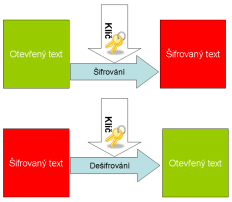
Šifrování

- princip kryptografických metod a jejich využití v praxi(symetrická, asymetrická, hybridní)
- zásada důvěrnosti
- šifrování z historického pohledu
- způsoby distribuce a uložení šifrovacích klíčů

Princip kryptografických metod a jejich využití v praxi(symetrická, asymetrická, hybridní)

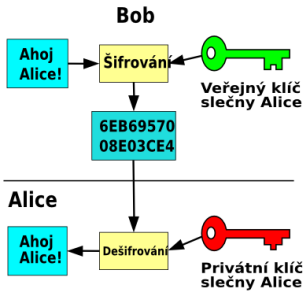
- **Symetrická**
  - vývoj od starověku po současnost
  - jeden klíč pro šifrování i dešifrování
  - většina symetrických šifrovacích algoritmů je velmi rychlá
  - síla (bezpečnost)
    - nesmí záviset na utajení algoritmu
    - síla šifer se poměřuje délkou klíče udávanou v bitech, např. 5-ti bitový klíč představuje 2^5 kombinací - tedy 32 různých klíčů (kombinací)
    - za bezpečné se dnes považují algoritmy, které používájí klíče o minimální délce 128bitů – tedy více než $3{,}4 \times 10^{38}$ kombinací



Princip fungování symetrické šifry

  - **Nevýhody**
    - nutnost sdílení tajného klíče - odesílatel a příjemce tajné zprávy se musí předem domluvit na tajném klíči
    - problém - Jak bezpečně předat klíč příjemci šifrovaných dat?
  - **Druhy**
    - **proudové šifry** - zpracovávají otevřený text po jednotlivých bitech
    - **blokové šifry** - rozdělí otevřený text na bloky stejné velikosti a doplní vhodným způsobem poslední blok na stejnou velikost
      - u většiny šifer se používá blok o 64 bitech, AES používá 128 bitů
  - algoritmy - např. DES, 3DES, CAST, IDEA, Blowfish, RC, ...
    - **DES** - algoritmus navržený pro bankovní sektor
    - RC - blokové šifry původně vyvinuté Ronaldem Rivestem a uchovávané jako obchodní tajemství firmy

- **Asymetrická**
  - používá dvojici klíčů: **veřejný** a **soukromý**
  - **veřejný klíč** může použít kdokoliv pro zašifrování zprávy
  - dešifrovat lze pouze pomocí **soukromého klíče**
  - eliminován problém s předáváním klíčů
  - algoritmy jsou výrazně (řádově 1000x) pomalejší než algoritmy symetrické
  - jelikož většina symetrických algoritmů pracuje pouze se speciálními čísly (např. prvočísly) používají se zde klíče o velikosti až 2048 bitů
  - algoritmy - např. RSA, Diffie-Hellman, DSS, ...
    - **RSA** - nejznámější asymetrická šifra
      - základ většiny systému využívajících asymetrické šifrování
      - založen na problému faktorizace velkých čísel
      - faktorizace - problém rozložení čísla na součin menších čísel, v nejčastější podobě pak rozklad celého čísla na součin prvočísel (např. číslo 15 na 3×5)
    - Diffie-Hellman - systém pro výměnu kryptografických klíčů mezi dvěma stranam
      - nejedná se vlastně o šifrovací algoritmus, ale o metodu pro vyvinutí a výměnu sdíleného privátního klíče přes veřejné komunikační kanály
      - v zásadě se obě strany dohodnou na nějaké společné číselné hodnotě a pak vytvoří klíč



Princip fungování asymetrické šifry

- **Hybridní**
  - spojuje výhody obou předchozích řešení (symetrické a asymetrické šifrování)
  - eliminuje tak jejich problémy

    - u symetrického šifrování - problém s přenosem klíče pro šifrování a dešifrování
    - u asymetrického šifrování - náročnost na výpočetní výkon
    - Princip
        - nejprve náhodně vygenerujeme klíč pro symetrickou šifru a zašifrujeme jím zprávu
        - poté klíč samotný zašifruje asymetricky
        - poté odešleme zašifrovaný klíč spolu se šifrovanou zprávou příjemci
        - ten si pomocí asymetrické šifry dešifruje klíč
        - pak pomocí klíče k symetrické šifře dešifruje i samotnou zprávu
- bezpečnost systému je závislá na bezpečnosti obou použitých šifer
- principy hybridního šifrování využívá např. **HTTPS** *(viz dále)*

Využití šifrování v praxi

- **HTTPS**
    - nadstavba síťového protokolu HTTP
    - umožňuje zabezpečit spojení mezi webovým prohlížečem a webovým serverem před odposloucháváním, podvržením dat a umožňuje též ověřit identitu protistrany
    - používá protokol HTTP, přenášená data jsou šifrována pomocí SSL nebo TLS, standardní port na straně serveru je **443**
    - využívá **asymetrické** šifrování (pro předání symetrického klíče) i **symetrické** šifrování (pro šifrování komunikace)
    - **Princip**
        - obě strany si před zahájením komunikace vygenerují privátní a veřejný klíč.
        - při zahájení komunikace si vymění veřejné klíče (tyto klíče by měly obě strany ověřit pomocí jiného komunikačního kanálu)
        - ověření může proběhnout kontrolou *výtahu* (otisk, hash, miniatura) veřejného klíče u protistrany
            - např. pomocí telefonu, nebo lze použít princip přenosu důvěry - protistrana předává veřejný klíč, který je digitálně podepsaný (nejlépe důvěryhodnou certifikační autoritou - VeriSign, GeoTrust, RapidSSL, ...). Digitální certifikáty jsou základním kamenem zabezpečení poskytovaného protokoly SSL/TLS.

- Šifrování je také velice důležité pro fungování **digitálního podpisu**. Digitální podpis používá pro své fungování **asymetrické šifrování**.

Zásada důvěrnosti

- Vyjadřuje potřebu uložit data tak, aby jejich obsah mohl přečíst jen ten, komu jsou určena.

Šifrování z historického pohledu

V průběhu historie se šifrování rozvíjelo, stejně tak se uplatňovaly stále složitější šifry.

- **Substituční šifry**
    - Spočívá v nahrazení každého znaku zprávy jiným znakem podle nějakého pravidla
    - **Posun písmen**
        - Každé písmeno tajné zprávy je posunuto v abecedě o pevný počet pozic. Šifra je z dnešního pohledu velmi snadno rozluštitelná, protože je jen málo možných klíčů. Ve své době ale představovala nevídanou metodu a osvědčila se velmi dobře
        - např. Caesarova šifra
    - **Tabulka záměn**
        - Šifrování pomocí tabulky záměny, které je založeno na záměně znaku za jiný bez jakékoli vnitřní souvislosti či na základě znalosti klíče

- **Steganografie**
    - Neboli ukrývání zprávy jako takové. Sem patří různé neviditelné inkousty, vyrývání zprávy do dřevěné tabulky, která se zalije voskem apod. V moderní době lze tajné texty ukrývat například do souborů s hudbou či obrázky namísto náhodného šumu.

- **Vigenèrova šifra**
    - Používá heslo, jehož znaky určují posunutí otevřeného textu

- **Vernamova šifra**
    - Jde dosud o jedinou známou šifru, o níž bylo exaktně dokázáno, že je nerozluštitelná. Podobně jako Vigenèrova šifra i tahle spočívá ve sčítání písmen otevřeného textu a hesla

- **Transpoziční šifra**
  - Spočívá ve změně pořadí znaků podle určitého pravidla. Například tak, že otevřený text je zapsán do tabulky po řádcích a šifrový text vznikne čtením sloupců téže tabulky

Způsoby distribuce a uložení šifrovacích klíčů

- **Způsoby distribuce**
  - Problémem symetrického šifrování je předávání šifrovacího klíče. Tento nedostatek řeší asymetrické šifrování. Klíč je třeba předávat zabezpečeným kanálem. Konkrétní řešení závisí na dané situaci, počtu subjektů mezi kterými je třeba klíč přenést (2 účastníci komunikace, 3 účastníci, ...), apod.

- **Uložení šifrovacích klíčů**
  - Doporučení pro uložení privátního klíče:
    - Klíč skladujte na přenosném médiu (USB flash, CD, ...)
    - Chraňte svůj privátní klíč dostatečně silným heslem pro případ, že vám někdo zcizí fyzické médium, kde jej máte uložen
    - Zálohujte klíč na jiná média uložená na bezpečných místech
    - Mějte klíč pro nejhorší scénář i vytištěný na papíře a uložený na bezpečném místě
    - Chraňte záložní klíče fyzicky (trezor) i elektronicky použitím dalších metod zabezpečení (například šifrováním celého záložního media)
    - Nezapomínejte, že elektronická média mají konečnou životnost (u DVD např. jen 2 roky)

Pojmy

- **Autentizace** - ověření identity uživatele, nejčastěji heslem, ale i otiskem prstu, předmětem atd.

- **Bezpečné heslo**
  - je takové, které není snadno zjistitelné, uhodnutelné nebo jinak snadno zneužitelné. Hesla slouží pro ochranu přístupu k nejrůznějším systémům a informacím, do kterých by se neměl dostat nikdo nepovolaný.
  - Nejbezpečnější hesla jsou tedy „nesmyslné“ kombinace znaků
  - Bezpečné heslo by mělo mít minimálně 8 znaků
  - V dobrém hesle by neměly být použité jen běžné znaky
  - Není příliš vhodné používat pouze jedno heslo jako "hlavní"

- **Jednorázové heslo** - používá se, např. když zapomenete heslo. Požádáte server o přidělení jiného, které vám bude odesláno emailem popř. sms. Toho heslo platí jen omezenou dobu.

- **Kryptografie (neboli šifrování)**- nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí
  - Šifrování může hrát významnou úlohu při každodenní komunikaci a práci s počítačem:
    - pomocí šifrování můžeme chránit informace uložené na našem počítači před neautorizovaným přístupem – a to dokonce i před lidmi, kteří jinak mají k našemu počítačovému systému přístup.
    - šifrováním můžeme chránit informace při přenosu z jednoho počítače na druhý.
    - šifrováním můžeme zabránit či detekovat náhodné nebo úmyslné změny dat.
    - pomocí šifrování je možno ověřit, zda autorem dokumentu je opravdu ten, kdo myslíme

- **Šifrovací algoritmus** - funkce sestavená na matematickém základě a provádí samotné šifrování a dešifrování dat.

- **Šifrovací klíč** - říká šifrovacímu algoritmu jak má data (de)šifrovat, podobá se počítačovým heslům, avšak neporovnává se zadaná hodnota s očekávanou, nýbrž se přímo používá a vždy tedy dostaneme nějaký výsledek, jehož správnost závisí právě na zadaném klíči.

- **Délka klíče** - ovlivňuje, kromě jiného, časovou náročnost při útoku hrubou silou – což je kryptoanalytická metoda, kdy postupně zkoušíme všechny možné hodnoty, kterých klíč může nabývat.

- **Šifra** - Kryptografický algoritmus, který převádí čitelnou zprávu neboli prostý text na její nečitelnou podobu neboli šifrový text.

Kódování

- co to je kódování (obecně)
- komprese dat
- samodetekující kód (příklady)
- přenosové kódování (MIME)
- kódové stránky (jednobytové a vícebytové)
- kodek

Kódování

- záznam informace pomocí pevně stanovené znakové tabulky nebo znakové sady (např. melodie je zapsaná v notách)
- kódování často znamená také převod již kódované informace do jiného kódu, jako je například transliterace textu do jiné abecedy nebo převod elektronické informace do jiného kódu či normy
- zvláštní význam má převod „otevřené“ informace do kódu, který je znám jen určitým osobám, čili šifrování (kryptografie)
- převod kódované informace do obecně přístupného tvaru je **dekódování** nebo dešifrování.

Komprese dat

- speciální postup, při ukládání nebo transportu dat
- úkolem komprese dat je zmenšit datový tok
- vhodné např. pro archivaci nebo pro přenos dat přes síť s omezenou rychlostí (snížení doby nutné pro přenos)
- komprese může být také nutná při omezené datové propustnosti (mobilní telefony komprimují hovor pro přenos přes GSM síť)

Lze rozdělit do 2 základních kategorií:

- **komprese ztrátová**
  - některé informace jsou nenávratně ztraceny a nelze je zpět rekonstruovat
  - používá se tam, kde je možné ztrátu některých informací tolerovat a kde nevýhoda určitého zkreslení je bohatě vyvážena velmi výrazným zmenšením souboru
  - používá se pro kompresi zvuku a obrazu, kde si člověk do určité míry chybějících údajů nevšimne nebo si je dokáže domyslet (je využíváno nedokonalostí lidského zraku a sluchu)
  - např. JPEG, MPEG, MP3, ...
- **komprese bezeztrátová**
  - obvykle není tak účinná jako ztrátová komprese
  - komprimovaný soubor lze opačným způsobem rekonstruovat do původní podoby (ale ztráta jediného znaku může znamenat nenávratné poškození souboru)
  - např. FLAC, GIF, PNG, soubory ZIP, RAR, apod.

- **Kompresní poměr** je podíl velikosti původních dat ku velikosti dat komprimovaných
  - Při kompresi 10MB souboru do 2MB souboru je kompresní poměr 5 : 1 (pětkrát zmenšeno), úspora je tedy 80%
  - Kompresní poměr je ovlivněn volbou kompresního algoritmu i typem komprimovaných dat
  - Například nekomprimované skladby na audio CD mají datový tok přibližně 1,35Mb/s, zatímco komprimované zvukové soubory (MP3, AAC) mají datový tok 128Kb/s

Samodetekující kódy

- slouží k jednoduchému ověření správnosti zadaných dat
- princip je založen na tom, že zadaná data jsou (buď rovnou, nebo po určité matematické operaci) dělitelná daným číslem, nejčastěji bývá voleno 11 (tzv. *jedenáctkový samodetekující kód*)
- **daný dělitel by měl splňovat několik kritérií:**
  - dvoucifernost (aby se dala vždy odhalit chyba zadaná v jedné cifře)
  - prvočíselnost

- čím vyšší dělitel se zvolí, tím větší bude pravděpodobnost odhalení chyby (ale o to delší kód)
- příklady
  - EAN-13 (čárový kód)
  - rodná čísla vytvořená od roku 1986 (jsou dělitelná jedenácti - poslední čtyři číslice jsou voleny tak, aby každé rodné číslo tuto vlastnost splňovalo)
  - ISBN (jedinečná identifikace knižního titulu), ISSN (jedinečná identifikace periodických publikací - noviny, časopisy, včetně těch vycházejících online)
  - čísla bankovních účtů a kreditních karet
- **nedostatky**
  - je možné ověřit správnost zadání, při chybě ale není možné zjistit původní informaci
  - tento nedostatek bývá odstraněn pomocí redundance informace u samoopravných kódů

Redundance znamená informační nebo funkční nadbytek, například větší množství informace, prvků nebo zařízení než je nezbytné. Zejména zápisy čísel, kódů a programů mají velmi nízkou redundanci, která se často plánovitě zvyšuje například paritou, kontrolní číslicí nebo kontrolním součtem, které umožňují aspoň odhalení části chyb. Ještě daleko složitější a nákladnější redundance se užívají v podobě tzv. „samoopravného kódování“ (samodetekující kódy), které dovoluje automatickou opravu jedné nebo i více chyb.

Přenosné kódování (MIME)

- internetový standard, jedná se o rozšíření internetové pošty
- umožňuje:
  - zasílat zprávy s diakritikou (podpora textu psaného ve znakových sadách jiných než US-ASCII)
  - podpora příloh (obrázky, zvuky, atp.)
  - vícedílné zprávy
  - využít funkci digitálního podpisu
  - informace v hlavičce v jiné znakové sadě než ASCII
- v současné době používán i dalšími protokoly (např. HTTP)

Kódové stránky (jednobytové a vícebytové)

Kódová stránka (znaková sada nebo také kódování) je kód, který páruje sekvence znaků z dané množiny (abecedy) s jejich jinou reprezentací, jako je sekvence přirozených čísel, bajtů nebo elektrických pulzů, za účelem ukládání textu v počítači nebo přenosu textu telekomunikačními sítěmi. Kódová stránka např. říká, že bajt s číselnou hodnotou **65** má být považován za písmeno velké **A**.

Může se jednat např. o Morseovu abecedu, v prostředí počítačů o znakovou sadu **ASCII**. Kódování je také prostředkem pro kompresi (tj. zmenšení) anebo šifrování (tj. utajování) dat.

- **ASCII** - standard z roku 1963, kóduje 128 znaků americké abecedy (společně s číslicemi a dalšími symboly) jako 7bitová čísla
  - jedná se o základní kódování, z kterého vychází v euro-americkém prostoru ostatní standardy
  - drtivá většina osmibitových (jednobytových) znakových sad pouze rozšiřuje ASCII tím, že přidávají významy kódům 128-255, které se v ASCII nepoužívají

- **8-bitová kódování** - jeden byte odpovídá jednomu konkrétnímu znaku (byte s hodnotou 65 = písmeno velké A)
  - 8-bitová kódování pro češtinu
    - **Windows-1250** (někdy také označován jako CP-1250; na platformě Windows)
    - **ISO 8859-2** (UNIXové systémy)
    - **Kód Kamenických** (kódování částečně kompatibilní s CP437 (zachovává semigrafické znaky))
  - 8-bitová kódování nejsou schopna pojmout znaky některých abeced, které obsahují víc jak 255 znaků (např. čínština, apod.)

- **Vícebytová kódování** - jednomu znaku odpovídá sekvence více bytů
  - takováto kódování (především standard Unicode) se používají, kvůli možnosti uložit a zobrazit písmena všech myslitelných abeced v rámci jedné znakové sady
  - díky vícebytovým kódováním lze např. v rámci jednoho textového souboru přirozeně kombinovat znaky anglické, české a japonské abecedy
  - to je hlavní rozdíl od 8-bitových kódování, u kterých je možné v rámci dokumentu použít jenom omezený počet znaků (max. 255)

- **Unicode** - tabulka znaků všech existujících abeced, která v současnosti obsahuje několik stovek tisíc znak
  - každý znak má jednoznačný číselný kód a svůj název
  - definuje u každého znaku některé základní vlastnosti jako např. zda se jedná o písmeno, symbol atd., zda je písmeno velké či malé atp.
  - prvních 128 znaků (tj. sedmibitové kódy) obsahuje znakovou sadu ASCII
  - Unicode se kóduje různými způsoby - mezi základní metody patří:
    - UTF-32 (každý znak reprezentován přímo 32bitovým číslem)
    - UTF-16 (každý znak reprezentován jedním 16bitovým číslem, znaky, které není možné uložit jako 16bitové číslo, jsou reprezentovány párem 16bitových čísel)
    - UTF-8
- **UTF-8**
  - každý znak má jinou délku (1-4 bajty)
  - prvních 128 znaků (U+0000 – U+007F (0-127)) je zapsáno pomocí 1 bajtu (odpovídají ASCII kódování)
  - jedná se o výchozí kódování pro **XML** dokumenty (v případě, že neurčíme jinou znakovou sadu)
  - **Výhody**
    - používá se hlavně pro přenos dat, neboť je **prostorově úsporné** (znaky běžných písem jsou kódovány krátkými posloupnostmi; nevýhodu mají uživatelé písem Dálného východu)
    - odolné proti chybám
    - zpětně kompatibilní s ASCII
  - **Nevýhody**
    - odlišné délky jednotlivých znaků

Kodek

- zkratka dvou slov **KO**der + **DEK**oder (respektive **ko**mprese, **dek**omprese)
- zařízení nebo počítačový program, který dokáže transformovat datový proud nebo signál
- software, který se používá ke kompresi a dekompresi digitálního mediálního souboru, například skladby nebo filmu. Program Windows Media Player a další programy používají kodeky k přehrávání a vytváření digitálních mediálních souborů
- počítačový program nebo hardwarové zařízení, které kóduje a dekóduje video do/z určitého formátu, zpravidla za účelem zmenšení objemu dat.; takový video proud se pak běžně ukládá do tzv. multimediálního kontejneru, který umožňuje kombinovat různé multimediální datové proudy (audio, video, titulky) do jednoho souboru

- kodeky se dají rozdělit několika způsoby; základním je dělení na *bezeztrátové* a *ztrátové*
  - **bezeztrátové:**
    - *Huffyuv* – využívá Huffmanova kódování, které je založeno na různé četnosti znaků
      - výhodou je rychlá komprese a dekomprese
  - **ztrátové:**
    - *DivX*
      - používá standardní kompresi MPEG-4 ASP, nikoli vlastní formát videa (je kompatibilní s ostatními MPEG-4 ASP kodeky)

Digitální podpis

- zásada neodmítnutelnosti odpovědnosti
- zásada integrity
- kryptografické algoritmy pro digitální podepisování
- princip digitálního podepisování
- PKI, CA, RA
- certifikát veřejného klíče
- způsoby uložení podpisových klíčů

Úloha digitálního podpisu

- Pro plnohodnotnou práci s elektronickými dokumenty je potřeba právně správný a ověřitelný digitální nebo chcete-li elektronický podpis.
- Zajistí ověření vaší totožnosti, nebo totožnosti toho, kdo vám dokument poslal (úřad, firma ...)
- Digitální podpis je velmi složitý, zašifrovaný číselný kód, který je pro každého uživatele ojedinělý obdobně jako otisk prstu, a který je právně ověřitelný.
- Podstata digitálního podpisu spočívá v "označkování" elektronického dokumentu, ze kterého je zřejmá nezpochybnitelná identita autora.
- K podepisování dokumentu slouží privátní, tajné klíče. Ke čtení dokumentu a ověření podpisů slouží veřejné klíče.

Zásady bezpečné elektronické komunikace

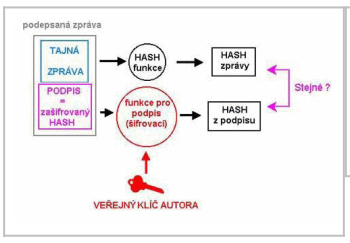
- **Zásada důvěrnosti**
  - Vyjadřuje potřebu uložit data tak, aby jejich obsah mohl přečíst jen ten, komu jsou určena, přičemž kdokoliv další nemá šanci obsah rozluštit ani za pomoci nejmodernějších technologií.
  - Zajišťuje se pomocí šifrování
- **Zásada neodmítnutelnosti odpovědnosti**
  - Vyjadřuje neméně důležitou potřebu možnosti dokázat, kdo je autorem zprávy. Zde nejde o utajení, ale naopak o průkaznost původu dat.
  - Požadavek neodmítnutelnosti odpovědnosti bývá často v praxi splněn digitálním, elektronickým podpisem.
- **Zásada integrity**
  - Má na starosti, aby data došla nejen úplná, ale též prokazatelně nezměněná.
  - Tuto funkci plní HASH

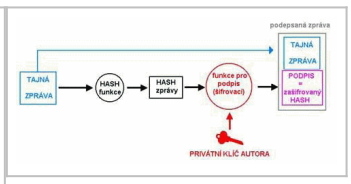
Kryptografie

- Kryptografie je věda, zabývající se šifrováním
  - slouží k ochraně dat před neautorizovaným odhalením
  - může zabránit neautorizovaným modifikacím (změnění) dokumentu

- **Šifrování** je proces, při němž se zpráva (nešifrovaný text) transformuje na jinou zprávu (zašifrovaný text) pomocí matematické funkce a speciálního šifrovacího hesla, tzv. klíče.
- **Dešifrování** je opačný proces: zašifrovaný text se pomocí matematické funkce a klíče převede zpět na text nešifrovaný.

