo zda některý datagram nebude doručen vícekrát. UDP je vhodný pro nasazení, které vyžaduje jednoduchost nebo pro aplikace pracující systémem otázka-odpověď (např. DNS, sdílení souborů v LAN).
Funkce poskytované transportní vrstvou:

- adresování (transportní na síťové)
- správa síťových spojení
- multiplexování a větvení
- rozdělení dat na datagramy, segmentace, formátování
- řízení proudu dat (správné pořadí datagramů), optimalizace služeb
- koncová detekce a oprava chyb

Umožňuje **duplexní přenos** (= přenos oběma směry).

Relační vrstva Smyslem vrstvy je organizovat a synchronizovat dialog mezi spolupracujícími relačními vrstvami obou systémů a řídit výměnu dat mezi nimi (např. sdílení síťového disku). Umožňuje vytvoření a ukončení relačního spojení, synchronizaci a obnovení spojení, oznamovaní výjimečných stavů. Jednotka přenosu je **relační packet**: poze data.
Funkce poskytované relační vrstvou:

- organizace a synchronizcae dialogu výměny dat (pomocí kontrolních bodů)
- zobrazení relačních spojení do transportních
- správa transportních spojení

Prezentační vrstva Funkcí vrstvy je transformovat data do tvaru, který používají aplikace (šifrování, konvertování, komprimace). Formát dat (datové struktury) se může lišit na obou komunikujících systémech, navíc dochází k transformaci pro účel přenosu dat nižšími vrstvami. Vrstva se zabývá jen strukturou dat, ale ne jejich významem, který je znám jen vrstvě aplikační.
Funkce poskytované prezentační vrstvou:

- transformace a výběr reprezentace dat
- formátování, komprese, zabezpečení (šifrování), integrita dat
- transparentní přenos zpráv (nezná jejich význam

Aplikační vrstva Účelem vrstvy je poskytnout aplikacím přístup ke komunikačnímu systému a umožnit tak jejich spolupráci.
Funkce poskytované aplikační vrstvou:

- zprostředkování funkcionality sítě
- přenos zpráv, určení kvality, synchronizace
- identifikace, stanovení pověření
- dohoda o ochraně, o opravách chyb

Protokoly: SMTP, SSH, Telnet, POP3, DHCP, FTP, DNS aj.

Funkce společné více vrstvám Komunikace **se spojením** má 3 fáze: navázání spojení, přenos dat, ukončení spojení. Dohoda na parametrech, použití potvrzování přijetí/nepříjetí (spolehlivost), stejné pořadí dat na vstupu i výstupu.
Komunikace **bez spojení**: při každém přenosu všechny parametry, nezávislý přenos datových jednotek, různé pořadí na vstupu a výstupu, spolehlivost i nespolehlivost.
Dále se může mezi těmito typy konvertovat. Transportní služby musí být se spojením.

3.2.3 TCP/IP

použití v síti Internet, nejpoužívanější síťová architektura. Síť tvořena směrovači, specializovanými bránami (bezpečnostní, aplikační, telekomunikační), lokálními sítěmi a koncovými zařízeními. Vlastní protokoly.

Vrstva síťového rozhraní Nejnižší vrstva umožňuje přístup k fyzickému přenosovému médiu. Je specifická pro každou síť v závislosti na její implementaci.

Síťová vrstva Vrstva zajišťuje především síťovou adresaci, směrování a předávání datagramů.

Transportní vrstva je implementována až v koncových zařízeních (počítačích) a umožňuje proto přizpůsobit chování sítě potřebám aplikace. Poskytuje transportní služby kontrolovaným spojením spolehlivým protokolem TCP nebo nekontrolovaným spojením nespolehlivým protokolem UDP.

Aplikační vrstva Vrstva aplikací. To jsou programy (procesy), které využívají přenosu dat po síti ke konkrétním službám pro uživatele.
Aplikační protokoly používají vždy jednu ze dvou základních služeb transportní vrstvy: TCP nebo UDP, případně obě dvě (např. DNS). Pro rozlišení aplikačních protokolů se používají tzv. porty, což jsou domluvená číselná označení aplikací. Každé síťové spojení aplikace je jednoznačně určeno číslem portu a transportním protokolem (a samozřejmě adresou počítače).