Výtahy zpráv (HASH) a digitální podpisy





- **Výtahy zpráv** (kryptografický kontrolní součet ☐ HASH) není nic jiného, než číslo – speciální číslo, vytvořené nějakou funkcí, která se jen velmi obtížně invertuje (těžko se provádí pozpátku)
  - **Hashovací algoritmy:**
    - MD5 (128 bitový výtah)
    - SHA (160 bitový výtah)
- **Digitální podpis** (digital signature) je nejčastěji výtah zprávy zašifrovaný něčím privátní klíčem. Tomuto procesu se říká podepsání. Digitální podpis plní funkce, které jsou pro bezpečnost systému důležité:

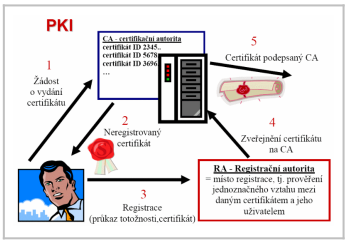
- **Integrita** – digitální podpis indikuje, zda nedošlo k modifikaci souboru nebo zprávy
- **Autentizace** – digitální podpis umožňuje matematicky ověřit, kdo zprávu podepsal
- **Nepopiratelnost** - jakmile zprávu podepíšete a odešlete, nemůžete nikdy v budoucnu tvrdit, že nejste autorem této zprávy. Nemůžete svou zprávu zapřít, protože byla podepsána vaším privátním klíčem, o němž se předpokládá, že jej vlastníte pouze vy.
- **Spočívá v šifrování s veřejným klíčem – provozovaném ovšem opačným směrem**
  - Výtah zprávy zašifrujeme svým privátním (**podpisovým**) klíčem a kdokoli ji pak může rozšifrovat klíčem veřejným ☐ tím je zajištěna autentizace, protože se šifruje naším jedinečným podpisovým klíčem
  - **Použitím privátního klíče a funkce pro výtah zprávy vypočítáme digitální podpis odesílané zprávy**
  - Šifruje se pouze otisk zprávy a přikládá se k zašifrované zprávě, protože použitím privátního klíče k zašifrování trvá poměrně dlouho
  - Když příjemce zašifrovanou hodnotu obdrží, může ji dešifrovat veřejným klíčem. Ze vstupního souboru se rovněž snadno vytvoří hashovaná hodnota.
  - Pokud se obě hodnoty shodují, máte jistotu, že jste obdrželi stejnou zprávu, která byla odesílána.

Kryptografické algoritmy pro digitální podpis

V současné době se pro vytváření digitálních podpisů nejčastěji používají:

- Kombinace algoritmu pro výtah zprávy MD5 a kryptografického mechanismu s veřejným klíčem RSA
- Kombinace algoritmu SHA (Secure Hash Function) a ElGamalova mechanismu veřejného klíče – tyto algoritmy dohromady vytvářejí algoritmus DSA (Digital Signature Algorithm).

Certifikáty



Problémem asymetrické kryptografie je způsob, jak ověřit pravost zveřejněných veřejných klíčů. K tomu slouží digitální či elektronický certifikát.

Digitální certifikát obsahuje:

- Uživatelův veřejný klíč
- Uživatelovi popisné údaje (jméno, adresa ...)
- To vše je zašifrováno privátním (podpisovým klíčem)

Veřejný klíč je veřejně známý a je dostupný z nezaměnitelných zdrojů.

Držitelem privátního klíče je tzv. certifikační autorita (v ČR např. PostSignum), tedy instituce nebo útvar, který tyto certifikáty neboli elektronické občanské průkazy vydává. Každý může požádat certifikační autoritu o digitální certifikát.

Certifikáty mají příponu:

- Pro uložení celého certifikátu: **.P12**
- Uložení jednoho certifikátu bez privátního klíče: **.DER**
- Soubor .DER, akorát v textové podobě v Base64 má příponu **.PEM**

Do zpráv se posílá pouze veřejná část certifikátu, podpisový klíč se neposílá a jeho majitel by ho měl chránit (viz. Uložení certifikátů)

- Řešením problému distribuce a uchování veřejných klíčů je tedy využití služeb certifikační autority (tzv. **PKI - Public Key Infrastructure** neboli Infrastruktura veřejného klíče).
- Instituce **CA** (certifikačních autorit) se podobají státním notářům, kteří při vzájemné komunikaci dvou subjektů vystupují jako třetí nezávislý důvěryhodný subjekt. Prostřednictvím jím vydaného certifikátu jednoznačně svazuje identifikaci subjektu s jeho veřejným klíčem a potažmo tedy i s jím vytvořeným digitálním podpisem.
- Certifikáty obsahují ve své nejjednodušší formě veřejný klíč, jméno a další údaje zajišťující nezaměnitelnost subjektů.
- Běžně používané certifikáty též obsahují datum počátku platnosti, datum ukončení platnosti, jméno certifikační autority, která certifikát vydala, sériové číslo a některé další informace.
- Certifikační autorita garantuje jedinečnost subjektů podle užité identifikace subjektu. To je zajištěno legislativními a technickými pravidly provozu instituce certifikační autority. Splnění těchto požadavků potvrdí certifikační autorita podepsáním dokumentu svým privátním klíčem a následným vydáním tohoto certifikátu.

Uložení certifikátů - podpisových klíčů

- v počítači (nedoporučuje se)
- zakryptován pomocí operačního systému (chráněn heslem)
- Hardwarové řešení
  - Čipové karty (i bezkontaktní, chráněna pinem)
  - USB Tokeny (iKey)

Elektronická pošta

- schéma funkce distribuce zprávy elektronického emailu (včetně role DNS)
- serverový a klientský software
- formát zprávy podle standardu RFC822
- protokoly pro práci s poštou
- MIME
- zabezpečení zprávy elektronického mailu

Elektronická pošta je forma datové komunikace po internetu, označována také jako e-mail nebo SMTP pošta (podle používaného protokolu zajišťující přenos), která není vlastněna žádnou osobou nebo firmou ☐ vychází z plně otevřených standardů (není proprietární).

Elektronická pošta je:

- rychlá (čas doručení v minutách a sekundách, i když může být ovlivněno stavem serveru...)
- levná (záleží na způsobu připojení, používaném softwaru...)
- pohodlná (možnost automatizace některých úkolů - třídění apod. (záleží na softwaru...))
- efektivní (snadná propojitelnost s ostatními aplikacemi, hromadné odesílání zpráv...)
- funguje "off-line" (nevyžaduje současné připojení odesílatele a příjemce)

Historický základ - Standardy

Původní zadání pro koncepci elektronické komunikace znělo asi takto: **Budou se přenášet co nejefektivněji krátké, čistě textové zprávy.** Od toho se odvíjí veškeré protokoly a techniky pro přenos, protože dnes se do tohoto zadání nevejdeme ☐ velké zprávy, nestandardní znakové sady, přílohy...

Koncepce elektronické pošty je dodnes založena na dvou dokumentech:

- RFC821 - definuje přenosový protokol SMTP
- RFC822 - definuje formát zpráv

RFC821 - Přenosový protokol SMTP

Tento dokument definuje přenosový protokol SMTP:

- Přenosový protokol **SMTP** (**S**imple **M**ail **T**ransfer **P**rotocol)
  - Podle tohoto protokolu spolu komunikují jednotlivé poštovní servery (jednotky MTA - Message Transfer Agents), když si mezi sebou předávají jednotlivé zprávy.
  - Spojení probíhá na smluveném **portu 25**
  - Předpokládá, že **přenášená data jsou sedmibitová**
    - Zpráva může obsahovat 128 ASCII znaků (základní sada ASCII)
    - Každý znak je zobrazitelný v sedmi bitech ($2^7 = 128$)
    - Při přenosu osmibitových zpráv není zaručen správný přenos

RFC822 - Definice zpráv

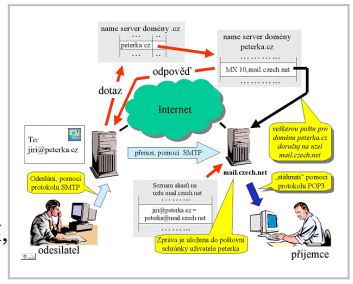
Tento dokument definuje formát zpráv přenášených přes SMTP. Říká že:

- Zpráva **se skládá** z hlavičky a těla
- Definuje typ a přesný tvar (syntaxi i sémantiku) jednotlivých položek v **hlavičce**:
  - **From:** adresy odesilatelů
  - **To:** adresáti
  - Cc: adresy na které se dopis odešle i když nejsou hlavními adresáty
  - Bcc: tajné kopie. Jako cc, ale adresy se vymažou z ostatních mailů
  - **Subject:** předmět
  - Return Path: zpáteční cesta k odesílateli

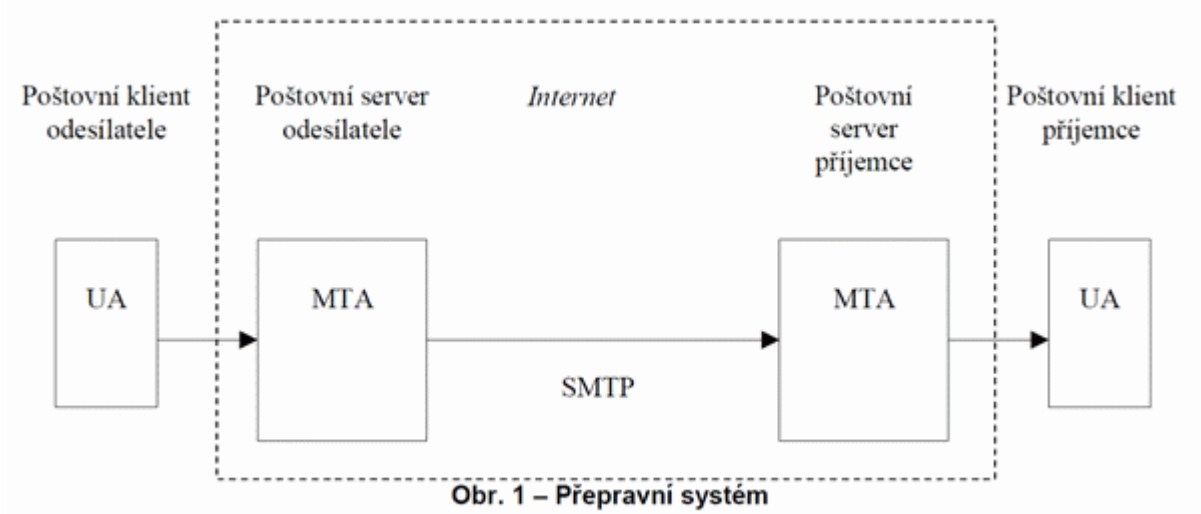
- Received: záznamy přidané během zpracování
- Reply to: adresa pro zaslání odpovědi
- Sender: adresa skutečného odesílatele pokud je jiná než From: nebo je ve From: více adres.
- Message ID: identifikace zprávy
- Resent-klic: při přeposílání se klíče původních hlaviček uvodí Resent-
- X-neco: doplňující hlavička (např. X-priority: 3)
- Mime-version: použitá verze MIME. viz dále

- Říká, co smí a nesmí být v **těle zprávy**
- Definuje přesný **formát adres**, které lze používat pro potřeby elektronické pošty
  - `frantisek@vysmrkmaslo.cz`

Postup při komunikaci



- Uživatel spustí klientský program (**Microsoft Outlook**, **Mozilla Thunderbird**,...) a napíše zprávu
- Zpráva je upravena tak, aby vyhovovala standardu RFC822 a pokud zpráva obsahuje nepovolené znaky (diakritika) nebo přílohy, je na ni aplikován standard **MIME**.
- Poté je zpráva předána serveru pro odchozí poštu pomocí protokolu SMTP. Klientský program zde vystupuje jako SMTP klient.
  - Jako poštovní servery se používají např.: **Sendmail, Postfix**
- Zpráva je na serveru zařazena do fronty zpráv. Server se je postupně pokouší odeslat: (pokud se nezdaří, zpráva jde zpátky do fronty, v případě vypršení limitu je zpráva označena jako nedoručitelná...)
  - Nejprve se podívá na část adresy vpravo od zavináče (seznam.cz)
  - Snaží se ptát systému DNS kam má být doručena pošta pro "seznam.cz"
  - Odpověď mu může dát pouze cílový name server (seznam.cz), (cesta k němu může vést postupně nejdřív přes name server pro CZ doménu)
  - V name serveru Seznam.cz bude tzv. **MX** záznam (Mail Exchange), který nám řekne, kam tuto zprávu doručit
    - může existovat druhý MX záznam, který určuje adresu záložního serveru příjemce - tento server je použit v případě, kdy je primární server nedostupný
  - Odesílací server tedy naváže spojení se serverem příjemce a zprávu mu pošle pomocí SMTP protokolu (server odesílatele je zde jako SMTP klient)
- Podle části adresy vlevo od zavináče uloží server příjemce zprávu k příslušnému uživateli
- Příjemce si poté zprávu vyzvedne pomocí svého e-mailového klienta (klientského programu).
- E-mailový klient stahuje zprávy ze serveru pomocí protokolu **POP3**, nebo **IMAP**.



Obr. 1 – Přepravní systém

Problémy e-mailu

Mezi hlavní problém e-mailové komunikace patří především nevyžádaná pošta - **SPAM**. Jako obrana před SPAMem se nasazuje na cestě od serveru příjemce do e-mailového klienta příjemce tzv. antispamová ochrana. Jedním ze zástupců takové antispamové ochrany je např. SPAM Assassin.

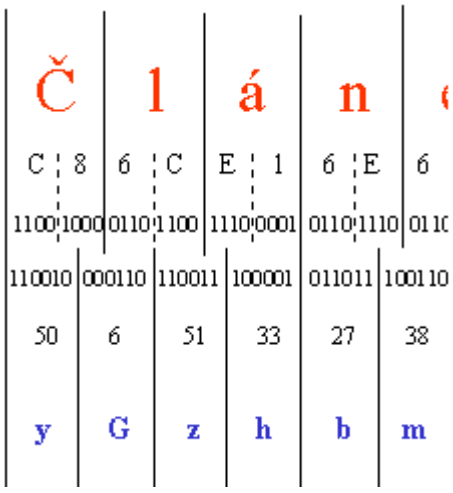
Protokoly pro stahování zpráv

Pro stahování příchozích zpráv z poštovního serveru se používají dva protokoly:

- **POP3**
  - Dávkově se stahuje pošta do PC, nezůstává na serveru
- **IMAP4**
  - Emaily zůstávají na serveru, pouze se stáhnou, když si je chceme prohlédnout

MIME

- Multipurpose Internet Mail Extension
- Umožňuje posílat nestandardní ASCII znaky (8-bitové - s ASCII kódem 128-255) a umožňuje posílat přílohy
- Pomocí MIME se 8-bitové znaky překódují do 7-bitového vyjádření, aby šli odeslat přes SMTP
- MIME je především záležitostí e-mailových klientů - ti mají za úkol zprávu používající MIME při odeslání zakódovat, při přijetí dekódovat
- Ukázka zprávy, na kterou bylo použito MIME ☐ viz. obrázek ☐
- Metody pro překódování nestandardních znaků do 7-bit. vyjádření:
  - **Quated Printable**
    - použití tam, kde se text příliš neliší od čistého ASCII textu
    - převod nestandardních znaků na základní ASCII znaky v 7-bitovém vyjádření
    - např.: Č ☐ "=C8" (kód znaku v šestnáctkové soustavě)
    - zakódovaný text je stále pro člověka čitelný (při malém výskytu cizích znaků)
  - **Base64**
    - použití při větší odlišnosti textu od klasických ASCII znaků
    - určeno především pro obecná binární data
    - kódovaná data jsou o třetinu delší než originální text
    - pro člověka zcela nesrozumitelný text
    - text se rozdělí na bity a ty se pak po šesti useknou a vytvoří standardní ASCII znak ☐ viz. obrázek



Princip fungování Base64

Zabezpečení zprávy

Pro zabezpečení elektronických zpráv se využívá šifrování pomocí klíčů, které zajistí, že zprávu přečtou jen povolané osoby, a digitální podpisy pro ověření identity odesílatele.

Šifrování

- nebo-li moderní kryptografie
- dělí se na:
  - **Symetrické šifrování**
    - použití jednoho (privátního) klíče

    - privátním klíčem se šifruje i dešifruje
    - rychlejší, ale nemožnost bezpečného předání klíče
    - Přehled šifrovacích algoritmů:
      - DES
        - klíč o 56 bitech
        - používá se i víckrát za sebou (Double DES, Triple DES…)
      - RC
      - IDEA
  - **Asymetrické šifrování**
    - používá dva klíče - veřejný a privátní
    - předává se jen veřejný klíč, kterým druhá strana zprávu zašifruje a příjemce ho pak svým privátním klíčem dešifruje
    - šifruje se veřejným klíčem příjemce, přečíst zprávu jde jen pomocí příjemcova privátního klíče
    - algoritmus RSA - klíč s libovolnou délkou, používá se i jako základ pro digitální podpisy

- díky svým vlastnostem se v případě el. pošty spíše používá **asymetrické** šifrování, protože je výhodnější

Podepisování - HASH

- HASH = otisk
- díky HASHi máme jistotu, že zpráva nebyla změněna
- HASH je obsažen v digitálním podpisu, který je zašifrován privátním (podpisovým klíčem)
- V podpisu je Veřejný klíč odesílatele, zašifrovaný HASH a adresa odesílatele
  - Obsahuje také podpis certifikační autority
  - Celý certifikát obsahuje jak privátní tak veřejný klíč, ale posílá se pouze veřejná část !
  - Certifikáty mají příponu .P12

- **Hashovací algoritmy:**
  - MD5 (128 bitový výtah)
  - SHA (160 bitový výtah)

- Uložení certifikátu
  - v počítači (nedoporučuje se)
  - zakryptován pomocí operačního systému
  - Šifrovací token (iKey)
  - Čipová karta

DNS a doménová jména

- co to je DNS
- jaký port používá DNS
- jak zjistí počítač s OS Windows, jaká konkrétní adresa odpovídá adrese vyjádřené doménovým jménem (konkrétně popište konfiguraci počítače)
- kořenové DNS servery
- zónové soubory a typy DNS záznamů
- princip tvorby doménového jména (TLD, domény 2. úrovně)
- URL
- registrátoři

DNS (Domain Name System)

- je to hierarchický systém (strom) doménových jmen, který je realizován **DNS servery** (name servery) a **DNS protokolem**
- umožňuje dát číselné **IP adrese** tzv. **doménové jméno**, které si uživatelé snadněji zapamatují (147.230.16.108 = www.cesnet.cz).
- v minulosti i současnosti je masově rozšířeno IPv4 (xxx.xxx.xxx.xxx), blízká budoucnost patří IPv6 (2001:200:8002:203:47ff:fea5:3085)
- **Struktura**
  - kořenem systému (stromu) je **kořenová domény** (samotná tečka **.**)
  - pod kořenovou doménou se nachází domény nejvyšší úrovně (**TDL** = top domain level), tj. domény států (.cz, .sk, .jo) a domény obecné (tématické) (.com, .org, .net,...)
  - pod TLD se nachází jednotlivé domény 2. až N-tého řádu
- **Software**
  - existuje množství implementací DNS serverů pro Linux, nejznámější je **BIND**. Je konstruován pro obrovské zatížení, a proto se s ním setkáme u všech velkých DNS serverů. Umožňuje kešování.

Jaký používá DNS port

- (2) používá protokol **TCP**, nebo **UDP** (= nespojový protokol, jednoduché rozhraní *(interface)* mezi IP protokolem a protokoly vyšší vrstvy; UDP používán hlavně kvůli rychlosti) a port **53** (na obou protokolech).

Princip zjištění IP adresy serveru; kořenové servery

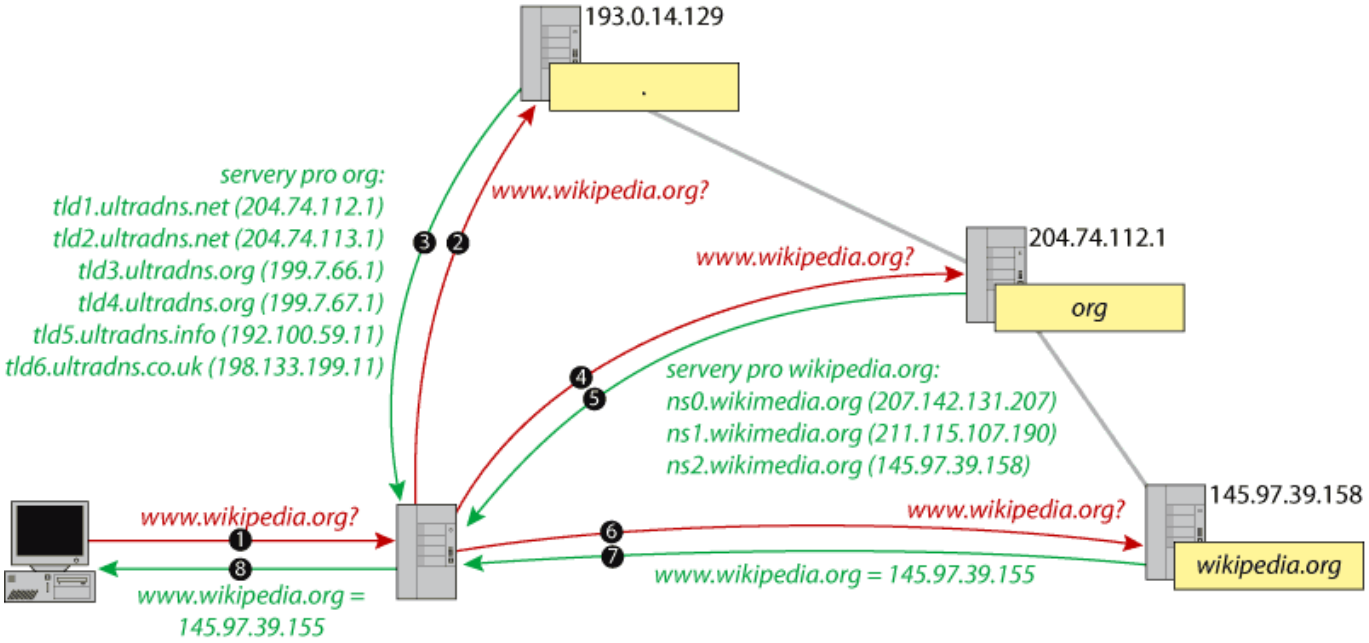
- (3-4) Kořenový DNS server je první z uzlů stromu DNS serverů. Jde o servery, které obsluhují Top-Level domény. Top-Level domény přiděluje organizace Internic , konfigurace kořenových DNS serverů je tedy v její kompetenci. Tyto servery převezmou dotaz, podívají se na něj a odpoví "nevím, ale tuto top-level doménu spravuje ten a ten DNS server, zeptej se tam". Tato odpověď se vrátí DNS serveru, kterého se ptal klient, a ten znovu pošle dotaz na klientem zadanou doménu, tentokrát ovšem nikoliv kořenovému DNS serveru, ale DNS serveru který dostal v odpovědi od kořenového DNS serveru. Tento buď odpoví IP adresou stroje, jemuž je přiřazena doména, nebo zase pošle adresu DNS serveru pro nižší doménu, a tak se klientův DNS server ptá, až se dobere poslední domény, na niž by měl jako odpověď dostat IP adresu stroje. Tuto adresu pak pošle klientovi jako odpověď.

Příklad: Podívejme se, jak by postupovalo hledání IP adresy ke jménu www.wikipedia.org:

1. Uživatel zadal do svého WWW klienta doménové jméno www.wikipedia.org. Resolver v počítači se obrátil na lokální DNS server s dotazem na IP adresu pro www.wikipedia.org.
2. Lokální DNS server tuto informaci nezná. Má však k dispozici adresy kořenových serverů. Na jeden z nich se obrátí (řekněme na 193.0.14.129) a dotaz mu přepošle.
3. Kořenový server také nezná odpověď. Ví však, že existuje doména nejvyšší úrovně org, a jaké jsou její autoritativní servery, jejichž adresy tazateli poskytne.
4. Lokální server jeden z nich vybere (řekněme, že zvolí tld1.ultradns.net s IP adresou 204.74.112.1) a pošle mu dotaz na IP adresu ke jménu www.wikipedia.org.
5. Oslovený server informaci opět nezná, ale poskytne IP adresy autoritativních serverů pro doménu wikipedia.org. Jsou to ns0.wikimedia.org (207.142.131.207), ns1.wikimedia.org (211.115.107.190) a ns2.wikimedia.org (145.97.39.158).
6. Lokální server opět jeden z nich vybere a pošle mu dotaz na IP adresu ke jménu www.wikipedia.org.
7. Jelikož toto jméno se již nachází v doméně wikipedia.org, dostane od jejího serveru nepochybně autoritativní odpověď, že hledaná IP

adresa zní 145.97.39.155

8. Lokální DNS server tuto odpověď předá uživatelskému počítači, který se na ni ptal.



Zónové soubory a DNS záznamy

Obsah zóny (domény či několika domén) je uložen v tzv. **zónovém souboru**. Skládá se z jednotlivých **záznamů** (přesný název zní zdrojové záznamy, resource records, RR) obsahujících dílčí informace. Jejich názvy a nesené informace jsou přesně definovány v příslušných dokumentech, většina v RFC 1035. Formát textového zápisu zónového souboru se liší v závislosti na použitém serveru.

- (5) Typy záznamů:
  - **SOA** = speciální typ záznamu, který se musí v zónovém souboru vyskytovat **pouze jednou**; musí obsahovat informace
    - **MNAME** (název primárního DNS serveru pro danou zónu)
    - **RNAME** (kontakt na správce souboru)
    - **SERIAL** (sériové číslo zóny)
    - **REFRESH** (počet sekund pro následující kontrolu zóny)
    - a několik dalších
  - **A** = IP adresa IPv4 protokolu, v protokolu jsou předávány jako 32 bit. číslo
  - **AAAA** = IP adresa protokolu IPv6
  - **CNAME** = oznamuje, že daná adresa je aliasem jiné domény
  - **NS** = sděluje seznam autoritativních DNS serverů pro danou doménu
  - **MX** = doménový název mailserveru, na který se má doručovat pošta pro tuto doménu
  - **PTR** = rezervní záznamy

TLD, tvorba doménového jména

- (6) Doménové jméno vzniká až na základě smlouvy o registraci (předtím právně neexistuje a není technicky možné jej užívat)
  - Smlouva o registraci je soukromoprávní úkon, jímž vzniká registrátorovi povinnost doménové jméno registrovat (umožnit jeho používání) a žadateli vzniká právo toto jméno užívat a povinnost platit příslušné poplatky. Právní režim smlouvy je patrně podobný jako režim smlouvy o dílo (spotřebitelská smlouvy?). Ohledně samotného právního posouzení způsobu uzavírání stávající smlouvy CZ.NIC o registraci doménového jména a jejího obsahu panují určité pochybnosti
- **Doména nejvyššího řádu** (TLD - Top Level Domain) je internetová doména na nejvyšší úrovni stromu internetových domén. V doménovém jméně je doména nejvyšší úrovně uvedena na konci (např. u cs.wikipedia.org je doménou nejvyššího řádu "org"). TLD popisuje základní skupinu doménových jmen, např. všechna doménová jména daného státu. Domény nejvyššího řádu jsou pevně stanoveny internetovou standardizační organizací IANA.

Existují TLD následujících **tří typů**:

- **Národní TLD** (country-code TLD, ccTLD) sdružující **domény** jednoho **státu**. Jejich název je dvoupísmenný (až na výjimky) odpovídající kódu země podle ISO 3166-1, např. "cz" pro Českou republiku.
- **Generické TLD** (generic TLD, gTLD) sdružující **obecné domény** (např. "org" pro neziskové organizace), nespojené s jedním konkrétním státem (až na výjimku TLD "mil" a "gov", které jsou z historických důvodů vyhrazeny pro vojenské, resp. vládní počítačové sítě v USA).
- **Infrastrukturní TLD** využívané **pro vnitřní mechanismy Internetu**. V současné době existuje jediná taková TLD: arpa, používaná systémem DNS.

- **Doménový alias** znamená, že na tentýž server vede více domén. Dá se tak využívat zároveň například doména mojefirma.cz a mojefirma.com, které odkazují na stejný prostor pro stránky na serveru.

URL (Uniform Resource Locator = „jednotný lokátor zdrojů“)

- (7) je to řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu.
  - URL definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, kterým je možné zdroj zpřístupnit.
  - Jednotlivá pole v URL: schéma, doménové jméno, port, specifikace souboru, parametry
  - Některá pole jsou nepovinná – buď nemají význam, nebo se předpokládá předdefinovaná hodnota, závislá např. na schématu (např. pro protokol HTTP je implicitní port 80), nebo na aplikaci (pro webový prohlížeč se předpokládá protokol HTTP).
  - Pomocí URL lze zadat také autentizační informace: mezi protokol a doménové jméno lze vložit uživatelské jméno a případně i heslo navzájem oddělené dvojtečkou a od domény odděleny zavináčem, např.: http://mirek:mojetajneheslo@www.example.com/

Registrátoři

- (8) Správce domény (CZ.NIC) nenabízí registraci jednotlivých domén přímo koncovým zájemcům. Ti si domény registrují prostřednictvím akreditovaných registrátorů, což jsou komerční subjekty nabízející služby registrace a správy domén svým zákazníkům. Tento model přináší konkurenční prostředí a variabilitu nabízených služeb (přidaná hodnota, dotování cen domén při využívání jiných služeb registrátora apod.).
  - Registrátorem se může stát právnická osoba se sídlem v některém z členských států Evropské unie či akreditovaný registrátor ICANN. Registrátorství se však vzhledem k některým poplatkům a nákladům na technický provoz vyplatí až od určitého počtu domén, proto registrátorů příliš mnoho není (k červenci 2009 jich je 37).
  - Na začátku září 2011 představil CZ.NIC projekt s názvem Certifikace registrátorů s cílem poskytnout zájemcům o domény základní přehled o úrovni služeb jednotlivých registrátorů a definovat ideální vlastnosti a funkce registračního systému a souvisejících služeb. V seznamu registrátorů tak teď mohou zájemci najít vedle návodů jak registrovat a symbolů mojeID, IPv6 a DNSSEC také hvězdičky ukazující, jak který registrátor v certifikaci uspěl.

Informační systém

- informace a data
- metainformace
- co to je informatika, odvětví informatiky
- informační gramotnost (funkční, počítačová)
- co to je informační systém
- příklady informačních systémů (podle oblasti nebo oboru použití)
- kde se ukládají data IS
- outsourcing informačních služeb, ASP
- architektura IS

Informace a data

- **Informacemi** míníme sdělení, které odstraňuje nejistotu nebo nevědomost. Informaci je možno také chápat jako data s nějakým přidaným významem (data + význam).
- **Daty** míníme jakékoli zaznamenané poznatky či fakta. Data jsou v podstatě prostředníky k přenosu a uložení informace.

Metainformace

- **Metainformace** jsou dílčí informace, které popisují atributy informačního zdroje.
- **Metadata** jsou strukturovaná data o datech.
  - Příkladem je katalogizační lístek v knihovně, obsahující data o původu a umístění knihy:
    - jsou to data o datech v knize uložené na lístku. Metadata mohou sloužit např. k snadnému vyhledávání.
  - U mp3 jde o ID3 tag, v kterém je uložen interpret, album, rok....
  - Fotografie pořizované digitálním fotoaparátem obvykle obsahují metadata ve formátu Exif. V těchto metadatech jsou uloženy informace o vzniku fotografie – datum a čas pořízení, použitá ohnisková vzdálenost, použití blesku, typ a výrobce fotoaparátu apod....

Informatika a její odvětví

- **Informatika** studuje výpočetní a informační procesy z hlediska hardware i software. V praxi se vztahuje k počítačům a od abstraktní analýzy algoritmů, formálních jazyků atd. pokračuje ke konkrétnějším tématům, jakými jsou programovací jazyky, software a hardware.
- **Odvětví informatiky** - *Vypustit. Pohovořit o tom, čeho všeho se informatika dotýká (otázka programování, grafiky, tvorby dokumentů, co bysme mohli po škole dělat - programátor, grafik, správce sítě, apod.).*

Informační gramotnost

- **gramotnost funkční** (schopnost zpracovávat informace – čtenářská, numerická, atd...)
- **počítačová gramotnost** (práce s počítači)

Informační systém

- **Informační systém** (IS) je soubor technologických prostředků a metod, které zabezpečují sběr,přenos, zpracování a uchování dat za účelem tvorby prezentace informací pro potřeby uživatelů. Nedílnou součástí IS jsou jeho uživatelé.
- Příkladem informačního systému může být kartotéka, telefonní seznam, kniha došlé pošty nebo účetnictví. Systém nemusí být nutně automatizovaný pomocí počítačů a může být i v papírové podobě.

Příklady (rozdělení) informačních systémů

- **Řídící a manažerské** (Business intelligence - systémy pro podporu rozhodování)
- **Taktické** (kontrolní, auditové, marketingové, propagační)
- **Konstrukční** (CAD)
- **Kancelářské a administrativní** (Office IS - kancelářské balíky, jednoduché databáze, poštovní programy)

- **Řízení vztahu se zákazníky a obchodní systémy** (Customer Relationship Management a Bussines To Bussines – B2B)
- **Řízení výroby** (Enterprise Resource Planning - výroba, logistika, distribuce, správa majetku, prodej, fakturace a účetnictví)
- **Systémy správy obsahu** (Content Management Systém nebo také Document Management System - správa dokumentů, redakční systémy, publikační systémy)
- **Bankovní systémy**
  - Skládají se ze systémů Front-Office: důležitý prvek při komunikaci s klientem, setkáváme se s nimi na bankovních přepážkách např. při uzavírání smlouvy, při provádění přímých hotovostních operacích či při sjednávání nových finančních produktů.
  - Dále ze systémů Back-Office: nejsou pro běžného člověka viditelné, neboť přímo nesouvisí s operacemi prováděnými při jednání s klientem. Jsou to interní systémy, které úzce spolupracují s přepážkovými aplikacemi. Zpracovávají data a evidují je, provádějí statistiky a generují příslušné reporty
  - A z podpůrných systémů: moderní komunikační kanály - informační systémy pro řízení přímého bankovnictví (e-banking, gsm-banking)
- **Geografické systémy** (Geographic Information System - pro získávání, ukládání, analýzu a vizualizaci dat, která mají prostorový vztah k povrchu Země)
- **Personální** (Human Resources - plánování lidských zdrojů, získávání a výběr zaměstnanců, vzdělávání a rozvoj zaměstnanců, řízení kariéry, odměňování a zaměstnanecké výhody (benefity), personální administrativa, výkaznictví, mzdové účetnictví)
- **Docházkové** (evidence docházky, stravovací systémy, kontrola vstupů osob či vozidel, evidence výrobních operací, kontrola obchůzky)
- **Knihovnické**
- **IS státní správy**
- **IS škol**

Kde se ukládají data IS?

- Informační systémy jsou většinou realizovány pomocí třívrstvé architektury - jsou složeny z prezentační vrstvy (webový prohlížeč), vrstvy logické (vlastní logika aplikace) a vrstvy datové.
- Pro ukládání dat se tak většinou používá vrstva datová - nejčastěji v podobě **databáze**.

Outsourcing informačních služeb, ASP

- **Outsourcing** znamená uskutečňování činností prostřednictvím jiných subjektů (firem). V oblasti IT se tedy například jedná o dodávku a pravidelnou obnovu hardware či o služby v oblasti správy počítačové sítě. Může se jednat i o správu webových aplikací či správu software informačního systému.

- **ASP** je ve volném překladu dodavatelský model, ve kterém uživatel-zákazník řeší potřeby automatizace svých činností-procesů prostřednictvím pronájmu aplikací jako služby od ASP operátora.
  - Služba je zpravidla nějakou formou zpoplatněna. Existují i služby poskytované zákazníkům zdarma, ty pak vydělávají nepřímo - zpravidla na reklamě.
  - Typickým představitelem ASP aplikací jsou opakovaně uplatnitelné, tedy do značné míry univerzální aplikace nikoli unikátní aplikace vyvinuté "na míru" pro konkrétního zákazníka (v tom případě se jedná o formu outsourcingu).

Architektura IS

- V současnosti převažuje tzv. **třívrstvá architektura**:
  - presentační (interakce s uživatelem)
  - funkční (vlastní aplikace, bezpečnost, propojení se světem, kontrola, atd..)
  - datová (vlastní data)

E-Government, E-Learning

- **e-Government** se zabývá elektronizací výkonu veřejné správy. Jedná se o transformaci vnitřních a vnějších vztahů veřejné správy pomocí informačních a komunikačních technologií. V České republice je plně v režii Ministerstva vnitra ČR. České elektronické projekty:
  - **Czech POINT** je český státní projekt, v jehož rámci obecní úřady s rozšířenou působností, krajské úřady, notáři a další právnické osoby (např. provozovny České pošty a lokální pracoviště Hospodářské komory ČR s příslušným oprávněním) mohou lidem vydávat výpisy z katastru nemovitostí, z rejstříku trestů či živnostenského rejstříku. Na Czech POINTech lidé získávají veškeré údaje, opisy a výpisy, které jsou vedeny v centrálních veřejných evidencích a registrech o jejich osobě, majetku a právech. Odpadává tak další obíhání po

úřadech dle hesla „nemá obíhat občan, ale dokument“.

- **Datová schránka** je v českém právním řádu od roku 2009 definována jako elektronické úložiště speciálního typu, které je určeno k doručování elektronických dokumentů mezi orgány veřejné moci na straně jedné a fyzickými a právnickými osobami na straně druhé. Datová schránka funguje na podobném principu jako běžná e-mailová schránka, ale systém odesílání a doručování je samostatný a s běžným internetovým e-mailovým provozem není funkčně propojen.

- **E-Learning** je vzdělávací proces, využívající informační a komunikační technologie k tvorbě kursů, k distribuci studijního obsahu, komunikaci mezi studenty a pedagogy a k řízení studia. Je to forma vzdělávání využívající multimediální prvky - prezentace a texty s odkazy, animované sekvence, video snímky, sdílené pracovní plochy, komunikaci s lektorem a spolužáky, testy, elektronické modely procesů, atd. v systému pro řízení studia (Learning Management System).

--Trnka.vaclav 24. 4. 2012, 21:52 (CEST)

Programovací jazyky

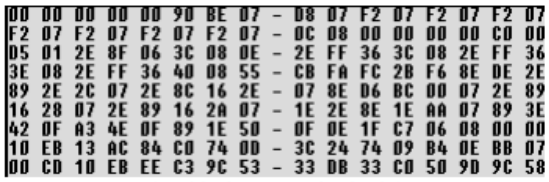
- co to je program, zdrojový kód
- co to je algoritmus, způsoby zápisu algoritmů, vlastnosti algoritmů
- rozdělení prg. jazyků: imperativní (procedurální) a deklarativní (neprocedurální)
- rozdělení prg. jazyků: kompilující a interpretující
- JAVA platforma
- událostmi řízené programování
- syntaxe a sémantika
- vysvětlit předložený zdrojový text v C#

Program

- **Je v informatice postup operací, který popisuje realizaci dané úlohy**
- Zápis algoritmů pomocí příkazů určitého programovacího jazyka
- **Počítačový program** (též jen program, obecně pak software) je v informatice **posloupnost instrukcí (ne nutně strojových instrukcí), která popisuje realizaci dané úlohy počítačem.** Aby počítač mohl vykonávat nějakou činnost, potřebuje mít ve své operační paměti alespoň jeden program

Zdrojový kód

- Zdrojový kód nebo zdrojový text je v informatice **označení zápisu textu počítačového programu v některém programovacím jazyce, který je uložen v jednom nebo více textových souborech.**
- Zdrojový kód obvykle programátor zapisuje pomocí textového editoru, ale může být též generován specializovaným programem.
- Textový editor může být součástí integrovaného vývojového prostředí (IDE), které programátorovi tvorbu zdrojového kódu usnadňuje a poskytuje mu další podporu:
  - zvýraznění syntaxe
  - vyznačení syntaktických chyb
  - nápověda
  - seznamy funkcí
  - příklady
  - generování zdrojového kódu
  - přímý přístup k navazujícím nástrojům (vyvolání kompilátoru, možnost krokování a sledování průběhu programu pomocí debuggeru, vytváření souborů pro řízení překladu - Makefile, zpracování dokumentace a podobně).....
- **Strojový kód** je v informatice posloupnost strojových instrukcí prováděných procesorem počítače. Strojové instrukce jsou zapsány formou číselných kódů.
  - Obecně řečeno strojový kód jsou příkazy napsané tak aby jim rozuměl procesor a mohl požadovanou akci vykonat. Strojový kod je ovšem pro člověka velmi nepřehledný, matoucí a složitý, neboť je to jen posloupnost číslic.



(Příklad strojového kódu)

Z tohoto důvodu se strojový kód k programování používá jen zřídka.

Instrukce zobrazené jako čísla v šestnáckové soustavě.

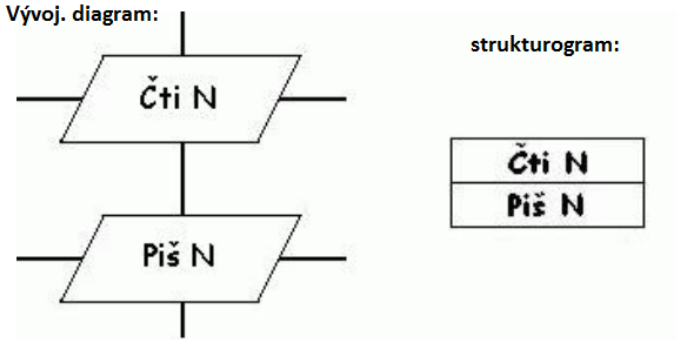
Algoritmus

- je přesný návod či postup, kterým lze vyřešit daný typ úlohy. - jinak řečeno **je to posloupnost operací, která řeší daný úkol.**
- Myslí se jím teoretický princip řešení problému, oproti přesnému zápisu v konkrétním programovacím jazyce. Obecně se ale algoritmus může objevit v jakémkoli jiném odvětví. Jako jistý druh algoritmu se může chápat i např. kuchařský recept.
- **Vlastnosti algoritmu**
  - **konečnost - každý algoritmus musí skončit v konečném počtu kroků.** Tento počet kroků může být libovolně velký (podle rozsahu a hodnot vstupních údajů), ale **pro každý jednotlivý vstup musí být konečný.**

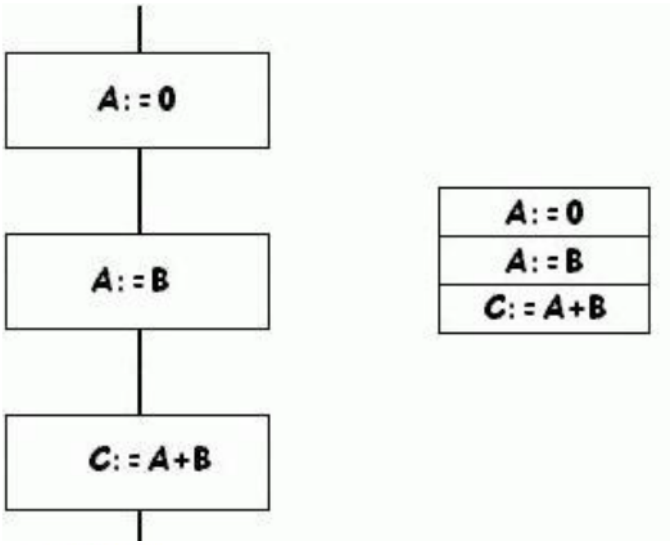
- **obecnost (hromadnost, masovost, univerzálnost)** - algoritmus **neřeší jeden konkrétní problém, musí řešit danou úlohu pro různé vstupní hodnoty**
- **determinovanost** - každý krok algoritmu musí být **jednoznačně a přesně definován**. V každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat.
- **opakovatelnost** - pro stejné vstupy dostaneme pokaždé stejné výsledky.

- **Způsoby zápisu algoritmu**
  - **slovní vyjádření**
  - **grafické vyjádření**
    - ->**Strukturogram** je algoritmus přepsaný do tabulkového způsobu seřazení příkazů pod sebou v návaznosti tak, aby se po přepsání do programovacího jazyka dosáhnul požadovaný výsledek.
    - ->**vývojový diagram** je postup řešení určité úlohy lze zapsat pomocí vývojového diagramu. Ten se skládá ze značek, do kterých se zapisují jednotlivé příkazy při postupu řešení určitého úkolu. Tento postup řešení se nazývá algoritmus.

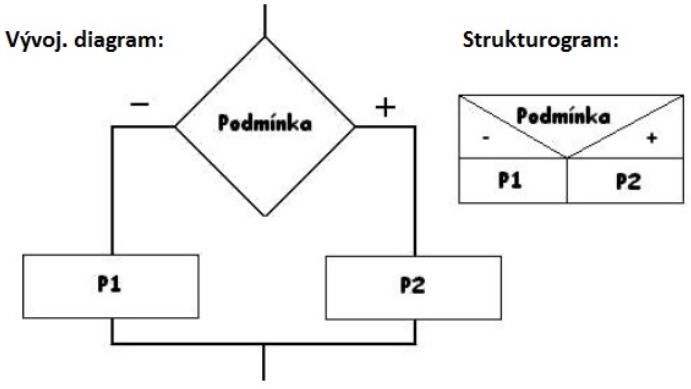
1. Vstup - výstup dat



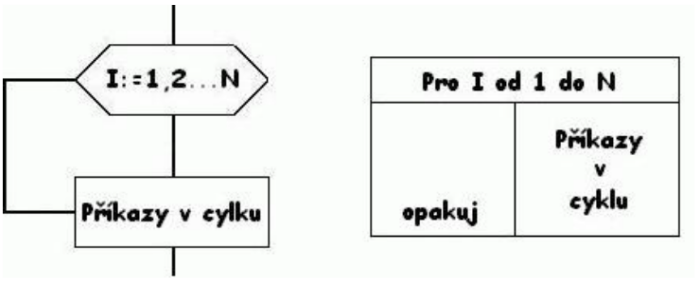
2. Přiřazení



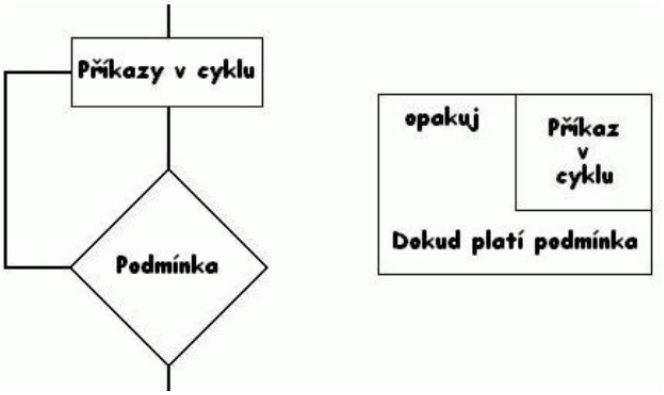
3. Rozhodování



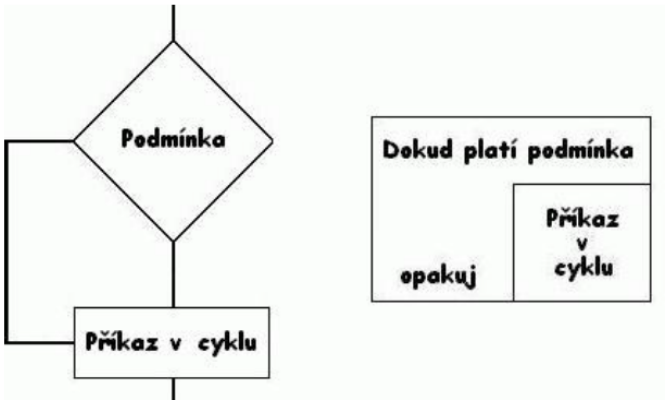
4. Cyklus řízený indexem



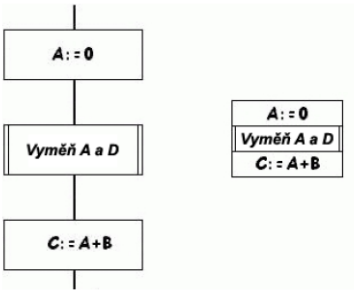
5. Cyklus s podmínkou na konci



6. Cyklus s podmínkou na začátku



7. Procedura (podprogram)
Příklad: zavolání externí části algoritmu, která zajistí výměnu obsahu dvou proměnných A a D)



Rozdělení programovacích jazyků

Jazyky se dělí na:

- **Imperativní (procedurální)**
  - Program se zapisuje v podobě programových struktur, používá proměnné a datové struktury. Program je složen z příkazů, které krok po kroku vycházejí z algoritmu řešení.
    - Příklady: Pascal Delphi Visual Basic, C++, C# Object C, Java, Visual Basic for Aplication - VBA, PHP, JavaScript, Python
- **Deklarativní (neprocedurální)**
  - Program neříká jak se problém krok po kroku řeší, ale pouze definuje požadavek na výsledek.
    - Příklady: SQL (jazyk nad databází), XML (definiční jazyk pro výměnu datových struktur)

dále na

- **Kompilované**
  - Výsledkem je zpravidla strojový kód. Vyjímku tvoří např. jazyk Java - ten se sice kompiluje, ale pouze do podoby tzv. byte-code (posloupnost instrukcí pro jiný program (interpret), nejedná se o instrukce pro procesor). Byte-code je poté dále interpretován pomocí interpretu (v případě Javy v JVM - Java Virtual Machine). Zdrojová data (tj. zdrojový text, reference na knihovny, definice objektů UI apod.) jsou nejprve převedena do strojového kódu kompilátorem, potom se až spouští přeložená aplikace, nebo knihovna.
    - Příklady: Pascal Delphi (objektový Pascal), Visual Basic, C++, C# (všechny tři jazyky patří do skupiny jazyků .NET, ale kompilátory C++ existují samozřejmě i v jiných podobách i pro jiné platformy, než je MS Windows), Object C, Java (kompiluje do ByteCode – bajtový mezikód, který je spouštěn – interpretován v prostředí virtuálního stroje Javy, v tzv. JVM)
- **Interpretované**
  - Program je překládán až v rámci vlastního spuštění, přičemž uložen je výhradně v podobě zdrojového textu. K provozu vyžaduje interpreter příslušného jazyka.
    - Příklady: Visual Basic for Aplication – VBA (makrojazyk v MS Office a dalších aplikacích), PHP (pro tvorbu webových aplikací), JavaScript (klientský skriptovací jazyk, pro tvorbu dynamických webových stránek), Python

a na konec ještě na

- **Vyšší**
  - Prakticky všechny dostupné nestrojově orientované jazyky. Programuje se na základě textových zápisů příkazů a vět.
- **Nižší**
  - Strojově orientované programy. Velmi primitivní jazyky, u nichž klíčová slova přímo zastupují instrukce procesoru.
    - Příklad: různé typy assembler

Java platforma

- Platforma Java je počítačová platforma (pracovní prostředí) zastřešující různé varianty použití programovacího jazyka Java pro vývoj a provoz různých typů aplikací.
- **Proč JAVA?**
  - Tisíce hotových komponent a zjednodušení, které je možno použít nebo je možno se jimi inspirovat
  - Dostupnost zdarma, velká komunita vývojářů, stabilní podpora
  - Osvědčená platforma prověřená velkými firmami
  - Vysoce výkonná platforma pro realizaci a běh výkonných řešení

Nejdůležitější ale na JAVĚ je její multiplatformovost! Stejný program napsaný v Javě lze spustit na OS Windows, Linux, Symbian apod. právě díky tzv. JVM (Java Virtual Machine). Nevýhodou je ale snížená rychlost běhu programu oproti aplikacím přímo zkompilovaným pro danou platformu. *Radek & Honza:)*

Ps... Nezaměňovat s JavaScriptem!