3.2.4 Bezpečnost

Obecné metody ochrany:

- omezování přenosu dat a přístupu k síti: blokace, filtrace
- autorizace přístupu: jméno a heslo, vícefaktorové, speciální protokoly
- zabezpečení kanálu: šifrování, výměna klíčů
- autenticita zpráv: digitální podpis, certifikáty

TCP/IP původně žádné zabezpečení, ponecháno na aplikaci.
Útoky:

- falešná adresace (spoofing)
- analýza hesla, trojský kůň
- odposlech
- odmítnutí služby (DoS, zahlcení, vyčerpání zdrojů)
- zneužití chyb aplikací
- ...

Ochrana:

- firewall (oddělení vnitřní sítě od vnější)
- překlad adres (NAT) - vlastní skrytá adresace
- aplikační brány (proxy)
- autentizace komunikujících stran

- zabezpečení komunikace (šifrování)
- opatření proti zahlcení
- ...

3.3 Strukturovaná kabeláž, přepínaný ethernet a WLAN

3.3.1 Strukturovaná kabeláž

Obecný pojem pro metalické a optické prvky, které umožňují propojení.

Telekomunikační přípojky Slouží pro připojení koncových uživatelských zařízení - např. stolní počítač, notebook, analogový nebo ISDN telefon, VoIP telefon či síťová tiskárna. Telekomunikační zásuvky bývají umístěny přímo v pracovních prostorách (např. kancelářích) každé budovy, a to buď přímo ve zdi, v parapetních žlabech, případně podlahových systémech tak, aby byly lehce dostupné.

Patch panely Na rozdíl od běžně dostupných zásuvek jsou patch panely umístěny v rozvaděčích v telekomunikační místnosti a nejsou tedy pro běžné uživatele přístupné. Patch panely slouží správci sítě k připojení jednotlivých uživatelů do aktivních zařízení jako jsou switche nebo telefonní ústředny. Pro připojení vodičů do zářezových konektorů se používá narážecí nástroj.

Horizontální kabely, Patch kabely Měděné kabely obsahující čtyři kroucené páry, které vzájemně propojují telekomunikační zásuvky a patch panely.

Koaxiální kabel se dnes už nepoužívá. Vnější (stínění) a vnitřní (jádro) vodič. BNC konektor. Maximálně 500m.

Kroucená dvojlinka je tvořena páry vodičů, které jsou po své délce pravidelným způsobem zkrouceny a následně jsou do sebe zakrouceny i samy výsledné páry. Oba vodiče jsou v rovnocenné pozici (i v tom smyslu, že žádný z nich není spojován se zemí či s kostrou), a proto kroucená dvojlinka patří mezi tzv. symetrická vedení. Maximálně 100m. Nestíněná (UTP) vs. stíněná (STP). Konektor RJ-45.

Jednou ze základních odlišností kroucené dvojlinky od koaxiálního kabelu je skutečnost, že na kroucené dvojlince není možné dělat odbočky. Kroucená dvojlinka je proto použitelná jen pro vytváření dvoubodových spojů. Nemožnost vytvářet odbočky pak ale nutně znamená, že prostřednictvím kroucené dvojlinky nelze vytvořit sběrnicovou topologii sítě.

Optická vlákna Vlákna imunní vůči elektromagnetickému rušení. Dvě vrstvy skla (obal a jádro). Vícevidové vlákno - odraz od rozhraní skel. Jednovidové vlákno - buzení laserem. Různé optické konektory (FC, LC, ST) nutné navařit. Vlákno simplexní, pro duplexní přenos dvojice vláken.

3.3.2 WLAN

= bezdrátové lokální sítě. Důvodem pro WLAN je rozšiřitelnost, nižší náklady, mobilita, snadná použitelnost, roaming (vysílače si klienta předávají). Připojení od poskytovatele i v rámci LAN. Norma IEEE 802.11.

Konfigurace

- peer-to-peer/ad-hoc: přímá komunikace mezi stanicemi (do 10ti stanic)
- AP (access point): propojení WLAN a LAN, bezpečnostní prvky (autorizace, filtrace, šifrování, atd.)
- s více AP (roaming): AP propojeny pevnou sítí, klient se připojuje k zařízení s nejsilnějším signálem
- point-to-point: připojení pomocí dvou AP

Přenosové medium jsou radiové vlny (2,4 GHz - 802.11b/g/n nebo 5GHz - 802.11a/n) Není třeba licence, vzájemné rušení.
antény: omezení výkonu normou ČTÚ na 100 mW

Bezpečnost

- obtížná ochrana proti odposlechu na fyzické vrstvě
- SSID: označení AP (název sítě), AP jej nemusí vysílat
- WEP: Autentizace stanic vůči AP (40 bitů heslo společně s MAC adresou), lze v krátkém čase prolomit
- WPA, WPA2: silná šifra AES, autentizace (heslo, EAP)