Událostmi řízené programování

- Takto naprogramované programy nepracují tak, že *postupně* vyžadují nějaká data a nakonec vypíšou nějaký výsledek. Po spuštění (např. textového editoru) nabídnou své funkce a *čekají na události* - na které tlačítko klikneme, kterou nabídku vybereme, jakou klávesu či klávesovou zkratku stiskneme.
- Při definování způsobu chování programované aplikace používáme tzv. událostmi řízené programování, když zobrazíme formulář, provede se událost Load formuláře, když klikneme na tlačítko, provede se událost Click tlačítka apod. definujeme tedy, co se má provést když nastane nějaká událost
  - Např.: JAVA, Visual studio

Sémantika, syntaxe

sémantika jazyka = význam jednotlivých symbolů

- Z nepochopení významu příkazů jazyka vznikají chyby sémantické (logické), které se často projeví až při vlastním běhu programu. Zdrojový text v mnoha případech jde přeložit (zkompilovat), ale výsledný program dělá něco jiného než má. Odladění tohoto typu chyb se prování díky tzv. debuggeru (součást vývojových prostředí).

- Příklad chyb:
  - Dělení nulou
  - Nekonečný cyklus
  - Chyba formátu vstupu (program předpokládá číselný vstup a uživatel zadá nečíselný formát)

syntaxe jazyka= pravidla jejich spojování

- Při zapsání nepovoleného spojení příkazů, případně při uvedení neznámého či nepovoleného symbolu, vzniká tzv. syntaktické chyba. Tato chyba je detekována při vlastní kompilaci zdrojového textu. Zdrojový text obsahující syntaktické chyby nejde přeložit do spustitelného kódu.

Etapy programátorské práce

- **Nápad** , nadšení, velké plány, představení problému
- **Analýza problému** - se ukazuje jako složitější, vystřízlivění, podrobení problému důkladné analýze, vypracování základního algoritmu řešení, vybrání programovacího jazyka
- **Programování** - programátoři zapisují algoritmy v programovacím jazyce
- **Ladění** - nalezení a oprava chyb v programu
- **Používání** - vlastní využívání programu
- **Modifikace, aktualizace** - úprava, vylepšení a rozšíření verze programu

--Trnka.vaclav 24. 4. 2012, 21:44 (CEST)

Programové struktury

- způsob zápisu algoritmu pomocí VD (vývojových diagramů)
- událostní funkce
- obecná funkce, objektová metoda, předávání parametrů funkci, návratová hodnota
- binární větvení
- cyklus (podmíněný, s definovaným počtem opakování)
- objekt
- příklady programů ve vybraném jazyce

Strukturované programování (strukturovaný programovací jazyk)

- Označuje v informatice **programovací techniku**, kdy se implementovaný algoritmus rozděluje na dílčí úlohy (tzn. na procedury či funkce), které se spojují v jeden celek. Na strukturované programování lze nahlížet jako na imperativní programování za využití vybraných řídicích struktur. U strukturovaného programování se vyhýbáme řídicímu příkazu skoku.

- Strukturované programování definuje, že se program může skládat pouze z následujících čtyř struktur:
  - **sekvence:** provádí posloupnost příkazů jeden po druhém
  - **větvení:** jeden nebo více příkazů je vykonán v závislosti na stavu programu (obvykle vyjadřováno klíčovými slovy if-else)
  - **cyklus:** příkazy jsou prováděny do té doby, dokud program nedosáhne nějakého stavu (obvykle vyjadřováno klíčovými slovy while, for)
  - **podprogram:** příkazy jsou shromažďovány do samostatného bloku, který má své jméno a definuje vstupy a výstupy. Tento blok (funkci nebo proceduru) lze z jiné části programu volat jeho jménem (identifikátorem). Funkce s návratovou hodnotou lze zařadit do výrazu.
- Nejznámějším důsledkem těchto zásad je snaha zabránit nebo v závislosti na programovacím jazyce alespoň omezit používání příkazu skoku. Programový kód nerespektující výše uvedené zásady se často hanlivě označuje jako „špagetový kód“.

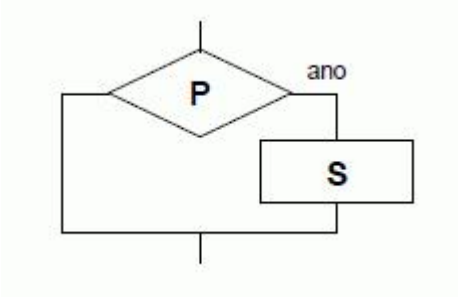
Sekvence

- Sekvence představuje posloupnost jednotlivých příkazů řazených v pořadí za sebou tak, jak mají být vykonány. Uzavřená sekvence se nazývá **složený příkaz nebo blok.**

Větvení

- Větvení umožňuje volit další postup řešení na základě splnění nebo nesplnění určité podmínky. Z obecného pohledu ale nemusí být vždy testována podmínka, neboť rozhodující pro větvení je až logická hodnota (true/false), která je výsledkem zmíněné podmínky. Větvit lze tedy i jen na základě hodnoty v logické proměnné. Větvení může být realizováno jako neúplný podmíněný příkaz, úplný podmíněný příkaz nebo jako přepínač.

1) Neúplný podmíněný příkaz



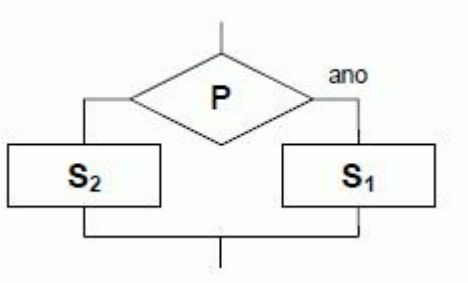
Nejprve je vyhodnocena podmínka P. Když je splněna, provede se sekvence S. Není-li podmínka splněna, nic se neprovádí a běh programu pokračuje dále.

C#	VB

```
int x;
if (x > 0) MessageBox.Show("Kladné číslo");
```

```
Dim x As Integer
If (x > 0) Then
    MsgBox "Kladné číslo"
End If
```

2) Úplný podmíněný příkaz



Nejprve je vyhodnocena podmínka P. Když je splněna, provede se sekvence S1. Není-li podmínka splněna, provede se sekvence S2.

C#	VB

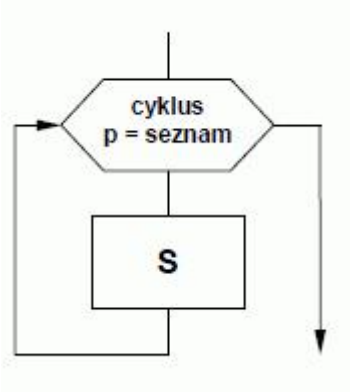
```
int x;
if (x > 0)
{
        MessageBox.Show("Je přirozené číslo");
}
else
{
        MessageBox.Show("Není přirozené číslo");
}
```

```
Dim x As Integer
If (x > 0) Then
        MsgBox "Je přirozené číslo"
Else
        MsgBox "Není přirozené číslo"
End If
```

Cyklus

- Slouží k vyjádření opakujících se operací. Počet opakování může být předem zadán, nebo je odvozen od splnění některé dané podmínky. Existují tak tři základní druhy cyklu: **cyklus s výčtem hodnot, cyklus s podmínkou na počátku a cyklus s podmínkou na konci.**

1) Cyklus s výčtem hodnot (cyklus s předepsaným počtem opakování)



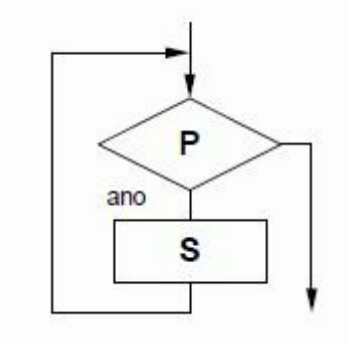
Tento druh cyklu se provádí tolikrát, kolikrát je opakování cyklu předepsáno. Předpis je dán definicí dolní hodnoty p, kroku a horní hodnoty p (případně podmínky, která musí být po celou dobu vykonávání cyklu splněna).

C#	VB

```
for (int i = 0; i < 10; i++) x[i] = nahoda.next(0,
11);
```

```
Dim i As Integer
Dim x(10) As Double
Randomize
For i = 0 To 9
        x(i) = 11 * Rnd
Next i
```

2) Cyklus s podmínkou na začátku



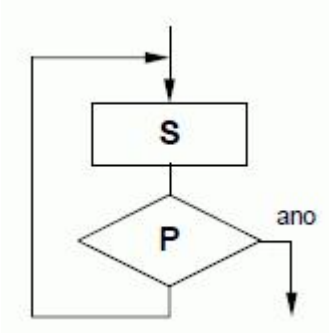
Nejprve je vyhodnocena podmínka P. Je-li podmínka splněna, provedou se příkazy sekvencí C a následuje návrat na začátek cyklu. Tato se opakuje, pokud stále platí podmínka P. Z uvedeného je zřejmé, že v sekvenci S musí být operace, která mění některou z hodnot logického výrazu podmínky P. Kdyby tomu tak nebylo, cyklus by nemohl skončit. Při dosažení neplatnosti podmínky P cyklus skončí.

Je tedy zřejmé, že cyklus s podmínkou na začátku (někdy zvaný cyklus POKUD) nemusí proběhnout vůbec (je-li podmínka P nesplněna před vstupem do cyklu), nebo jednou, či n-krát. V případě chybného naprogramování cyklu může dojít k zacyklení, což nastane, pokud podmínka P platí stále.

C#	VB

`int i=0;`
`while (i<10)`
`{`
`  x[i] = nahoda.next(0, 11);`
`  i++;`
`}`
	`Dim i As Integer`
	`Dim x(9) As Double`
	`i = 0`
	`Randomize`
	
	`Do While (i < 10)`
	`  x(i) = 11 * Rnd`
	`  i = i + 1`
	`Loop`

3) Cyklus s podmínkou na konci



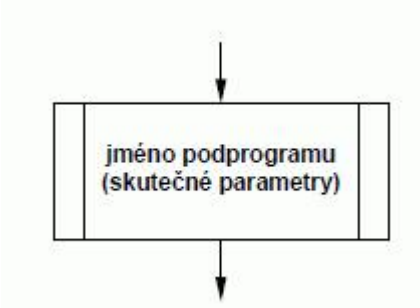
Nejprve je provedena sekvence S. Poté je vyhodnocena podmínka, pokud platí, sekvence S se znovu neprovádí a běh programu pokračuje za cyklem. Není-li podmínka splněna opakuje se sekvence S a to tak dlouho, dokud podmínka stále neplatí. Z uvedeného je zřejmé, že v sekvenci S musí být operace, která mění některou z hodnot logického výrazu podmínky P. Kdyby tomu tak nebylo, cyklus by nemohl skončit.

Tento cyklus se někdy nazývá cyklus DOKUD, čímž se myslí - dokud podmínka neplatí, opakuje se sekvence S. Tento cyklus proběhne vždy alespoň jednou. Četnost výskytu potřeby tohoto typu cyklu v programech je asi desetkrát menší než cyklu s podmínkou na začátku.

C#	VB

`int i=0;`
`do`
`{`
`  x[i] = nahoda.next(0, 11);`
`  i++;`
`} while (i<10)`
	`Dim i As Integer`
	`Dim x(9) As Double`
	`i = 0`
	`Randomize`
	
	`Do`
	`  x(i) = 11 * Rnd`
	`  i = i + 1`
	`Loop While (i < 10)`

Podprogram



- Podprogramy představují ucelené, relativně samostatné části programu, napsané v tomto programu jenom jednou, ale s možností využít je vícekrát v různých místech tohoto programu. Podprogram musí mít své jméno (identifikátor), jehož prostřednictvím se na něj v programu odvoláváme a seznam formálních parametrů, které slouží po jejich nahrazení skutečnými parametry při odvolání se na podprogram.

- Podprogram se může vyskytovat v podobě funkce bez návratové hodnoty nebo s návratovou hodnotou. Funkci s návratovou hodnotou můžeme zahrnout do výrazu.
  - Funkce, která je volána při nějaké události (kliknutí na tlačítko, stisknutí klávesy,...), se nazývá jako **událostní funkce** (také událostní procedura).

Parametry předávané do podprogramu

- Parametr je označení pro vstupní data příslušné funkce. Existují dva způsoby předávání parametrů funkci.
  - Při **předávání hodnotou** (call-by-value) se těsně před zpracováním těla funkce předávaný parametr vyčíslí a výsledek se zkopíruje do lokální proměnné uvnitř volané funkce. Jakékoli změny parametru uvnitř volané funkce nemají vliv na volající funkci, neboť se pracuje s lokální kopií hodnoty původního parametru. Předávání hodnotou tedy lze používat pouze pro vstupní parametry. Tento způsob je typický např. při vytváření aritmetických funkcí.
  - Při **předávání odkazem** (call-by-reference) se formální parametr uvnitř volané funkce bere jen jako jiné označení (alias) pro proměnnou předanou jako skutečný parametr, tzn. ve volané funkci se pracuje přímo s předávanou proměnnou, nevytváří se tedy kopie hodnoty (což zvláště u strukturovaných proměnných znamená zpravidla úsporu času i paměti). Volaná funkce změnou parametru ovlivňuje i původní volající funkci, takže předávání odkazem lze používat pro výstupní či vstupně-výstupní parametry. Nevýhodou však je, že parametrem může být jen proměnná a nikoli výsledek obecného výrazu. Předávání odkazem se obvykle implementuje pomocí ukazatele na předávanou proměnnou.

1) Předávání parametrů hodnotou

C#

```
void Zamen(int a, int b) //zameni hodnoty v
promennych
{
        int pom;
        pom = a;
        a = b;
        b = pom;
}
```

Tuto funkci budeme volat z volající funkce následně:

```
static void Main(string[] args)
{
        int promennaA = 10, promennaB = 20;
        Zamen(promennaA, promennaB);
        return 0;
}
```

VB

```
Private Sub Zamen(ByVal a As Integer, ByVal b As
Integer)
        Dim pom As Integer
        pom = a
        a = b
        b = pom
End Sub
```

Tuto proceduru (funkci) budeme volat z následující funkce následně:

```
Private Sub NejakaFunkce()
        Dim promennaA As Integer
        Dim promennaB As Integer
        promennaA = 10
        promennaB = 20
        Zamen promennaA, promennaB
End Sub
```

Při předávání parametrů funkci hodnotou zůstanou původní proměnné (`promennaA` a `promennaB`) po návratu zpět do volající funkce beze změny.

2) Předávání parametrů odkazem

C# | **VB**

```
void Zamen(ref int a, ref int b) //zameni hodnoty
v promennych
{
        int pom = a;
        a = b;
        b = pom;
}
```

Tuto funkci budeme volat z volající funkce následně:

```
static void Main(string[] args)
{
        int promennaA = 10, promennaB = 20;
        Zamen(ref promennaA, ref promennaB);
        return 0;
}
```

```
Private Sub Zamen(ByRef a As Integer, ByRef b As
Integer)
        Dim pom As Integer
        pom = a
        a = b
        b = pom
End Sub
```

Tuto proceduru (funkci) budeme volat z následující funkce následně:

```
Private Sub NejakaFunkce()
        Dim promennaA As Integer
        Dim promennaB As Integer
        promennaA = 10
        promennaB = 20
        Zamen promennaA, promennaB
End Sub
```

Po předání parametrů funkci tímto způsobem budou po návratu zpět do volající funkce v našich proměnných (`promennaA` a `promennaB`) hodnoty 20 a 10.

Objekt

*Poznámka: Pojem **objekt** souvisí s objektovým programování. Podle Procházkových slov je objektové programování vyučováno až na vysoké škole. U maturity tento pojem pravděpobně nezazní. Pokud přece jen ano, můžete říci, že jste se s objekty setkali především při programování v **C#** (ostatně i ve VB - za objekt se tam dá považovat každé tlačítko, vstupní pole, zaškrtávátko, ...). Objekt je datová struktura, která seskupuje určité **vlastnosti** (tj. proměnné objektu) a **metody** (funkce, které pracují s vlastnostmi objektu). Objektem může být cokoliv - např. člověk. Vlastnostmi (proměnnými) by poté bylo např. jméno, věk, datum narození, počet životů, barva očí, apod. Metodami (funkcemi) např. změnBarvuVlasů(), jdiSpát(), najezSe(), jdiNaMísto(xyz), pracuj(), apod.*

Proměnné a datové typy

- základní datové typy v jazyce C#
- explicitní a implicitní deklarace proměnných
- konstanta
- pole (velikost pole, rozměr pole, index, asociativní pole)
- datové typy u databází
- označování proměnných
- operace s proměnnými a sestavování výrazů
- priorita operátorů

Základní datové typy

Konkrétní hodnoty rozsahu čísel si pamatovat nemusíte, je to tu, aby jste měli představu o tom, o jakých číslech mluvíte.

VB		C#	
Celočíselné			
`Byte`	zabírá 1 byte (0–255)		
`Integer`	zabírá 2 byty, číslo se znaménkem (-32768 až 32767)	`int`	zabírá 4 byty
`Long`	zabírá 4 byty, celočíselná hodnota (-2147483648 až 2147483647)	`long`	zabírá 8 bytů
Reálné			
`Decimal`	zabírá 14 bytů, číslo se znaménkem, lze ho použít jen pomocí Variant, jehož je podtypem		
`Single`	zabírá 4 byty, číselná hodnota (-3.402823E38 až –1.401298E-45 pro záporné hodnoty, 1,401298E-45 až 3,402823E38 pro kladné hodnoty)	`float`	zabírá 4 byty
`Double`	zabírá 8 bytů, obdoba `Single` s dvoujnásobnou přesností	`double`	zabírá 8 bytů
Logické			
`Boolean`	zabírá 2 byty, uchovává hodnotu `True` (pravda), nebo `False` (lež)	`bool`	zabírá 1 byte, logická hodnota – nabývá hodnot `true`, nebo `false`
Řetězcové			
`String`	řetězec libovolných znaků kódovaných pomocí ASCII, délka prakticky neomezena (asi 2 miliardy znaků); dělí se na řetězec pevné a dynamické délky	`string`	posloupnost znaků (řetězec), 2 byty na jeden znak
Nestandardní			

`Object`	zabírá 4 byty, obsahuje odkaz na objekt (adresu objektu v paměti)	`object`	podle přiřazené hodnoty přiřazuje i datový typ
			`object a;`
			`a = 1;`
			proměnná `a` je typu `Int32`

`Variant`	zabírá 16 bytů, obecný datový typ, který může obsahovat jeden z výše uvedených datových typů, proměnná tohoto typu může dynamicky měnit svůj datový typ		
`Date`	zabírá 8 bytů, obsahuje hodnotu odpovídající datumu a času (1.leden 100 až 31.prosinec 9999,00:00:00 až 23:59:59	`DateTime`	uchovává datum a čas

Explicitní a implicitní deklarace proměnných

Explicitní

- Předem určíme, jaké proměnné budeme používat.
- Pro proměnnou rezervujeme prostor v paměti pro uložení její hodnoty za běhu programu
- Explicitní deklarace proměnné se zapisuje před jejím vlastním použitím, zpravidla na začátku procedury události, programu, funkce.
- Ve VisualBasicu začíná deklarace příkazem `Dim` (v podstatě znamená dimenze nebo dimenzovat)
- V deklaraci můžeme sdělit jaký typ dat budeme do proměnné ukládat (`Integer`, `Double`, `String`,...)
- Neuvedeme-li ve Visual Basicu v deklaraci typ proměnné (typ dat) je automaticky použit typ `Variant` - ten může obsahovat data libovolného typu a velikosti
- Ve VisualBasicu můžeme vynutit použití explicitní deklarace pomocí příkazu **`Option Explicit`**

Příklad explicitní deklarace

Visual Basic	C#

`Dim Prijmeni As String`
`Prijmeni = "Karel IV"`
Explicitní deklarace proměnné `Prijmeni`, která je datového typu `String`, ve VisualBasicu.
	`string Prijmeni = "Karel IV";`
	Explicitní deklarace proměnné `Prijmeni`, která je datového typu `string`, v C#.

S takto deklarovanými proměnnými můžeme následně dále pracovat, např. přiřadit novou hodnotu:
`Prijmeni = "Novotný"`

Implicitní

- V případě, že přímo neuvedeme deklaraci proměnné, ale rovnou proměnnou použijeme (např. v přiřazení hodnoty, ve výrazu apod.), dojde k tzv. **implicitní deklaraci**
- V případě VisualBasicu se jedná o deklaraci proměnné bez příkazu `Dim`
- Implicitní deklarace znamená, že proměnnou přímo použijeme v programu (viz příklad)
- Implicitní deklarace má výhodu **rychlejšího zápisu** programového kódu, ale nese s sebou **určitá rizika**
  - hlavním rizikem je to, že při použití implicitních deklarací nepovažuje kompilátor jazyka případný překlep ve jméně za chybu, ale považuje překlep za deklaraci nové proměnné

- Příklad implicitní deklarace proměnných `Jmeno` a `Prijmeni` - tyto proměnné budou mít datový typ `Variant`:

```
' Implicitni deklarace
Jmeno = "Pepa"
Prijmeni = "Novak"
```

- V případě použití explicitní deklarace by náš příklad vypadal následovně:

```
' Explicitni deklarace
Dim Jmeno As String
Dim Prijmeni As String
Jmeno = "Pepa"
Prijmeni = "Novak"
```

- Implicitní deklarace zavádí do kódu nebezpečí toho, že nedojde k rozpoznání překlepu, např. mějme proměnnou `Jmeno`, ve které je uložena hodnota `"Pepa"`. Budeme chtít tuto hodnotu změnit (na hodnotu `"Karel"`), ale nevšimneme si toho, že jsme ve jméně proměnné udělali překlep:

```
' Nastavime promennou Jmeno na danou hodnotu
Jmeno = "Pepa"

' Chceme hodnotu zmenit, ale omylem udelame preklep a nevsimneme si toho
Jemno = "Karel"

' Misto toho, abychom zmenili hodnotu promenne Jmeno, vytvorili jsme dalsi promennou Jemno s
hodnotou "Karel". Promenna Jmeno zustala nezmenena!
```

- *C# tuto deklaraci proměnné **nepodporuje**. V C# lze použít **pouze explicitní** deklaraci proměnné!*

Konstanta

- Konstanty jsou symboly reprezentující v programu neměnnou hodnotu. Konstantě může být přiřazen řetězec, nebo číslo.
- Příklad konstant v jazyce C#:

```
const int months = 12, weeks = 52, days = 365;
```

- Uživatel si může nadefinovat svou konstantu příkazem `const`. Definice vlastních konstant má tuto syntaxi (ve VisualBasicu):

```
[Public|Private] Const název konstanty [As typ] = výraz

' např.:
Const Months As Integer = 12
Const Weeks As Integer = 52
Public Const Days As Integer = 365

Private Const MyTitle As String = "HELP"
```

Pole

- kolekce hodnot (čísel, textových řetězců, ...) daného datového typu
- **Velikost pole** - odpovídá počtu jeho prvků. U jednorozměrného pole je tedy dána maximální hodnotou indexu. U pole dvourozměrného odpovídá součinu maximálních hodnot obou indexů.

```
' Jednorozměrné pole - 3 indexy (0-2) => velikost pole = 3; jedna se v podstate o 'seznam hodnot'
pole1(0) = 18
pole1(1) = 45
pole1(2) = 10

' Dvourozměrné pole - indexy 2x3 (0-1, 0-2) => velikost pole = 2*3 = 6; jedna se v podstate o
'tabulku hodnot' s dvema radky a tremi sloupci
pole2(0,0) = 18
pole2(0,1) = 45
pole2(0,2) = 12

pole2(1,0) = 16
pole2(1,1) = pole(0,2)   ' tj. nastavi prvku s indexem (1,1) stejnou hodnotu jakou obsahuje prvek s
indexem (0,2) - tj. 12
pole2(1,2) = 60
```

- **Index** (také nazýván jako *klíč*) - jednoznačně identifikuje prvek v poli
  - většinou je indexem celé číslo, indexy většinou začínají číslem 0
  - pomocí indexu lze přistupovat ke konkrétnímu prvku v poli a dále s tímto prvkem pracovat (*"vrať mi **3**. prvek pole"*, *"nastav **60**. prvku v poli hodnotu XYZ"*)
  - jak se s indexy pracuje (ve VisualBasicu) ukazuje příklad u předchozího bodu (*Velikost pole*)
- **Deklarace pole**

VisualBasic	C#
`Dim MePole(10) As Integer` • Deklaruje pole s názvem `MePole` o velikosti 10 prvků, každý jeden prvek pole bude datového typu `Integer` • Pole může být indexováno od 0 nebo od 1. Způsob indexování závisí na nastavení příkazu `Option Base`. Není-li uveden příkaz `Option Base = 1`, začínají se všechna pole indexovat od nuly	`int[] mePole = new int[10];` • Deklaruje pole s názvem `mePole` o velikosti 10 prvků, indexem budou čísla od `0` do `9`, každý jeden prvek bude datového typu `Int32` (celé číslo o velikosti 4 byty)

- **Asociativní pole**
  - je pole jehož prvky nejsou indexovány pomocí posloupnosti celých čísel, ale pomocí *klíčů*. Klíčem může být číslo (v nesekvenční posloupnosti), textový řetězec a jiné.

Příklad asociativního pole v jazyce PHP:

```
$foo = array(
    "a" => "1",
    "b" => "10",
    "c" => "100",
    1254 => "apple item",
```

```
    "name" => "John Doe"
);
echo $foo["c"] . $foo["b"] . $foo["a"] . $foo[1254] . $foo["name"]; // vypise text 100101apple
itemJohn Doe
```

Datové typy u databází

- **MySQL datové typy**

Číselné typy	
Název	**Rozsah čísel**
TINYINT	−128 až 127 nebo 0 až 255
SMALLINT	−32768 až 32767 nebo 0 až 65535
MEDIUMINT	−8388608 až 8388607 nebo 0 až 16777215
INT	−2147483648 až 2147483647 nebo 0 až 4294967295
DOUBLE	nejmenší nenulové hodnoty jsou ±2,2250738585072014E-308; největší nenulové hodnoty jsou −±1,7976931348623157E+308. Je-li sloupec UNSIGNED, jsou záporné hodnoty zakázané.
Řetězcové typy	
Název	**Velikost**
VARCHAR	řetězec s pevně danou délkou – od 0 do 255 znaků
TEXT	řetězec o velké velikosti
a spousta dalších řetězcových typů...	
Typy pro datum a čas	
DATE, TIME, DATETIME, TIMESTAMP, YEAR	typy pro reprezentaci datumu a času

Označování proměnných

- To jak proměnnou označíme (pojmenujeme) závisí na použitém programovacím jazyce. Obecně lze pojmenovat proměnnou podle následujících pravidel (tato pravidla většinou platí i pro pojmenování funkcí a procedur):
  - název proměnné (**identifikátor**) může být sestaven z libovolné kombinace písmen a číslic a případně dalších znaků (většinou se používá ještě podtržítko _)
  - název nesmí začínat číslicí
  - rozlišují se malá a velká písmena
  - jako identifikátor nesmí být použito klíčové (vyhrazené) slovo daného programovacího jazyka (`if`, `then`, `for`, `else`, ...)
  - název by neměl obsahovat háčky a čárky (může způsobovat problémy)

Operace s proměnnými a sestavování výrazů

VisualBasic	**C#**	**Popis**	**Příklad**
Aritmetické operátory			
+	+	sčítání	`1 + 2`
-	-	odčítání	`2 - 1`
/	/	dělení	`30/10`
*	*	násobení	`3 * 3`
Mod	%	modulo – zbytek po celočíselném dělení	`7 Mod 3` výsledek bude `1`
\		celočíselné dělení	`7 \ 3` výsledek bude `2`
Relační (porovnávací) operátory			
Relační operátory vrací hodnotu TRUE , pokud vztah platí, jinak vrací FALSE .			
<	<	menší než	`1 < 2`
>	>	větší než	`2 > 1`
<=	<=	menší nebo rovno	`10 <= 10` , `20 <= 30` ,...
>=	>=	větší nebo rovno	`3 >= 3` , `10 >= 5` , ...
<>	!=	není rovno	`7 <> 3` ☐ výsledek bude `TRUE` , `5 <> 5` ☐ výsledek bude `FALSE`

`=`	`==`	je rovno	`10 = 10` ☐ výsledek bude `TRUE` , `5 = 2` ☐ výsledek bude `FALSE`
`Like`		prověří , zda se v řetězci nachází určitý vzorek	
Logické operátory			
`AND`	`&&`	logický součin	
`OR`	`||`	logický součet	

`XOR`	`^`	exkluzivní `OR` (*buď ..., anebo ...*)	`0 XOR 0` ☐ `0`
			`0 XOR 1` ☐ `1`
			`1 XOR 0` ☐ `1`
			`1 XOR 1` ☐ `0`

`NOT`	`!`	negace argumentu	`NOT TRUE` ☐ `FALSE` `NOT FALSE` ☐ `TRUE`

Priorita operátorů

Určuje pořadí vyhodnocování částí výrazu. Nejvyšší prioritu mají operátory aritmetické, pak následují relační a nejnižší prioritu u kombinovaných výrazů mají logické operátory. V rámci aritmetických operátorů mají vyšší prioritu `*` a `/` před `+` a `-`.

Číselné soustavy

- co to jsou číselné soustavy
- rozdělení číselných soustav (poziční, nepoziční)
- obecný zápis čísla v poziční číselné soustavě
- binární a hexadecimální soustava
- nibl
- převody z desítkové do libovolné číselné soustavy, princip, příklad
- převod z libovolné číselné soustavy do desítkové, princip, příklad

10. Číselné soustavy

Číselná soustava

- číselná soustava je způsob reprezentace čísel
- zápis čísla dané soustavy je posloupností symbolů, které se nazývají číslice
- podle způsobu určení hodnoty čísla z dané reprezentace rozlišujeme dva hlavní druhy číselných soustav: poziční číselné soustavy a nepoziční číselné soustavy

Poziční číselné soustavy

- poziční soustavy jsou charakterizovány tzv. základem neboli bází (anglicky *radix*, značí se *r*), což je obvykle kladné celé číslo definující maximální počet číslic, které jsou v dané soustavě k dispozici
- v běžně používaných číselných soustavách se jednotlivé číslice zapisují za sebe, nijak se neoddělují

Decimální (desítková, dekadická) soustava

- základem je číslo 10 (r = 10), toto je pravděpodobně odvozeno od počítání s deseti prsty na rukou
- pro zápis čísel používá symboly 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- je dnes nejužívanější číselná soustava jak v běžném životě, tak ve vědě a technice

1000	100	10	1
10^3	10^2	10^1	10^0
4	2	8	6

$4286_D = 4 \cdot 10^3 + 2 \cdot 10^2 + 8 \cdot 10^1 + 6 \cdot 10^0 = 4 \cdot 1000 + 2 \cdot 100 + 8 \cdot 10 + 6 = 4286_D$

Binární (dvojková) soustava

- základem je číslo 2 (r = 2)
- pro zápis čísel používá symboly 0 a 1
- používá se ve všech moderních digitálních počítačích a elektronice, neboť její dva symboly (0 a 1) odpovídají dvěma jednoduše rozdělitelným stavům elektrického obvodu (vypnuto a zapnuto), popřípadě nepravdivosti či pravdivosti výroku (false a true)

8	4	2	1
2^3	2^2	2^1	2^0
1	1	0	1

$1101_B = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13_D$

Hexadecimální (šestnáctková) soustava

- základem je číslo 16 (r = 16)
- pro zápis čísel používá symboly 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A(10), B(11), C(12), D(13), E(14), F(15)
- písmena A - F představují (reprezentují) cifry s hodnotou 10 - 15
- díky jednoduchému vzájemnému převodu mezi šestnáctkovou a dvojkovou soustavou, se hexadecimální zápis čísel často používá v oblasti informatiky, například pro adresy v operační paměti počítače, pro zápis barvy v HTML kódu nebo IPv6 adresaci
- je to v podstatě zkrácená forma zápisu dvojkové soustavy

256	16	1
16^2	16^1	16^0
4	B	E

$4BE_H = 4 \cdot 16^2 + 11 \cdot 16^1 + 14 \cdot 16^0 = 4 \cdot 256 + 11 \cdot 16 + 14 = 1214_D$

Další poziční číselné soustavy

- **Čtyřková** – polovina bajtu = 4 bity, 16 stavů (použití v osmibitových počítačích a packed BCD)
- **Sedmičková** – dny v týdnu
- **Osmičková** – (Oktalová) dříve se používala pro adresaci
- **Dvanáctková** – hodiny (dříve též jednotka tucet)
- **Šedesátková** – nejstarší číselná soustava, snadná dělitelnost čísly 2,3,4,5,6, čas (sekundy) a geometrie (dříve též jednotka kopa)

Nibl

- nibl je polovina bajtu tj. 4 bity

Příklad:

- číslo $1010\ 1101_B$ je číslo o velikosti 1B(bajtu) tj. 8b(bitů), z toho vyplývá, že je složeno ze dvou niblů
- první nibl je **1010** a druhý **1101**, nibl obsahuje 4 číslice, protože každá z číslic představuje(reprezentuje) 1 bit

Nepoziční číselné soustavy

- hodnota číslice není dána jejím umístěním v dané sekvenci číslic
- stačí sečíst hodnoty jednotlivých číslic
- výhody: jednoduché sčítání a odečítání
- nevýhody: dlouhý zápis čísel, neobsahuje symbol pro nulu a záporná čísla
- Římské číslice, Egyptské číslice, Řecké číslice, Etruské číslice

Příklad:

- Pokud: *A = 1, B = 10, C = 100, D = 1000*
- Pak: *AAB = 1 + 1 + 10 = 12; AABBBBCCCCCDDD = 3542*

Římské číslice

- způsob zápisu čísel pomocí písmen abecedy
- dnes se tento způsob zápisu čísel používá jen výjimečně
- pro snazší zapamatování se dají používat mnemotechnické pomůcky jako např. **I**van **V**edl **X**énii **L**esní **C**estou **D**o **M**ěsta (nebo **I**van, **V**ašek, **X**énie **L**ijí **C**ín **D**o **M**umie), kde první písmena určují jak jdou římské číslice po sobě

Znak	Hodnota
I	1
IV	4
V	5
IX	9
X	10
XL	40
L	50
XC	90
C	100
CD	400
D	500
CM	900
M	1000

Převody mezi soustavami

Převodní tabulka

Dec	Hex	Bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Převody z desítkové soustavy

- postupně celočíselně dělíme základem cílové soustavy, dokud nedojdeme k hodnotě nula, přičemž zbytky po dělení v opačném pořadí představují hodnoty číslic v cílové soustavě

Desítková -> Binární

- dělíme dvojkou
- provádíme celočíselné dělení => dělením lichých čísel nám vznikne zbytek

$59_D = 111011_B$

59_D	59:2 = 29	29:2 = 14	14:2 = 7	7:2 = 3	3:2 = 1	1:2 = 0
zbytek	**1**	**1**	**0**	**1**	**1**	**1**

POZOR !!!

- zbytky se zapisují zprava

Malá čísla převedeme z hlavy:

- rychlejší způsob než rozepisování, ikdyž možná trochu obtížnější na pochopení

$13_D = 1101_B$

- jelikož víme, že 13 < 16 a 16 (2^4) je číslo o velikosti 5 bitů, budeme postupovat následovně
  - 1.) *osmička se do třináctky vejde -> 13 - 8 (2^3) = **5** -> zapíšeme 1 (**1**)*
  - 2.) *čtyřka se do pětky vejde -> 5 - 4 (2^2) = **1** -> zapíšeme 1 (**11**)*
  - 3.) *dvojka se do jedničky nevejde -> 1 < 2 (2^1) -> zapíšeme 0 (**110**)*
  - 4.) *jednička se do jedničky vejde -> 1 - 1 (2^0) = **0** -> zapíšeme 1 (**1101**)*
  - 5.) *máme výsledek **1101_B***

$9_D = 1001_B$

- jelikož víme, že 9 < 16 a 16 (2^4) je číslo o velikosti 5 bitů, budeme postupovat následovně
  - 1.) *osmička se do devítky vejde -> 9 - 8 (2^3) = **1** -> zapíšeme 1 (**1**)*
  - 2.) *čtyřka se do jedničky nevejde -> 1 < 4 (2^2) -> zapíšeme 0 (**10**)*

- 3.) *dvojka se do jedničky nevejde -> 1 < 2 (2^1) -> zapíšeme 0 (**100**)*
- 4.) *jednička se do jedničky vejde -> 1 - 1 (2^0) = **0** -> zapíšeme 1 (**1001**)*
- 5.) *máme výsledek **1001_B***

Desítková -> Hexadecimální

- dělíme šestnáctkou
- provádíme celočíselné dělení

$175_D = AF_H$

175_D	175:16 = 10	10:16 = 0
zbytek	15	10
zbytek	**F**	**A**

POZOR !!!

- zbytky se zapisují zprava

Převody do desítkové soustavy

- sčítáme hodnoty součinů jednotlivých číslic se základem zdrojové soustavy umocněným na pořadové číslo pozice číslice zprava počítáno od nuly

Binární -> Desítková

- Sčítáme mocniny dvojky vynásobené hodnotami binárních číslic

$111011_B = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 32 + 16 + 8 + 4 + 2 + 1 =$ ***59_D***

Hexadecimální -> Desítková

- sčítáme mocniny šestnácky vynásobené desítkovými hodnotami šestnáctkových číslic

$ABCDEF_H = 10 \cdot 16^5 + 11 \cdot 16^4 + 12 \cdot 16^3 + 13 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0 =$ ***11259375_D***

$6C_H = 6 \cdot 16^1 + 12 \cdot 16^0 = 96 + 12 =$ ***108_D***

Převody mezi dvojkovou a šestnáctkovou soustavou

Binární -> Hexadecimální

- převádíme z hlavy převodem po čtveřicích binárních číslic – tzv. NIBL (čtveřici sestavujeme zprava)

$1011100001010011101_B = 2E14D_H$

Bin	10	1110	0001	0100	1101
Dec	2	14	1	4	13
Hex	2	E	1	4	D

Hexadecimální -> Binární

- převádíme z hlavy po jednotlivých číslicích a vzniknou nibly. Ty řazeny za sebou dávají výsledné binární číslo

$AB37_H = 1010101100110111_B$

Hex	A	B	3	7
Dec	10	11	3	7
Bin	1010	1011	0011	0111

Operační systémy

Operační systém je základní softwarové vybavení počítače, které je zavedeno do paměti počítače při jeho startu a zůstává v činnosti až do jeho vypnutí. Skládá se z jádra (kernel) a pomocných systémových nástrojů. Hlavním úkolem OS je zajistit uživateli možnost ovládat počítač, vytvářet pro procesy stabilní rozhraní a spravovat systémové prostředky. Nejčastěji používané OS: OS Microsoft Windows, Linux a MAC OS.

Historicky první počítač

- Historicky první (elektronkový) počítač byl ENIAC (*Electronic Numerical Integrator And Computer*). Do provozu byl uveden v roce 1943 a to pro armádní účely. Vstup i výstup obstarávaly děrné štítky. Zabíral plochu o rozloze 63 metrů čtverečních (jedna velká hala) a vážil 27 tun. Nepoužíval dvojkovou (jako dnešní počítače), ale dekadickou (desítkovou) soustavu.

Základní programové vybavení

- = Software/Operační systém
- je programové vybavení počítače, které umožňuje spuštění nebo zpracovávání aplikačního software. Je to rozhraní mezi hardwarem a aplikačním softwarem.

Aplikační programové vybavení

- = aplikační software
- je v informatice veškeré programové vybavení počítače, které je určeno pro přímou práci s uživatelem. Účelem je zpracování a řešení konkrétního problému uživatele. Má grafické nebo textové rozhraní.
  - textové editory (např. MS Word, OO Writer...)
  - databázové systémy (např. MS Acces, OO Base...)
  - antiviry (např. Avast, Eset...)
  - webové prohlížeče (např. IE, Google Chrome, Firefox...)
  - a další...

Základní funkce operačního systému

- 3 základní funkce
  - **Ovládání počítače** - umožňuje uživateli spouštět programy, předávat jim vstupy a získávat jejich výstupy s výsledky. Součástí bývá i grafické rozhraní, kde je pro uživatele ovládání počítače jednoduché a srozumitelné.
  - **Abstrakce hardware** (poznání, používání) - Operační systém skrývá detaily ovládání jednotlivých zařízení v počítači a definuje standardní rozhraní pro volání systémových služeb tak, že vytváří abstraktní vrstvu s jednoduchými funkcemi. Zjednodušeně řečeno vytváří rozhraní pro programy, kterým umožňuje ovládat hardware a jiné možnosti do snadno použitelných funkcí. (**API** = rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí či tříd nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat.)
  - **Správa prostředků** - přiděluje a odebírá procesům systémové prostředky počítače (řídí chod procesů, přiděluje procesům prostor v paměti...) V případě potřeby může operační systém procesům přidělené prostředky i násilně odebrat (preempce). Zároveň si i samozřejmě nechává místo pro chod sebe samotného.

Multitasking = schopnost OS provádět v reálném čase několik procesů současně (střídá je). (Linux, Windows ale NIKOLI DOS)

Historie OS

- UNIX - 1969, multitaskingový a víceuživatelský OS, první verze psány v assembleru, poté v jazyce C
- MS DOS - 1981, OS komercializovaný Microsoftem, pracuje se v něm pomocí příkazového řádku
- MAC OS - 1984, první kompletně grafický OS vyvinut firmou Apple
- první Windows pro DOS - 1990, Windows pouze nadstavba DOSu, spouštěl se jako samostatný program a přinášel grafické rozhraní
- Linux - 1991, postaven na UNIXU
- První Windows NT - 1993, samostatný operační systém Windows

Rozdělení OS

Podle distribuce

- **Linux** (různé distribuce, odlišují se podle použití konkrétním uživatelem a jsou vytvářeny proto, aby uživatel nemusel skládat jádro a SW do funkčního celku. Volně dostupný na Internetu. Ubuntu, Debian, Red Hat, Mandriva...)
- **Windows** Pochází od firmy Microsoft, mají grafické uživatelské rozhraní a podporují multitasking. Nejrozšířenější OS. (Windows 7, Windows Vista, Windows XP, Milenium...)
- **Mac OS** je označení původního operačního systému pro počítače Macintosh firmy Apple. Tento systém se používal na počítačích Macintosh od roku 1984 do začátku 21. století. Během posledních let svého používání přestal systém vyhovovat rostoucím nárokům a

přešlo se na Mac OS X.(Mac OS DP1-4, Mac OS X 10.0 Cheetah, Mac OS X 10.1 Puma, Mac OS X 10.2 Jaguar, Mac OS X 10.7 Lion…)
- **BSD** – je odvozenina Unixu distribuovaná Kalifornskou univerzitou už v 70. letech, (Solaris (2.5 – 10))

Podle účelu

- **Desktopové** - Je systém určený k kancelářské práci nebo pro domácnost. Pro uživatele jednoduché prostředí, kde mohou instalovat své aplikace a využívat je například ke kancelářským účelům. Mnoho OS však mohou sloužit jako desktopový systém tak i jako server . (Windows 7, Windows Vista, Windows XP… Ubuntu…)
- **Serverově orientované** - Systém který je sestaven tak, aby mohl poskytovat klientům služby. Optimalizován na výkon a schopnost poskytnout službu hodně uživatelům zároveň. (Windows server… Debian…)

Open source / uzavřený vývoj

- **Linux** - open source má otevřený kód = umožňuje uživateli volně měnit nastavování operačního systému nebo přidávat doplňující aplikace (a všechny údaje jsou volně k dispozici). Smí se volně šířit, je zdarma.
- **Windows** - Licencovaný, uživatel si nemůže volně nastavovat operační systém, měnit jeho kód.

operační systémy pro mobilní zařízení

-umožňují instalovat další aplikace, je zde možnost synchronizace s PC

- **PalmOS** - systém určený pro PDA, výhodou jsou nízké nároky na výkon, paměť a rychlost, jelikož v jednom okamžiku může běžet pouze jedna aplikace, Při přepínaní mezi aplikacemi si aplikace uloží poslední stav, takže uživatel nepozná, že se aplikace spouští znovu. (Novější verze umí spustit i dvě aplikace zároveň kvůli přehrávání hudby na pozadí.)
- **Windows Mobile** - je operační systém firmy Microsoft. Používá vzhled odvozený od klasických Microsoft Windows a malou podmnožinu jejich komponent. Má jiné jádro než klasické windows. Novější verze: Windows phone 7
- **Symbian** - operační systém navržen pro plné využití mobilních zařízení (tzv. „chytrých telefonech“ – smartphone). Umožňuje instalovat spousty aplikací, které však závisí na verzi systému. Převážně je dnes používán v mobilních telefonech značky Nokia. Od roku 2011 však Nokia definitivně přešla k používání platformy Windows Phone 7.
- **iOS** - původně iPhone OS je operační systém vytvořený společností Apple pro mobilní telefony iPhone. Dnes se používá i na zařízeních jako jsou iPod Touch nebo iPad. iOS je odlehčenou verzí operačního systému Mac OS X, používaného v počítačích společnosti Apple.
- **Android** - je rozsáhlá open source platforma, která vznikla zejména pro mobilní zařízení. Operační systém je založený na jádru Linuxu. Při vývoji systému byla brána v úvahu omezení, kterými disponují klasické mobilní zařízení jako výdrž baterie, menší výkonnost a málo dostupné paměti. Zároveň bylo jádro Androidu navrženo pro běh na různém hardwaru. Systém tak může být použit bez ohledu na použitý chipset, velikost či rozlišení obrazovky. Vyvíjí společnost Google.

Uživatelské rozhraní

- **GUI** (= graphical user interface) – ovládání PC pomocí ikonek, tlačítek, oken a jiných grafických prvků. Začal s ním Xerox, následně Windows)
- **Příkazový interpreter** – ovládání PC příkazovým řádkem OS

Souborový systém

- způsob organizace dat ve formě souborů (a většinou i adresářů) tak, aby k nim bylo možné snadno přistupovat.
- **Fat 32** (fat původně vznikl pro dos, fat 32 v dnešní době není ideální jelikož neumožňuje ukládat velké soubory jako třeba obrazy disků nebo velká videa)
- **NTFS** (nový souborový systém od Microsoftu, přizpůsoben novým náročným požadavkům (např. ukládání velkých souborů, nebo třeba kompresi už na úrovni souborového systému)
- **Ext4** - žurnálovací souborový systém vytvořen pro jádro Linuxu. Systém souborů ext4 může podporovat svazky až o velikosti 1 EiB a soubory s maximální velikostí 16 TiB.

BIOS

- základní vstupně–výstupní systém
- Programový kód BIOSu je uložen na základní desce v nevolatilní (stálé) paměti typu ROM, EEPROM nebo modernější flash paměti s možností jednoduché aktualizace
- používá se hlavně při startu počítače pro inicializaci a konfiguraci připojených hardwarových zařízení a následnému spuštění operačního systému, kterému je pak předáno další řízení počítače

Virtualizace

Jsou to postupy a techniky, které umožňují k dostupným zdrojům přistupovat jiným způsobem, než jakým jsou fyzicky propojeny. Lze

virtualizovat celý stroj (virtuální PC) nebo jeho HW komponenty (virtuální paměť , procesory,...). Aplikace: VMWare, MS Virtual PC...

- Emulace = virtualizace hardwarových komponent za účelem simulace jiné hardwarové platformy. Hostované operační systémy a aplikace není nutné modifikovat
- Paravirtualizace = virtuální stroj nesimuluje hardware, ale místo toho nabízí speciální aplikační rozhraní (API), které vyžaduje určité modifikace hostovaného operačního systému, aby mohl být tento OS nad virtuálním strojem spouštěn
- Nativní virtualizace = virtuální stroj simuluje dostatek hardwaru, aby bylo umožněno nemodifikovanému operačnímu systému běžet izolovaně od hostitelského operačního systému

Administrace OS, příklady Windows a Linux

- **Windows**
  - Windows umožnuje dva způsoby administrace. Pomocí přikazového řádku (CMD) a pomocí grafického rozhraní.
    - Grafické rozhraní
      - Umožnuje nastavit v podstatě veškeré možné nastavení ve systému Windows, všechno je to popsané, proto není problém pro běžné uživatele nastavení měnit.
    - Příkazový řádek
      - U Windows se moc nepoužívá. Avšak se pomocí něho dá nastavit úplně všechno.
      - příklad: netsh interface ipv4 set address name="Wifina" dhcp -> nastaví na připojení s jménem Wifina získávání ip adresy z dhcp serveru
      - Lze použít více příkazů najednou pomocí dávkového souboru .bat
- **Linux**
  - V operačních systémech na jádře Linux se převážně používá příkazový řádek. V grafickém prostředí (Ubuntu například) se dá použít i aplikace grafické.
    - Příkazový řádek
      - Je interpretován skriptovacím jazykem Bash
      - Pomocí příkazového řádku lze nastavit úplně vše, v linuxu se dá měnit i nastavení jádra.
      - příklad
        - cp - kopíruje soubory
        - rm - ruší soubory
        - mkdir - vytváří adresáře
        - rmdir - ruší prázdné adresáře
        - passwd - mění heslo u daného uživatele
    - Grafické prostředí
      - například aplikace StartUpManager pro nastavení grub.cfg (Soubor, který se spouští hned po spuštění Biosu, pomocí grubu lze například nastavit DualBoot (Více operačních systémů na jednom disku))

Databáze

Databáze je propracovaný systém pro *ukládání dat* a jejich *následné zpracování*. Obsahuje data uložená na *paměťovém médiu*. Tato data mezi sebou mají určité *vztahy* a jsou určitým způsobem *členěna*. V širším pojetí spadají do pojmu databáze i *nástroje*, které s daty pracují (ukládají je, mění je a mažou je).

Na internetu se databáze používají při *programování webových aplikací*. Například tento slovníček je tvořen databází jednotlivých pojmů a skriptem, který s nimi pracuje (vypisuje, řadí a propojuje). Tato data by samozřejmě bylo možné uložit i do souboru, databáze však obvykle fungují mnohem rychleji, bývají optimalizovány pro přístup více uživatelů (návštěvníků stránek) a obsahují mechanismy, které práci s daty usnadňují. Opravdu složité aplikace, jako např. *redakční systém* či *internetový obchod* se bez využívání databáze neobejdou.

- Typy databáze:
  - Relační (níže)
  - Objektové - umožňují skladování dat s libovolnou strukturou
  - Speciální - (např Hierarchická - data uspořádána ve stromové struktuře, Síťová)

Relační databáze

Relační databáze = *kolekce dat*, která slouží pro popis reálného světa (např. evidence školní knihovny, sklad chemikálií, evidence studentů). Nechová jen jako běžné úložiště dat, ale díky *aplikační logice systému*, který databázi řídí, je schopen svá data rychle *třídit, vyhledávat* v nich a *řadit* je podle různých kritérií.

Relace = samostatná tabulka, která se skládá ze sloupců a řádků. Každý sloupec (atribut) má předem definovaný datový typ (např. celé číslo, řetězec, datum, logická hodnota ...). Složitějším příkladem relace pak je propojení více takovýchto tabulek.

Index = slouží ke zrychlení vyhledávacích a dotazovacích procesů v databázi, definování unikátní hodnoty sloupce tabulky.

Primární klíč = pole nebo kombinace polí, jednoznačně identifikující každý záznam v databázové tabulce. Žádné pole, které je součástí primárního klíče, nesmí obsahovat hodnotu NULL. Každá tabulka má mít definovaný právě jeden primární klíč.

Cizí klíč = (v prostředí relačních databází) u tabulky vytvoří spojení jednoho nebo více jejích sloupců se sloupcem nebo sloupci jiné („cizí“) tabulky

Entita = prvky reálného světa (např. člověk, stroj, vyučovaný předmět, město), který je popsán svými charakteristikami (vlastnostmi). Ty se většinou označují jako *atributy* (např. jméno, příjmení, stav, plat, hmotnost).

- Vztahy mezi entitami:
  - **1:1** - např. jeden člověk má uloženy stejné osobní údaje na městském úřadě stejně jako na správě sociálního zabezpečení
  - **1:N** - např. jeden člověk vlastní *n* platebních karet (jedna karta může mít pouze jednoho vlastníka)
  - **M:N** - bez omezení, např. jedna kniha má více autorů, zároveň ale autoři napsali více knih než tu jednu

Referenční integrita

Referenční integrita je nástroj databázového stroje, který pomáhá udržovat vztahy v relačně propojených databázových tabulkách.

Definuje se jako vztah mezi dvojici tabulek. Tabulka, v níž je pravidlo referenční integrity uvedeno, se nazývá *podřízená tabulka*. Tabulka, jejíž jméno je v integritním omezení uvedeno, je *nadřízená tabulka*. V nadřízené i podřízené tabulce je určen sloupec nebo skupina sloupců tvořící *klíč*. Pravidlo referenční integrity vyžaduje, aby pro každý záznam v podřízené tabulce, jehož klíč nemá hodnotu NULL, existoval v nadřízené tabulce záznam se stejným klíčem. Klíč v nadřízené tabulce musí být unikátním indexem, klíč v podřízené tabulce musí být nějakým indexem.

Při manipulaci s daty se referenční integrita projevuje takto:

- Přidáváte-li záznam do podřízené tabulky (resp. přepisujete-li hodnotu sloupce z klíče), kontroluje se existence stejného klíče v nadřízené tabulce.
- Při smazání hodnoty sloupce z klíče (zrušením záznamu nebo přepsáním hodnoty) v nadřízené tabulce se kontroluje, zda v podřízené tabulce není záznam se stejnou hodnotou klíče.

Systém řízení báze dat (SŘBD)

Databázový systém, softwarové vybavení, které zajišťuje práci s databází, tzn. tvoří rozhraní mezi aplikačními programy a uloženými daty. Musí být jednak schopen efektivně pracovat s velkým množstvím dat, ale také musí být schopný řídit a definovat strukturu těchto

perzistentních dat.

Relational DataBase Management System (RDMS)

Síťové služby (servery), zpracovávající současně více požadavků na data od mnoha uživatelů do více databází.

- Typy:
  - *freeware, shareware* - MySQL, PostgresSQL, Firebird
  - *licencované* - Oracle, MS SQL server, Sybase

Structured Query Language (SQL)

Strukturovaný dotazovací jazyk sloužící pro tvorbu univerzálních dotazů v databázích, zakládání tabulek, ošetření přístupu k datům, sdílení dat nebo třeba pro zabezpečení databází. SQL se většinou používá v kombinaci s některým výkonějším programovacím jazykem.

SQL management software - univerzální nástroje pro správu dat SQL databází - *PHPMyAdmin, Adminer*

MS Access = nástroj pro správu relačních databází, který umí pracovat s daty MS-Jet, MS SQL server, Oracle a je součástí kancelářského software MS Office, jednoduché uživatelské prostředí, bez znalostí SQL jazyka

Základy jazyka vznikly v 70. létech 20. století při výzkumu relační databáze firmou IBM - jazyk *SEQUEL*. Dalšími vývoji (např. spolupráce s firmou Oracle) vznikl jazyk dnešní podoby.

- Typy dotazů:

  - Dotazy manipulující s daty:
    - SELECT - příkaz vrací množinu záznamů z jedné a nebo více tabulek
    - INSERT - přidá do tabulky relační databáze nový záznam
    - UPDATE - příkaz upravuje data (záznamy) v relační databázi. Může
    - DELETE - příkaz, sloužící k odstranění záznamů z tabulky relační databáze
  - Dotazy pro definici dat:
    - CREATE - je příkaz , který slouží k vytváření databázových objektů.
      - CREATE TABLE - nejběžnější, vytvoření tabulky
    - ALTER - je příkaz , který slouží ke změně databázových objektů.
    - DROP - je příkaz , který slouží k odstranění databázových objektů
  - Dotazy pro pro definici práv k datům:
    - GRAND - příkaz, kterým lze v relačních databázích nastavit přístupová práva k jednotlivým tabulkám
    - REVOKE - příkaz, kterým lze odebrat přístupová práva k jednotlivým tabulkám

Aplikační model databázové aplikace (obrázek propojení tabulek) - klikni zde

Příklady SQL

Vytvoření tabulky

```
CREATE TABLE Adresar (Jmeno varchar(50), Mail varchar(50), Adresa varchar(50),);
```

Přidání hodnoty

```
INSERT INTO Adresar (jmeno, email, adresa) VALUES ('Karel', 'karel@praha', 'Praha 5');
```

Výpis tabulky

```
SELECT * FROM Adresar;
```

Podmíněný výběr

```
SELECT * FROM Adresar WHERE Jmeno LIKE 'Karel';
```

Příklady MySQL

```
// navazani vlakna k MySQL
$spojeni=@mysql_connect("host","user","password");
```

```
// výběr uživatelské databáze
@mysql_select_db("userdb",$spojeni);
$result = mysql_query("SELECT * FROM zbozi", $spojeni);
```

Externí odkazy

- Wikipedia - Databáze
- Wikipedia - Relační databáze

Tvorba webových stránek

- co to je webová stránka (statická, dynamicky generovaná)
- serverový a klientský software pro provoz webového obsahu
- protokoly webu
- role W3C
- popis HTML (tagy párové a nepárové, příklady
- struktura HTML dokumentu
- CSS
- formuláře na webu
- klientské skriptování a DOM
- ActiveX, JavaApplet

Co to je webová stránka (statická, dynamicky generovaná)

Webová stránka:

- má svůj obsah a nese určitou informaci
- skládá se z textu, multimediálních dat (obrázky, videa, zvuky, ...) a odkazů, které umožňují přechod na další webové stránky
- může být uložena v podobě souborů na pevném disku nebo je poskytuta webovým serverem prostřednictvím počítačové sítě (LAN, Internet, ...)
- po síti je přenášena pomocí protokolu HTTP
- je psána pomocí HTML nebo XHTML (případně může být použito i XML, ale není to příliš pohodlné)
- zobrazuje se uživateli skrz webový prohlížeč

Webová stránka poskytuje své informace prostřednictvím WWW (World Wide Web). WWW je služba internetu – informační systém založený na hypertextovém modelu. Jedná se v podstatě o nejrozšířenější internetovou službu. Veškeré webové stránky jsou součástí WWW.

Webové stránky se dělí na:

- **statické** - obsahují stále stejný obsah, jsou uloženy v souborech
- **dynamické** - mění svůj obsah v čase, vytváří je program na straně webového serveru

Stránka se může měnit i přímo v prohlížeči použitím **skriptovacích jazyků**, **Javy**, **ActiveX** a dalších technologií.

Serverový a klientský software pro provoz webového obsahu

- **serverový software**
  - **webový server** - zpřístupňuje uživateli požadované webové stránky na základě jeho požadavku
    - generují dynamické webové stránky (používají další technologie - PHP, CGI, Perl, ...)
    - např. Apache, nginx, ISS (od Microsoftu vyžaduje Windows)
  - **software pro dynamické generovaní webových stránek** - jedná se především o webové skriptovací jazyky - PHP, Perl, ..., ale i o externí programy, které vygenerují stránku na základě požadavku serveru
- **klientský software**
  - **webový prohlížeč** - slouží k zobrazení webových stránek na koncovém zařízení - nejčastěji monitoru počítače
    - např. Opera, Firefox, Chrome, Internet Explorer, Safari, Links (pracuje v textovém režimu), apod.

Protokoly webu

HTTP (port TCP/80)

Internetový protokol pro výměnu hypertextových dokumentů ve formátu HTML. Aktuální je verze 1.1 (definována v RFC 2616).

V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e-mail), používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.

Používá (podobně jako některé jiné programy a technologie) tzv. jednotný lokátor prostředků - URL (Uniform Resource Locator). URL specifikuje jednoznačné umístění nějakého zdroje v Internetu.

Vlastnosti

- funguje způsobem **dotaz - odpověď** (uživatel pošle serveru dotaz a server mu zašle odpověď)
- **dotaz** - jedná se o čistý text
  - obsahuje označení požadovaného dokumentu, informace o schopnostech prohlížeče apod.
- **odpověď** - začíná několika řádky textu
  - tyto úvodní řádky popisují výsledek dotazu - zda se podařilo dokument najít, jakého je dokument typu, atd.
- za těmito úvodními řádky následují vlastní data dokumentu - text, HTML kód, MP3 soubor, obrázek, video-klip, apod.
- jednotlivé dotazy jsou na sobě nezávislé - svým způsobem nevýhoda - viz další bod
- jedná se o **bezstavový protokol** - neumí uchovávat stav komunikace, jednotlivé dotazy spolu nemají žádnou souvislost
  - to je nepříjemné při realizaci složitějších procesů přes HTTP - např. e-shop potřebuje uchovávat informace o totožnosti (identitě) zákazníka, o obsahu košíku, apod.
  - proto byl protokol rozšířen o tzv. HTTP Cookies - umožňují uložit serveru na počítači uživatele libovolné informace - ty lze využít pro uchování informací o stavu komunikace

První verze HTTP (verze 0.9) pochází z roku 1991. Obsahovala pouze metodu GET s jediným parametrem - názvem požadovaného dokumentu. Server jako odpověď poslal přímo požadovaný dokument bez hlavičky (`HTTP/... 200 OK...`). Případná chybová hlášení vracel server ve formě HTML dokumentu.

Verze 1.0 pochází z roku 1996. Zavádí **HTTP hlavičky**, metody `POST` a `HEAD` a pro identifikaci typu dokumentu používá MIME.

Verze 1.1 byla poprvé popsána v roce 1997 (RFC 2068), ale tento popis byl nahrazen v červnu 1999 (v RFC 2616). Zavádí možnost udržet TCP spojení (tzv. keep-alive connection) - to umožňuje zpracovat více dotazů po sobě v rámci jednoho TCP spojení. Dále přidává metody `OPTIONS`, `PUT`, `DELETE`, `TRACE` a `CONNECT`.

Příklad komunikace

Klient (např. webový prohlížeč) se připojí na server cs.wikipedia.org a zašle následující dotaz:

```
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
User-Agent: Opera/9.80 (Windows NT 5.1; U; cs) Presto/2.5.29 Version/10.60
Accept-Charset: UTF-8,*
```

Tím žádá server o zaslání dokumentu /wiki/Wikipedie na server cs.wikipedia.org, sděluje svou totožnost (Opera verze 10.60) a oznamuje, že podporuje kódování UTF-8. *(ve skutečnosti obsahuje dotaz takových informací mnohem více)*

Server odpoví:

```
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8
```

za touto hlavičkou následuje jeden prázdný řádek (označení konce hlavičky) a následují vlastní data - v našem případě se bude jednat o HTML dokument. Hlavička obsahuje informaci o tom, že dotaz se podařil (první řádek: „200 OK“), datum a čas vyřízení dotazu, popis serveru, který odpovídá, informace o typu vráceného dokumentu (MIME typ text/html v kódování UTF-8) a další informace.

Dotazovací metody

HTTP definuje metody (příkazy), které se mají provést nad uvedeným objektem (dokumentem). Každý HTTP doraz začíná právě specifikací metody:

```
<metoda> <objekt> HTTP/<verze-http>
```

- **GET** - příkaz "vrať mi daný objekt"; nejpoužívanější metoda
  - požadavek na uvedený objekt - požadavek (dotaz) může obsahovat další případná data (proměnné prohlížeče, Cookies, Session ID, ...)
- **HEAD** - příkaz "vrať mi informace o daném objektu"
  - to samé jako GET, ale v odpovědi nejsou žádná data - pouze hlavičky obsahující informace o požadovaném objektu - velikost, typ, datum poslední změny, ...
- **POST** - odesílá uživatelské data na server
  - daty může být odeslaný vyplněný formulář, soubor, atd.
  - s objektem se pak zachází podobně jako při metodě GET
  - data může odeslat i metoda GET, ale POST je určen pro příliš velká data (víc než 512 bajtů - to je velikost požadavku GET), nebo pro situace, kdy nechceme, aby data nebyla součástí URL (adresy) - při POSTu jsou data obsažena v HTTP požadavku a ne v adrese.
- **PUT** - příkaz "nahraj data na server"
  - objekt je jméno vytvářeného souboru
  - používá se zřídka - pro nahrání dat na server se používá hlavně FTP nebo SCP/SSH
  - jsou pro to potřeba jistá oprávnění
- **DELETE** - příkaz "smaž uvedený objekt ze serveru"
  - jsou na to potřeba jistá oprávnění stejně jako u metody PUT
- **TRACE** - odešle kopii obdrženého požadavku zpět odesílateli
  - klient tak může zjistit, co na požadavku mění nebo přídávají servery, kterými požadavek prochází
- **OPTIONS** - dotaz na server, jaké podporuje metody
- **CONNECT** - spojí se s uvedeným objektem přes uvedený port
  - používá se při průchodu skrze proxy pro ustanovení kanálu SSL

HTTPS (port TCP/443)

Pro zabezpečenou komunikaci přes protokol HTTP se používá jeho zabezpečená varianta známá jako **HTTPS**. HTTPS je syntakticky identické jako HTTP, pouze přidává signalizaci prohlížeči, aby použil šifrovací metodu **SSL/TLS** k přenosu dat. SSL je pro HTTP vhodné, protože dokáže poskytnout ochranu přenosu, i když je ověřená pouze jedna strana komunikace (typicky pouze server a uživatel např. jenom potvrdí certifikát).

Druhou možností zabezpečení přenosu přes protokol HTTP je nadstavba protokolu HTTP ve verzi 1.1, která byla představena v RFC 2817. Tato metoda ale není příliš podporována prohlížeči, takže se pro zabezpečenou komunikaci častěji využívá HTTPS.

Role W3C (World Wide Web Consortium)

Mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Bylo založeno v roce 1994. Cílem konsorcia je *„Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic, které zajistí dlouhodobý růst Webu“*. Zabývá se také vzděláním a přístupností, vyvíjí software a nabízí otevřenou diskuzi o Webu prostřednictvím fóra. Úkolem konsorcia je vydávat specifikace (doporučení), kterými by se poté měli řídit tvůrci webových prohlížečů a ostatních programů, které pracují s webem. Před založením konsorcia nabízely různé firmy různé upravené verze jazyka HTML které byly nekompatibilní s verzemi od ostatních výrobců. Konsorcium sjednotilo verze od různých výrobců a dohodlo se s nimi na základních principech a komponentách nových standardů.

Na půdě konsorcia tak vznikla řada technologií - např. kaskádové styly (CSS), XML, formát pro publikaci vektorových obrázků na webu SVG a mnoho dalších.

Vývoj specifikací W3C trvá velice dlouho, protože každý standard (doporučení) prochází několika fázemi (od pracovního návrhu (Working Draft), LastCallWorkingDraft, Candidate Recommendation, Proposed Recommendation až po W3C Recommendation). Doporučení je pak postupně revidováno prostřednictvím samostatně vydávaných errat (*Errata*). Když se nashromáždí větší množství těchto úprav, je vydána nová edice doporučení. Než doporučení projde do závěrečné fáze, může to trvat i několik let. Díky tomu je konsorciu často vyčítána malá pružnost a neschopnost reagovat na aktuální situaci v rychlém vývoji moderního webu.

Popis HTML (tagy párové a nepárové, příklady)

Zdrojový kód HTML je složen z textu a **značek** (tagů). Všechny značky jsou uzavřeny ve špičatých závorkách < a >. Tagy mohou obsahovat parametry - ty nazýváme **atributy** a nachází se mezi špičatými závorkami hned za jménem značky.

```
<jmenoZnacky atribut1="hodnota atributu 1" atribut2="hodnota atributu 2">
```

Atributy upravují a nastavují výsledné zobrazení/chování značky. Např. chceme vložit obrázek, proto použijeme značku `<img>`. Když značku zapíšeme do dokumentu v této podobě, nic se víceméně nestane - prohlížeč totiž neví, jaký obrázek má na místě značky zobrazit (zobrazí bílé ohraničené místo, nebo také nic). Proto do značky doplníme atribut `src`: `<img src="obrazek.jpg">`. Jestliže takový obrázek existuje prohlížeč ho zobrazí v místě, kde se ve zdrojovém kódu nachází značka `<img>`. Pokud ale takový obrázek existovat nebude, opět se nám nic nezobrazí. Proto můžeme tag rozšířit o další atribut `alt`: `<img src="neexistujici-obrazek.jpg" alt="Obrázek stromu">`. Atribut `alt` obsahuje tzv. **alt**ernativní text - tzn. text, který se zobrazí místo obrázku, pokud daný obrázek nebude existovat. Nyní se nám místo bílého místa (v případě neexistující obrázku) zobrazí text "Obrázek stromu". Pomocí atributů lze upravit chování řady značek v HTML kódu.

Tagy se můžou vyskytovat v párech `<něco>` a `</něco>`. Takové tagy nazýváme jako **párové**. Ten první tag něco začíná a druhý něco končí. Lomítko znamená, že jde o **uzavírací tag** (ten bez lomítka je **otevírací**). Příklady párových tagů:

- `<head>` je začátek hlavičky a `</head>` je její konec
- `<body>` je začátek těla stránky a `</body>` je konec těla stránky
- `<b>` je začátek tučného textu a `</b>` je konec tučného textu
- atd.

Tagy, které nemají uzavírací tag, nazýváme jako **nepárové**. Párové tagy něco ohraničují/vyznačují (např. tučný text) - **nepárové** zastupují ve stránce nějaký prvek - např. obrázek, horizontální čáru, apod., nebo mají vztah k celému dokumentu (připojují styly CSS, sdělují doplňující informace o stránce - autor, popis, znaková sada, apod.). Příklady nepárových tagů:

- `<img src="obrazek.jpg">` na dané místo ve stránce vloží obrázek "obrazek.jpg"
- `<hr>` vloží na dané místo horizontální čáru
- `<meta name="description" content="Popis mé skvělé webové stránky">` - obsahuje popis stránky
- `<meta name="author" content="John Doe">` - sděluje informaci o tom, kdo je autorem stránky
- `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">` - sděluje prohlížeči jakou stránka využívá znakovou sadu (kódování)

Struktura HTML dokumentu

Na začátku každého HTML dokumentu je tzv. **DOCTYPE**. Ten oznamuje prohlížeči, kterou konkrétní verzi HTML používáme. Za ním následuje element `html` a v něm vnořené elementy `head` a `body`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <title>Titulek stránky</title>
    </head>
    <body>
        Vlastní obsah stránky.
    </body>
</html>
```

DOCTYPE obsahuje nejrůznější informace, např. podle jakého DTD (viz XML) je HTML stránka napsána. V případě HTML na DOCTYPE, ale tolik nezáleží, ve většině případů ho prohlížeče ignorují.

> *... Jediné, na čem záleží, je to, aby se stránka vykreslila v IE stejně jako v jiných prohlížečích. Toho se dosáhne tím, že se do stránky přidá libovolná doctype deklarace, která Explorer přepne do standardního módu.*
> Zdroj: http://www.jakpsatweb.cz/doctype.html

Nejnovější HTML5 používá jako DOCTYPE prostý zápis `<!DOCTYPE html>`.

Hlavička stránky (element `head`) obsahuje informace o připojených stylech (CSS), skriptech (JavaScript), informace o autorovi, klíčová slova, stručný popis stránky, titulek stránky, použité kódování (UTF-8, windows-1250, ...), apod. Vlastní obsah stránky, který vidí návštěvník webu je umístěn v elementu `body`.

CSS

Kaskádové styly (v anglickém originále Cascading Style Sheets se zkratkou CSS) je jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

Jazyk byl navržen standardizační organizací W3C, autorem prvotního návrhu byl Håkon Wium Lie. Byly vydány zatím dvě úrovně specifikace CSS1 a CSS2, dne 7. června 2011 byla dokončena revize CSS 2.1 a pracuje se na verzi CSS3. Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Výhody

- rozsáhlejší možnosti formátování oproti vizuálním elementům v HTML
- jednodušší údržba webové prezentace. Pokud chceme změnit nějaký detail, jako třeba barvu nadpisu, nemusíme složitě procházet HTML kód nebo různé HTML šablony, ale můžeme změnit pouze jednu vlastnost v CSS souboru, který je přilinkován ke všem stránkám.
- **oddělení struktury a stylu** - v jednom (HTML) dokumentu budeme mít pouze sémanticky označen obsah a v druhém (CSS) dokumentu máme definice vzhledu stránek. Tím dosáhneme snadnějšího strojového zpracování samotného obsahu stránek, do kterého se nám nepletou prvky definující vzhled.
- cachování stylů – webový prohlížeč si může soubor se styly uložit do cache paměti, čímž může být dosaženo zrychlení načtení stránky. Na druhou stranu při použití externího CSS souboru dochází k dalšímu HTTP požadavku navíc oproti tomu, když bychom použili buď online CSS nebo přímo formátování HTML
- CSS vlastnosti jednotlivých elementů můžeme dynamicky měnit pomocí Javascriptu
- můžeme určit styl pro různá **výstupní zařízení** - máme tak možnost nadefinovat jiný styl pro tisk, projekt, mobil, PDA či webový prohlížeč. Specifikace CSS nezapomínají ani na zrakově postižené – je možno napsat styly pro hlasový syntetizátor nebo hmatovou čtečku Braillova písma
- koncový uživatel si může napsat svůj vlastní CSS styl pro libovolnou stránku. Většina prohlížečů nějakým způsobem podporuje uživatelské styly, takže uživatel si může například nastavit, aby měl všechny odkazy na všech webech vždy podtržené nebo aby na tomto konkrétním webu mělo písmo vždy černou barvu

Formuláře na webu

Formuláře jsou nezbytnou součástí www stránek. Používají se například pro získání názoru čtenáře stránky, slouží k odesílání vzkazů z www stránek apod. Formulář je obsažen v HTML elementu `<form>` a obsahuje různá vstupní pole (textová políčka, rozbalovací seznamy, zaškrtávátka, přepínače, textová pole, pole pro hesla, pro výběr souboru, apod.) a tlačítka. Vstupní pole slouží k vyplnění a zadání daných údajů do formuláře, tlačítka slouží např. pro odeslání formuláře na server.

Formuláře neodvratně patří k modernímu webu a hlavně k webových aplikacím. Můžou mít různou podobu a účel - např. registrační formulář, formulář pro přihlášení k e-mailu, anketní formulář, formulář pro napsání diskusního příspěvku, apod.

Data z formuláře by měla být vždy nějak zpracována - jinak přítomnost formuláře postrádá smysl. Data z formuláře bývají často odeslána na server, kde jsou zpracována nějakým programem nebo skriptem (např. registračním PHP skriptem). Pro odeslání formulářových dat na server se používají 2 metody - **get** a **post**.

- **metoda GET** - odešle data na server jako součást adresy HTTP požadavku. Tato data jsou v adrese umístěna za otazníkem. Protože je délka adresy omezena, nehodí se tato metoda pro odesílání většího množství dat. Jednotlivé údaje z formuláře jsou odděleny znakem &. Příklad adresy s odeslanými daty - `http://example.com/zbozi.php?hledej=televize&serad=sestup&maxcena=5000`. Data v adrese jsou na straně serveru zpracována nějakým programem nebo skriptem. Adresa s daty je normálně dostupná v historii prohlížeče - to může být nevýhoda, pokud chceme přenášená data skrýt.
- **metoda POST** - odesílá data na server v těle HTTP požadavku a ne v jeho adrese. To umožňuje odesílat objemnější data (např. soubory), protože nejsme omezeni maximální délkou adresy. Také není přenášená data vidět v historii prohlížeče a v logovacích souborech webového serveru, proxy serverů a dalších zařízeních "po cestě". Tuto metodu se doporučuje používat hlavně v případech, kdy odesílaná data nějakým způsobem mění stav aplikace/serveru - např. když odesíláme diskuzní příspěvek, mažeme článek z webu, přihlašujeme se k

webové aplikaci, apod. Takto se liší od metody GET, která se používá hlavně v případech, kdy chceme nějaká data získat (např. vyhledat všechny televize pod 5000 Kč v e-shopu) a nechceme data (např. v databázi) měnit.

Formuláře také mohou být zpracovány JavaScriptem na straně klienta - v takovém případě se data na server nepřenáší.

Klientské skriptování a DOM

Na straně klienta (webového prohlížeče) je možné spouštět programový kód, který je schopen manipulovat se stránkou, upravovat ji, reagovat na akce uživatele, apod. V oblasti webových stránek je nejrozšířenějším klientským skriptovacím jazykem **JavaScript** (v IE jeho obdoba JScript). Je možné použít i jiné jazyky, ty ale nemají takovou podporu v prohlížečích a díky jejich minimálnímu využívání je lze zanedbat (patří sem např. některé jazyky z dílny Microsoftu, které jsou podporovány pouze v Internet Exploreru).

Jelikož se jedná o skriptovací jazyky (a ne jazyky kompilované) je na straně klienta nutná přítomnost **interpretu**, který je schopen do stránky vložený kód načíst a vykonat dané příkazy. Takový interpret bývá zabudován do webového prohlížeče. V posledních letech vedou tvůrci webových prohlížečů zuřivý boj o to, kdo z nich vytvoří rychlejší a méně chybový interpret JavaScriptu a tím nabídnou svých uživatelům možnost používat stále náročnější JavaScriptové aplikace. Díky JavaScriptu lze v dnešních prohlížečích editovat fotografie, psát dokumenty, vytvářet tabulky jako v Excelu, editovat prezentace, malovat, hrát hry (existují např. JavaScriptové emulátory pro staré hry - např. emulátor původního DOOMa). Spousta společností a služeb využívá JavaScript ve svých aplikacích - bez JavaScriptu si neškrtnete např. na Facebooku, Twitteru a spoustě dalších webových stránkách.

Využití

- kontrola formulářů
- nejrůznější efekty na stránkách
- potřeba okamžitě reagovat na podměty uživatele
- manipulace s DOM
- atd.

DOM (Document Object Model - objektový model dokumentu)

DOM je objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je API (aplikační rozhraní) umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí. Původně měl každý webový prohlížeč své vlastní specifické (a s ostatní nekompatibilní) rozhraní pro manipulaci s HTML elementy pomocí JavaScriptu. Proto došlo v rámci W3C ke standardizaci tohoto rozhraní a vznikl tak W3C Document Object Model (zkráceně W3C DOM). Tato specifikace je platformně a jazykově nezávislá.

DOM umožňuje přístup k dokumentu jako ke **stromu**, což je zároveň datová struktura používaná ve většině XML parserů a XSL procesorů.

Specifikace W3CDOM je rozdělena úrovní (levels). Každá úroveň specifikuje určité požadavky a podmínky na programy, které chtějí prohlásit, že podporují daný DOM level. Programy pak musí implementovat všechny požadavky této úrovně a úrovní nižších. Např. pokud chce aplikace prohlásit, že podporuje DOM level 1 musí umět navigaci v DOM dokumentu a manipulaci s jeho obsahem (včetně přidávání elementů).

Pomocí DOM tak můžeme v JavaScriptu vybrat konkrétní elementy a dále s nimi manipulovat, upravovat a mazat je nebo vkládat zcela nové elementy.

ActiveX, Java Applet

Kromě JavaScriptu je možné vložit do stránky také jiné objekty, které jsou schopné se stránkou nějakým způsobem manipulovat. Na rozdíl od JavaScriptu se nejedná o skriptovací jazyky, ale o externí programy, většinou zkompilované do podoby nějakého balíčku, který je vložen do webové stránky. Mezi tyto technologie patří např. **ActiveX** od Microsoftu a **Java applety**.

ActiveX

ActiveX je technologie, kterou vyvinula společnost Microsoft pro sdílení informací mezi různými aplikacemi. ActiveX spolupracují pouze s aplikacemi, jako je Microsoft Word, Excel, Internet Explorer, PowerPoint a pracují pouze na počítači s operačním systémem Windows.

Nejznámější použití ActiveX je tzv. ActiveX Controls – jde o malé soubory kódů, které si mohou uživatelé prohlížeče Internet Explorer

stáhnout a spustit v počítači. ActiveX Controls jsou psané v běžných programovacích jazycích, jako je Visual Basic a C++. Nejedná se o samostatné aplikace a mohou být spuštěny pouze z hostitelské aplikace jako například Internet Explorer, MS Office apod. ActiveX Controls lze přirovnat k Java Appletům, na rozdíl od nich však mají plný přístup k operačnímu systému, neboť jde o objekty COM a mají tak neomezený přístup k počítači. ActiveX Controls mohou přistupovat k místnímu systému souborů a měnit nastavení registru operačního systému.

Na webových stránkách lze ActiveX komponenty používat od roku 1996, kdy byla tato představena a zároveň byla vydána nová verze Internet Exploreru - verze 3.0. ActiveX komponenty jsou objekty postavené na technologii COM. Prakticky jakákoliv aplikace může být vytvořena v podobě ActiveX componenty. Lze implementovat jednoduché objekty zobrazující prostý text i složité objekty typu tabulek programu Excel apod.

Na stránku můžeme ActiveX objekt umístit pomocí tagu `<object>`, jemuž nastavíme parametr `classid`. Ten obsahuje hodnotu GUID (Global Unique IDentifier) – jednoznačné hexadecimální číslo identifikující COM objekt. Seznam objektů se na klientovi uchovává v systémovém registru Windows. Pokud na cílovém počítači ještě není přítomen objekt, na který se stránka odkazuje, je prohlížečem stáhnut z adresy dané parametrem CodeBase tagu `<object>` a nainstalován, tzn. jsou přidány záznamy do registru systému. Pak již lze COM objekt spustit a zobrazovat jeho grafický výstup v prohlížeči.

ActiveX jsou často zneužívány k virovým útokům. ActiveX se po stažení stávají součástí operačního systému se schopností manipulace s hardware i software počítače. Microsoft po kritice namísto omezení funkcí ActiveX zvolil v následujících verzích IE jinou cestu zabezpečení. Microsoft vyvinul systém registrací a certifikátů, díky nimž může prohlížeč prvky ActiveX ověřovat ještě před samotným zavedením. Je ponecháno na rozhodnutí uživatele, zda je ActiveX legitimní, nebo se jedná např. o trojského koně.

ActiveX je dnes již překonaná technologie, která navíc dokáže být nebezpečná a její používání se nedoporučuje.

Podobné funkcionality bývá v ostatních prohlížečích a na ostatních platformách dosaženo pomocí tzv. plug-inů. Je to knihovna, kterou prohlížeč načte při svém spuštění a která modifikuje vnitřní chování prohlížeče. Tímto způsobem přidává prohlížeči novou funkcionalitu a to různého typu. Některé plug-iny lze asociovat s MIME typem. Je-li pak pomocí tagu <embed> vložen do HTML stránky soubor určitého MIME typu, je spuštěn asociovaný plug-in (např. pro přehrávání videí a spuštění flashe (např. pro flash hry) bývá používán plug-in Flash Player, pro zobrazení PDF souborů plug-in Adobe Readeru, apod.).

Java Applet

V programovacím jazyku Java je zdrojový kód nejprve napsán jako jednoduchý text a uložen do souborů s koncovkou `.java`. Tyto zdrojové soubory se potom kompilujjí do souborů s koncovkou `.class` pomocí Java kompilátoru. Soubor s koncovkou `.class` však neobsahuje kód, který dokáže přímo zpracovávat procesor; místo toho obsahuje tzv. bajtkód (bytecode) - instrukce pro virtuální počítač Java Virtual Machine (Java VM, JVM). Při spuštění Java programu, tak dojde ke spuštění JVM - pomocí něho je pak program spouštěn (interpretován). Java tak kombinuje vlastnosti kompilovaných jazyků (kompilace - program je rychlejší, než kdyby byl interpretován) a jazyků skriptovacích (interpretovaných) - snadná přenositelnost a nezávislost na kontrétní platformě či operačním systému.

Applety jsou malé programy, jejichž soubor `.class` s byte kódem je uložen na serveru - vždy je ale vykonáván na straně klienta v rámci jiného programu (nejčastěji webového prohlížeče). Aby bylo možno používat applety, musí být na klientském počítači instalován Java Virtual Machine a prohlížeč je musí podporovat.

Abychom v prohlížeči mohli spustit applet, vložíme do stránky HTML tag `<applet>` s atributem, který specifikuje umístění appletu na serveru. Jakmile prohlížeč načte tento tag, spustí tzv. Class Loader (součást JVM), jenž ze serveru stáhne kód appletu a kód všech potřebných tříd, na něž se applet odkazuje a které ještě nemá klient v počítači. Jsou-li všechny požadované soubory dostupné, je applet spuštěn.

Aby byla zajištěna nezbytná bezpečnost při běhu appletu, omezuje dostupnou funkcionalitu tzv. Security Manager, který umožňuje uživateli zvolit, jaké skupiny funkcí mají být povoleny či zakázány. Obvykle nemají applety přístup do souborového systému a nemohou navazovat spojení s jinými servery, než odkud byly stáhnuty.

Applety se používají především k implementaci složitějšího grafického prostředí na straně klienta, než je schopna poskytnout technologie dynamického HTML. Za tím účelem je appletu na stránce HTML vymezen obdélníkový prostor, jehož rozměry jsou dány dalšími atributy tagu `<applet>`. V této oblasti je veškeré vykreslování ponecháno na appletu a prohlížeč ho neovlivňuje.

Webová aplikace

- princip webové aplikace, důvody zavedení této platformy
- struktura webové aplikace, vrstvy
- Aplication Service Provider
- webový server a jeho funkce ve webové aplikaci
- metody přenosu dat od klienta na server
- uživatelské rozhraní webové aplikace
- technologie tvorby aplikační loginy webové aplikace
- datová vrstva

Princip webové aplikace, důvody zavedení této platformy

Webová aplikace je v softwarovém inženýrství aplikace poskytovaná uživatelům **z webového serveru** přes **počítačovou síť** Internet, nebo její vnitropodnikovou obdobu (intranet). Webové aplikace jsou populární především pro všudypřítomnost webového prohlížeče jako klienta. Prohlížeč se pak nazývá tzv. **tenkým** klientem**, neboť sám o sobě nezná logiku aplikace.**

Schopnost aktualizovat a spravovat webové aplikace bez nutnosti šířit a instalovat software na potenciálně tisíce uživatelských počítačů je hlavním důvodem jejich oblíbenosti. Webové aplikace jsou používány pro implementaci mnoha podnikových i jiných informačních systémů, ale i freemailů, internetových obchodů, online aukcí, diskusních fór, weblogů.

Struktura webové aplikace, vrstvy

Ačkoliv je mnoho možností, webové aplikace jsou obvykle strukturovány jako **třívrstvé**. V té nejběžnější formě je:

- **prezentační vrstva** (první vrstva) - webový prohlížeč
- **vrstva aplikační logiky** (střední vrstva) - nástroje pro dynamické generování stránek (např. CGI, PHP, javové servlety nebo ASP)
- **datová vrstva** (třetí vrstva) - např. databáze

Webový prohlížeč posílá požadavky střední vrstvě, která je obsluhuje prostřednictvím dotazů do databáze a generováním uživatelského rozhraní.

Aplication Service Provider (ASP)

Jednou ze strategií pro softwarové firmy je poskytnout přístup přes web k aplikacím, které byly dříve nabízeny a šířeny jako aplikace lokální. V závislosti na typu aplikace může takový přechod vyžadovat vývoj zcela odlišného uživatelského rozhraní určeného webovému prohlížeči nebo jen přizpůsobit stávající aplikaci pro použití jiné prezentační technologie. Tyto programy umožňují uživatelům platit měsíční či roční poplatek za používání aplikace, aniž by si jej museli nainstalovat na svůj pevný disk. Firmy podnikající s touto strategií jsou známé jako **poskytovatelé aplikačních služeb** (Application Service Provider - ASP).

Webový server a jeho funkce ve webové aplikaci

Samotná aplikace je umístěna na webovém serveru (řekněme centrálně), kam se k ní můžou připojit ostatní uživatelé. Webový server zpřístupňuje aplikaci uživatelům (klientům - skrz webový prohlížeč) a je na něm spouštěna samotná aplikace, která reaguje na uživatelovi podněty - vypiš XYZ, vlož ABC, smaž DGX, apod.

Nejpoužívanějším webovým serverem je **Apache**. Jedná se open-source projekt, který tak lze zdarma šířit, upravovat a měnit. Apache je velice komplexní a robustní server. Z toho důvodu je mu někdy vyčítána komplikovanost (např. komplikovaná změna nastavení - existují desítky různých konfiguračních voleb - direktiv).

Často nasazovaným je také webserver **nginx** (čti engine-x). Ten je zaměřen především na vysoký výkon. Základním cílem je rychlá distribuce statického obsahu a možnost rozložení zátěže na další servery dle nastavené priority. Často bývá nasazován tam, kde je nutné poskytnout uživateli přístup k velkému množství statických souborů (např. obrázků, statických HTML souborů). V těchto případech bývá rychlejší než např. Apache. Bývá používán i jako proxy server. Často ho nasazují velké firmy (např. Seznam.cz, Nokia, GitHub (služba pro sdílení zdrojového kódu), WordPress, Dropbox, apod.).

Na platformě Windows (především v rámci **Windows Serveru**) se používá webový server od Microsoftu - IIS. Jedná se o softwarový

webový server s kolekcí rozšiřujících modulů, vytvořený společností Microsoft pro operační systém Windows. IIS 7.5 podporuje protokoly HTTP, HTTPS, FTP, FTPS, SMTP a NNTP. Je součástí produktové řady Windows Server a také některých vydání Windows XP, Windows Vista a Windows 7. IIS není ve výchozím nastavení Windows zapnut.

Pro generování dynamických webových stránek se používá řada technologií:

- **SSI (Server Side Includes)**
  - do HTML kódu se zapisují jednoduché instrukce, které zpracovává přímo webový server
  - to, že se v souboru mají hledat SSI, se pozná podle přípony souboru (obvykle .shtml)
- **CGI**
  - aplikace používající Common Gateway Interface - protokol, který definuje způsob komunikace webového serveru s aplikací
  - CGI aplikace (nejčastěji skripty) lze psát v téměř libovolném jazyce, stačí dodržet konvence rozhraní Perl, C/C++, Pascal, Python, ...
  - rozhraní CGI bylo v prostředí internetu přítomno již od počátku 90. let a ve své době představovalo jediný způsob dynamického zpracování obsahu
  - podpora CGI nebývá implicitní, musí se ve web-serveru zapnout (bezpečnost)
  - hlavní nevýhoda: pro obsluhu každého požadavku je spouštěn nový proces
- **FastCGI**
  - vylepšená varianta CGI, které významně snižuje zátěž serveru
  - každý skript se do paměti načítá jen jednou, pak postupně obsluhuje další požadavky
  - web-server s aplikací komunikuje pomocí TCP/IP
  - web-server a aplikaci je možné rozdělit na samostatné počítače
  - po určitém počtu obsloužených požadavků skript je možné kdykoliv podle potřeby ukončit, webový server si ho při dalším požadavku sám znovu spustí
- **SAPI (moduly webserverů)**
  - v průběhu času začala většina serverů nabízet kromě CGI rozhraní i speciálně přizpůsobené rozhran, které umožní, že aplikace je integrována přímo do webového serveru (nejčastěji jako DLL knihovna)
  - dnes nejpoužívanější je ISAPI – podporují ho servery Microsoftu a mnohé další
  - do paměti se podobně jako FastCGI skripty načtou při prvním požadavku a pak v ní již zůstanou
  - SAPI moduly jsou binární nativní kód – pro tvorbu si musíme sehnat vhodný kompilátor
- **ASP (Active Server Pages)**
  - přímo do HTML kódu se zapisují jednoduché příkazy
  - ASP je tzv. framework
  - standardně je používán programovací jazyk JScript nebo VBScript
  - ve jazycích jsou dostupné základní objekty s důležitými informacemi (data z formulářů apod.)
  - standardní součást webových serverů MS
  - podpora jiných serverů a platforem je velice slabá
- **ASP.NET**
  - ASP.NET je součást .NET Frameworku pro tvorbu webových aplikací a služeb
  - ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku
  - programátoři tak mohou realizovat své projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic.NET, JScript.NET, C#, Managed C++, ale i mutace Perlu, Pythonu a další
  - aplikace založené na ASP.NET jsou rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu interpretovány
- **PHP (PHP Hypertext Preprocessor)**
  - přímo do HTML kódu se zapisují jednoduché příkazy
  - jednoduchá syntaxe založená na C, Perlu a Javě
  - speciálně navržený jazyk pro tvorbu webových aplikací
  - velmi rozsáhlá knihovna funkcí
  - nezávislost na platformě – může spolupracovat s v podstatě libovolným serverem na libovolné platformě
  - dostupný zdarma včetně zdrojových kódů
- **Java servlet**
  - servlet je speciální třída zapsaná v jazyce Java
  - podobně jako u ISAPI a FastCGI zůstává servlet po prvním načtení v paměti a obsluhuje další požadavky
  - vyžaduje webserverovou podporu Javy - např. Apache Tomcat pro server Apache nebo Adobe JRun pro server IIS
- **JSP (Java Server Pages)**
  - do HTML kódu se zapisují příkazy Javy
  - k dispozici jsou (podobně jako v ASP) speciální objekty pro čtení dat předávaných pomocí HTTP

    - zdrojový kód stránky JSP je vyparsovaný a následně kompilovaný na servlet (bytecode), který generuje HTML kód
    - vyžaduje webserverovou podporu Javy - např. Apache Tomcat pro server Apache nebo Adobe JRun pro server IIS
- a další programovací jazyky nebo frameforky, jako jsou např. Ruby on Rails, Django - Python, ...

Metody přenosu dat od klienta na server

Pro přenos dat od uživatele na server (např. data z formuláře) se používají dvě metody - metoda **POST** a metoda **GET**.

Data odeslaná metodou **GET** se stanou **součástí URL**, doplní se za otazník a jsou tedy vidět v adresním řádku prohlížeče. Naopak metoda **POST** data odesílá **odděleně od URL**.

Pokud se předaná data dají chápat jako parametry stránky (tedy např. předání ID článku nebo vyhledávaného řetězce), je vhodné je poslat metodou GET, v ostatních případech je lepší použít metodu POST (obzvláště pokud je dat hodně – např. Internet Explorer dokáže zpracovat URL jen do délky 2083 znaků).

Jakékoliv stažení objektu ze serveru je provedením příkazu GET. Vyvolá jej každé kliknutí na odkaz `<a href=...>`, transparentně jej provede např. stažení obrázku (`<img src=...>`).

POST je ryze formulářová záležitost. Data se odesílají mimo URL – to se ovšem týká jen položek formuláře. Zároveň je totiž možné odesílat ještě další data v URL, které je uvedeno v action formuláře. Zjednodušeně řečeno, POST umí totéž co GET a ještě o dost víc.

Použití metody POST by také mělo být samozřejmé v případě, kdy předaná data obsahují citlivé údaje (jako např. heslo), protože jinak se tato data dají získat řadou způsobů (např. z historie prohlížeče, na proxy serveru, z logů webového serveru nebo z HTTP hlavičky Referer).

Metoda POST se používá hlavně v případě, když:

- odeslání formuláře způsobí vnitřní změnu serveru (zápis do databáze, odeslání e-mailu, ...)
- odesílaná data nejsou zcela veřejná
- velikost dat může překročit 1000 bajtů (např. odeslání souboru na server - v tom případě je potřeba ve formuláři uvést atribut `enctype="multipart/form-data"`)

Uživatelské rozhraní webové aplikace

Uživatelské rozhraní webové aplikace je podobně jako obyčejná webová stránky (ona to je totiž ve skutečnosti webová stránka) napsána pomocí HTML. Vizuální podoba rozhraní a rozložení jednotlivých prvků na stránce je v režii kaskádových stylů (CSS). Pro získání většího dojmu interaktivity se používá klientské skriptování - pomocí skriptovacího jazyka JavaScript.

JavaScript manipuluje s prvky na stránce pomocí objektového rozhraní DOM (Document Object Model - objektový model dokumentu). Např. můžeme prvky na stránce dynamicky měnit na základě činnosti uživatele. S JavaScriptem a moderními webovými aplikacemi souvisí i technologie AJAX.

AJAX se používá pro přenos dat ze serveru ke klientovi bez nutnosti načítat znovu celou stránku. Pouze se stáhnou potřebná data nebo úryvek HTML kódu. Tento kód nebo data následně nahradí nebo doplní určitou oblast stránky (stáhnou se např. nejnovější příspěvky v chatu, stáhne a zobrazí se náhled výsledné podoby blogového příspěvku, apod.).

Součástí webového uživatelského rozhraní jsou **odkazy**, **tabulky**, **tlačítka**, **formuláře** a případně další prvky odpovídající konkrétnímu účelu aplikace.

Technologie tvorby aplikační loginy webové aplikace

Ačkoli je mnoho webových aplikací psáno přímo v čistém programovacím jazyce jako je PHP či Perl, existuje pro jejich tvorbu řada systémů, tzv. **frameworků**, které díky automatizaci tohoto procesu nabízejí programátorům možnost popsat program na vyšších úrovních. Užití takových systémů může často snížit počet chyb v aplikaci, především díky větší jednoduchosti a přehlednosti kódu a také možností koncentrovat se na důležitější části kódu.

Datová vrstva

- zajišťuje práci s vlastními daty aplikace

- uchovává tato data a skrz dané rozhraní je zpřístupňuje vlastní aplikační logice (samotné aplikaci)
- často se jedná o databázi, kterou pohání některý z databázových serverů (MySQL, MSSQL, Oracle, ...)
- v databázi jsou data uchovávána v databázových **tabulkách**
- nic však nebrání tomu napsat si datovou vrstvu založenou na ukládání a čtení dat do/ze souborů

Použití databáze má tu výhodu, že poskytuje dostatečnou abstrakci pro operace s daty. Tyto operace bývají často dobře navržené a optimalizované pro maximální možný výkon při práci s daty. Umožňují tak velice rychle vybírat data z tabulek, provádět s daty výpočty, spojovat data z více tabulek do jedné datové sady, rychle nalezená data řadit (vzestupně/sestupně) apod. V případě databází se pro manipulaci s daty nejčastěji používá jazyk **SQL**.

Počítačová grafika

Počítačová grafika

Z technického hlediska se jedná o obor informatiky, který používá počítač k tvorbě umělých grafických objektů a k úpravě nasnímaných obrazů z reálného světa.

Historie

- r. 1960: vznik pojmu počítačová grafika
- konec 80. let: 3D počítačová grafika

Rozdělení 2D počítačové grafiky

Vektorová grafika

Je to jeden ze dvou základních způsobů reprezentace obrazových informací v počítačové grafice. Zatímco v rastrové grafice je celý obrázek popsán pomocí hodnot jednotlivých barevných bodů (pixelů) uspořádaných do pravoúhlé mřížky, vektorový obrázek je složen ze základních geometrických útvarů, jako jsou body, přímky, křivky a mnohoúhelníky.

Výhody

Vektorová grafika má proti rastrové grafice některé výhody:

- Je možné libovolné zmenšování nebo zvětšování obrázku bez ztráty kvality (viz ukázka v úvodu článku).
- Je možné pracovat s každým objektem v obrázku odděleně.
- Výsledná velikost v bajtech, je u vektorového obrázku obvykle mnohem menší než u rastrové grafiky.

Nevýhody

- Oproti rastrové grafice zpravidla složitější pořízení obrázku. V rastrové grafice lze obrázek snadno pořídit pomocí fotoaparátu nebo skeneru.
- Překročí-li složitost grafického objektu určitou mez, začne být vektorová grafika náročnější na operační paměť a procesor než grafika bitmapová.

Použití

- Vektorová grafika se používá zejména pro počítačovou sazbu, tvorbu ilustrací, diagramů a počítačových animací.
- Pro práci s vektorovou grafikou se používají vektorové editory (např. Adobe Illustrator, CorelDraw, Inkscape, Sodipodi, Zoner Callisto).

Bézierova křivka

Teoretickým základem vektorové grafiky je analytická geometrie. Obrázek není složen z jednotlivých bodů, ale z křivek – vektorů. Křivky spojují jednotlivé kotevní body a mohou mít definovanou výplň (barevná plocha nebo barevný přechod). Tyto čáry se nazývají Bézierovy křivky.

Francouzský matematik Pierre Bézier vyvinul metodu, díky které je schopen popsat pomocí čtyř bodů libovolný úsek křivky. Křivka je popsána pomocí dvou krajních bodů (tzv. kotevní body) a dvou bodů, které určují tvar křivky (tzv. kontrolní body). Spojnice mezi kontrolním bodem a kotevním bodem je tečnou k výsledné křivce.

Formáty vektorové grafiky

- PostScript - .eps, .ps
- Portable Document Format - .pdf
- Adobe Illustrator Artwork - .ai
- Corel Draw - .cdr
- Scalable Vector Graphics - .svg
- Zoner Callisto - .zmf

Rastrová grafika

Celý obrázek je popsán pomocí jednotlivých barevných bodů (pixelů). Body jsou uspořádány do mřížky. Každý bod má určen svou přesnou polohu a barvu v nějakém barevném modelu (např. RGB). Tento způsob popisu obrázků používá např. televize nebo digitální fotoaparát.

Kvalitu záznamu obrázku ovlivňuje především rozlišení a barevná hloubka.

Výhody

- jednoduché zobrazení a programová podpora.
- pořízení obrázku je velmi snadné například pomocí fotografie nebo pomocí skeneru.

Nevýhody

- velké nároky na zdroje (při vysokém rozlišení a barevné hloubce velikost obrázku dosahuje i jednotek megabytů, v profesionální grafice se běžně operuje i s podklady o desítkách megabytů)
- změna velikosti (zvětšování nebo zmenšování) vede ke zhoršení obrazové kvality obrázku
- zvětšování obrázku je možné jen v omezené míře, neboť při větším zvětšení je na výsledném obrázku patrný **rastr**

Použití

- digitální fotografie, snímky videa, textury
- používá se častěji než vektor

Barevná hloubka

- 1bitová barva (2^1 = 2 barvy) také označováno jako Mono Color (nejpoužívanější je, že bit 0 = černá = a bit 1 = bílá)
- 4bitová barva (2^4 = 16 barev)
- 8bitová barva (2^8 = 256 barev)
- 15bitová barva (2^{15} = 32 768 barev) také označováno jako Low Color
- 16bitová barva (2^{16} = 65 536 barev) také označováno jako High Color
- 24bitová barva (2^{24} = 16 777 216 barev) také označováno jako True Color
- 32bitová barva (2^{32} = 4 294 967 296 barev) také označováno jako Super True Color (někdy také jako True Color)
- 48bitová barva (2^{48} = 281 474 976 710 656 = 281,5 biliónů barev) také označováno jako Deep Color

Formáty rastrové grafiky

- BMP
  - pro true color zobrazení (24 bit) se nepoužívá komprese, u obrázků s menší barevnou hloubkou může být použita RLE komprese
  - obrázky mají velkou datovou velikost (šířka x výška x barevná hloubka)
  - barvy jsou reprezenovány ve formátu nezávislém na zobrazovacím zařízení, převod zajišťuje ovladač výstupního zařízení
  - univerzální, dobře dokumentovaný formát bez licenčních omezení
- GIF
  - bezztrátová komprese, LZW
  - pro obrázky s malým počtem barev (maximálně 256), s jednobarevnými plochami
  - ukládá barevnou paletu
  - obrázek se ukládá jako množina pravoúhlých obrazců se společným pozadím; části odpovídající pozadí tak nenesou žádnou informaci, nejsou kódované
  - možnost průhledného pozadí
  - možnost uložení více obrázků v jednom souboru
  - možnost animace
  - možnost prokládaného ukládání obrázku (rychlý náhled)
- PNG
  - zaměřen pro přenos obrázků po síti
  - slučuje výhody GIFu a JPGu
  - dovoluje uložit až 48 bitovou barevnou hloubku
  - má samostatný alfa kanál (8 nebo 16 bit), který uzmožňuje uložit průhlednost a průsvitnost
  - používá bezztrátovou kompresi i pro true color obrázky, LZ77 a Huffmanovo kódování
  - dvourozměrné prokládání
- TIFF

  - velice univerzální formát
  - umožňuje uložit obrázek ve vysoké kvaliatě
  - navržen pro pořeby DTP (Desktop Publishing)
  - používá různé typy bezztrátové komprese
  - soubory mají velkou velikost
- RAW
  - bezztrátový formát pro uložení snímků z digitálních fotoaparátů
  - pro ukládání dat přímo z CCD snímače
  - různé varianty RAW pro různé typy fotoaparátů, formát není nikde specifikovaný
  - zvlášť uložena jasová složka a barevné složky
  - umožňuje následnou lepší korekci expozice, vyvážení bílé aj.
- JPEG
  - formát se ztrátovou kompresí, určený především pro ukládání fotografií na webu
  - pro obrázky s velkým počtem barev a s barevnými přechody
  - nevhodný pro obrázky s velkými s jednobarevnými plochami a ostrými hranami
  - pro ztrátovou kompresi se nepoužívá jako pracovní formát pro úpravu obrázků (každým uložením souboru se snižuje kvalita obrázku)
  - při ukládání umožňuje nastavit stupeň komprese (kvalita obrázku)

Pixel

- jinak také obrazový bod, představuje nejmenší, elementární (nedělitelný) prvek obrazu (vznikl zkrácením z picture element). Je charakterizován svou barvou (z fyzikálního hlediska frekvencí záření), jejíž zápis je dán použitým barevným modelem. Pixely jsou většinou uspořádány v dvojrozměrné matici, která tvoří výsledný obraz.

DPI

- body na palec (dots per inch), je údaj určující počet obrazových bodů (pixelů), které se vejdou do délky jednoho palce. Tak vyjadřují výstupní, fyzické rozlišení obrazu. Čím je tato hodnota vyšší, tím je obraz detailnější. Hodnota DPI charakterizuje obrázek o konrétních fyzických rozměrech, pokud nedojde k jeho zmenšení, počet bodů se nemění, a tím ani jeho velikost v paměti.

Barevná hloubka

- je veličina vyjadřující objem dat, která popisují barvu jednoho pixelu. Uvádí se v bitech a prakticky značí počet barev, které může pixel, a tím i obraz, mít (n bitů = 2n barev). Čím je vyšší, tím může být obraz reálnější, na druhou stranu, zabírá více místa v paměti.

Počítačová 3D grafika

- Je to speciální část počítačové grafiky, která pracuje s trojrozměrnými objekty.
- Převod 3D objektů do 2D zobrazení se nazývá renderování.
- Nejznámějším využitím počítačové 3D grafiky je vytváření animací (pro tvorbu filmů nebo počítačových her), avšak 3D grafika je využívána i ve vědě a průmyslu (například pro počítačové simulace nebo trojrozměrné zobrazení orgánů).

3D modelování

Proces tvarování a vytváření 3D modelu, který může být reprezentován několika způsoby. Modely mohou být vytvořeny na počítači člověkem pomocí modelovacího nástroje, podle dat získaných měřicím přístrojem z reálného světa nebo na základě počítačové simulace.

- architektura, interiérový design
- nábytek, solitéry, katalogy zařizovacích předmětů
- průmyslový design
- šperkařství

Renderování

Rendering je vykreslení dvourozměrného obrazu na základě modelu scény a dalších informací (polohy pozorovatele, textur, osvětlení a stínování).
Simulují se zejména tyto vlastnosti obrazu:

- Stínování – kolísání barvy a jasu povrchu v závislosti na osvětlení
- Texturování – dodání realistického vzhledu povrchu modelu

- Mlha – tlumení světla při průchodu atmosférou
- Stíny – důsledek zakrytí zdroje světla jiným objektem
- Odraz světla – zrcadlové nebo velmi lesklé reflexe
- Průhlednost – šíření světla skrze objekty bez zkreslení
- Hloubka ostrosti – objekty vzdálené od objektu v centru pozornosti se jeví nezaostřené
- Pohybové rozostření – rychle pohybující se objekty se jeví rozmazané
- Nefotorealistické zobrazování – vykreslování scény v uměleckém stylu, který má připomínat malování nebo kreslení

Barevné modely

RGB model

Barevný model RGB je aditivní barevný model (složky barev se sčítají a vytváří světlo o větší intenzitě) založený na lidském vnímání barev. Každá barva je dána intenzitou tří složek: červené, zelené a modré (v tomto pořadí), což jsou zároveň barvy, které vnímá lidské oko. Používá se všude, kde je světlo vyzařováno a přijmáno, tedy u monitorů a digitálních fotoaparátů a kamer. Každý pixel je pak v takovém zařízení složen ze tří subpixelů, přičemž každý z nich vyzařuje světlo příslušející jedné složce. Uspořádání těchto subpixelů na ploše se liší podle zařízení, u vyzařujících zařízení jsou většinou vedle sebe, tak blízko, že je lidské oko od sebe nerozezná.

Hexadecimální kódy barev v modelu RGB

- Černá: #000000
- Bílá: #ffffff
- Červená: #ff0000
- Zelená: #00ff00
- Modrá: #0000ff

CMYK model

Barevný model CMYK je založen na subtraktivním míchání barev, kdy se bílý papír zakrývá inkousty a tím se omezujeme barevné spektrum, které se od povrchu papíru odráží. Proto se mu říká subtraktivní - odčítací míchání barev. Teoreticky by pro generování všech barev stačilo míchat inkousty tří barev CMY - Cyan (azurová), Magenta (purpurová), Yellow (žlutá). V praxi se ale používá ještě čtvrtá barva blacK (černá), která pomáhá tisknout typicky černý text, zlevňuje tisk a pomáhá míchat tmavé odstíny. Barevný model CMYK je používán při tisku a potřebuje vnější světlo pro generování bílé barvy odrazem od papíru.

Multimédia

Co to je multimediální informace

- často spojení pohyblivého obrazu s kvalitním zvukem a počítačem jako řídícím systémem za účelem zprostředkování nějaké informace, obecně se jedná o spojení alespoň dvou informačních složek: zvuk, obraz, text, grafika, animace a interaktivita
- Multimédia jsou oblast informačních a komunikačních technologií, která je charakteristická sloučením audiovizuálních technických prostředků s počítači či dalšími zařízeními. Jako multimediální systém se označuje souhrn technických prostředků (např. osobní počítač, zvuková karta, grafická karta nebo videokarta, kamera, mechanika CD-ROM nebo DVD, příslušný obslužný software a další), který je vhodný pro interaktivní audiovizuální prezentaci.

Význam multimédií

- názornost daná interaktivitou s uživatelem, rychlost a přehlednost předání informace

Historie multimédií(90. léta 20. století)

- Od počátku 90. let minulého století se začalo používat označení multimediální aplikace nebo multimediální software, které využívaly kombinace textových, obrazových, zvukových či animovaných nebo filmových dat. V roce 1991 vydalo konsorcium pod vedením společnosti Microsoft specifikaci standardního multimediálního počítače (MPC). Ta byla v dalších letech několikrát aktualizována, dnes jsou prakticky všechny osobní počítače multimediální

Dvě pojetí multimédií

- multimediální služby - zpracování a přenos informací – konverzační služby, vyhledávací služby, distribuční služby
- multimediální technologie - souhrn postupů a prostředků pro zpracování, archivaci a přenos multimediálních informací

Dva druhy nosičů multimédií

- statická média (text, grafika)
- dynamická média (animace, zvuky, interaktivita ...)

Jak vypadá multimediální systém

- počítač, zvuková karta, grafická karta s reproduktory, videokarta, mechanika DVD, obslužný software, kamera, vizualizér, dotyková tabule, projektor, hlasovací zařízení apod.
- Jako multimediální systém se označuje souhrn technických prostředků (např. osobní počítač, zvuková karta, grafická karta nebo videokarta, kamera, mechanika CD-ROM nebo DVD, příslušný obslužný software a další), který je vhodný pro interaktivní audiovizuální prezentaci.

Informace mají velkou kapacitu

- z důvodu přenosu a archivace se používá komprese, vyjmenovat kompresní algoritmy ztrátové (jpeg, mpeg, wmv, mp3, wma, aac) a bezeztrátové komprese (gif, png, tiff), co to je kodek (program, který dokáže transformovat datový proud – stream, např. DivX, Xvid)
- **Komprese dat** (také komprimace dat) je speciální postup při ukládání nebo transportu dat. Úkolem komprese dat je zmenšit datový tok nebo zmenšit potřebu zdrojů při ukládání informací. Obecně se jedná o snahu zmenšit velikost datových souborů, což je výhodné např. pro jejich archivaci nebo při přenosu přes síť s omezenou rychlostí (snížení doby nutné pro přenos).
  - **Ztrátová komprese** – při kompresi jsou některé informace nenávratně ztraceny a nelze je zpět rekonstruovat. Používá se tam, kde je možné ztrátu některých informací tolerovat a kde nevýhoda určitého zkreslení je bohatě vyvážena velmi významným zmenšením souboru. Používá se pro kompresi zvuku a obrazu (videa), při jejichž vnímání si člověk chybějících údajů nevšimne nebo si je dokáže domyslet (do určité míry).
    - Formáty:
      - JPEG, JPEG 2000, MPEG, MP3, WMA, AAC
  - **Bezeztrátová komprese** – obvykle není tak účinná jako ztrátová komprese dat. Velkou výhodou je, že komprimovaný soubor lze opačným postupem rekonstruovat do původní podoby. To je nutná podmínka při přenášení počítačových dat, výsledků měření, textu apod., kde by ztráta i jediného znaku mohla znamenat nenávratné poškození souboru.
    - Formáty:
      - GIF, PNG, JPEG 2000, TIFF
- **Kodek** - (složenina z počátečních slabik slov „kodér a dekodér“, respektive komprese a dekomprese, je zařízení nebo počítačový program, který dokáže transformovat datový proud (stream) nebo signál. Kodeky ukládají data do zakódované formy (většinou za účelem přenosu,

uchovávání nebo šifrování), ale častěji se používají naopak pro obnovení přesně nebo přibližně původní formy dat vhodné pro zobrazování, případně jinou manipulaci. Kodeky jsou základní součástí softwaru pro editaci (střih) multimediálních souborů (hudba, filmy) a často se používají pro videokonference a distribuci multimediálních dat v sítích (streamování).

Co je to digitalizace signálu

- vzorkování analogového signálu a kódování – u zvuku vzniká kvalitní signál, když zvolená frekvence vzorkování je alespoň dvojnásobkem nejvyšší frekvence ve zvuku (ucho slyší max. 20kHz – frekvence vzorkování 44kHz)
- Digitalizace starých zvukových a obrazových nahrávek znamená převod těchto informací do digitálního (číslicového) tvaru. Umožnil ji zejména prudký rozvoj digitálního způsobu nahrávání zvuku i obrazu na konci 20. století a také velký rozvoj digitálních nosičů dat, zejména pak CD disků, později DVD disků, číslicových pamětí a komprimovaných formátů obecně. Digitalizace těchto analogových dat probíhá již delší dobu.
- Vzorkování
  - Vzorkování se provede tím způsobem, že rozdělíme signál na malé rovnoměrné úseky a z každého úseku odebereme jeden vzorek (nejčastější hodnotu ve vzorku).Při vzorkování tak z původního signálu ztratíme mnoho detailů, proto aby vznikl kvalitní signál potřebujeme signál alespoň o dvojnásobně větší frekvenci.

Jak vzniká zvuková informace

- vytvořená počítačovými nástroji (nahráním pomocí fyzických nástrojů (mikrofon, elektrická kytara))
- snímáním (ripování) nahrávky
- syntézou řízených generátorů – MIDI - (Musical Instrument Digital Interface) je mezinárodní standard používaný v hudebním průmyslu jako elektronický komunikační protokol, který dovoluje hudebním nástrojům, počítačům i dalším přístrojům komunikovat v reálném čase prostřednictvím definovaného sériového rozhraní

Jak vzniká video informace

- snímáním videa kamerou a následná úprava (střih, ozvučení a titulkování)
- snímáním (ripování) nahrávky - Jedná se o techniku získávání dat z CD/DVD média. Abychom ona data dostali do počítače, tak k tomu potřebujeme nějaký nástroj, který nám tato data, zapsaná v jedničkách a nulách, převedou do nějakého běžného formátu. Většina těchto nástrojů používá jako svůj výstup zvukový formát WAV.
- obrazové animace (dynamická interpretace převážně objektů vektorové grafiky)

Formáty pro video

- **avi** – obsahuje jednu či více datových stop (zvuk, video, efekty, text titulků), každá stopa obsahuje mediální tok zakódovaný pomocí speciálního kodeku, zaveden společností Microsoft jako součást multimediální technologie Video for Windows
  - funguje jako multimediální kontejner, který obsahuje jednu nebo více datových stop. Každá stopa ukládá jeden typ dat: zvuk, video, efekty či text (pro zobrazení titulků). Každá stopa také obsahuje digitálně zakódovaný mediální tok (zakódován pomocí specifického kodeku)
- **mpeg** - kódování audiovizuálních informací (obraz se zvukem) pomocí digitálního kompresního algoritmu (mpeg2, mpeg4)
  - kódování audiovizuálních informací (např. film, obraz, hudba) pomocí digitálního kompresního algoritmu.
- **mpeg-1:** Kódování pohyblivého obrazu a přidruženého zvuku pro digitální datové nosiče s rychlostí přenosu 0,9 až 1,5 Mbitu/s. Standard pro kódování zvuku zahrnuje také oblíbený zvukový kompresní formát Layer 3 (MP3).
- **mpeg-2:** Všeobecné kódování pohyblivého obrazu a přidruženého zvuku. Zahrnuje přenosové, obrazové a zvukové kódovací standardy pro vzduchem šířené televizní vysílaní ATSC a DVB, digitální satelitní TV přenos, digitální kabelový TV signál a (s určitými změnami) disky DVD Video. Přenosová rychlost se pohybuje od 1,5 Mbitu/s až do 15 Mbitů/s (pro TV signál se používá rychlost 6 Mbitů/s).
- **mpeg-3:** Původně určený pro kódování standardu HDTV, později byl jeho vývoj pozastaven a standard MPEG-3 byl sloučen se standardem MPEG-2.
- **mpeg-4:** Kódování audiovizuálního obsahu s velmi nízkým bitratem. Rozšiřuje formát MPEG-1 o podporu audio/video „objektů“, 3D obsahu, kódování s nízkou rychlostí přenosu.
- **mpeg-7:** standard pro popis dat s multimediálním obsahem, čímž se zcela odlišuje od předchozích (neříká totiž, jak data kódovat). Tento formát by měl sloužit k rychlému a efektivnímu vyhledávání multimediálních dat dle klíče.
- **mjpeg** - každý snímek je zde komprimován zvlášť podle standardu JPEG, MJPEG samotný nemá oficiální standard
  - (Motion JPEG) je formát videa, který je nejčastěji používán v digitálních a IP kamerách. Každý snímek je zde komprimován zvlášť podle standardu JPEG. MJPEG samotný nemá oficiální standard. Implementuje jej však několik kodeků. Z absence standardu vyplývají problémy možné vzájemné nekompatibility těchto kodeků. Všechny zkomprimované snímky jsou zde klíčové, což má výhodu rychlého posunu ve videu. Tato vlastnost jej činí vhodným pro použití při úpravách videa. Nevýhodou je větší velikost výsledného videa

Aplikace pro střih videa

- (Adobe Premiere, Virtual Dub, Unlead Media Studio, Pinnacle Studio)
- Střihový program umožňuje editovat (upravovat) záběry, které jsme naimportovali (tento proces nazýváme zachytávání na disk počítače) jako videosoubory AVI (případně soubory s jinou příponou - podle typu střihové karty). U jednotlivých záběrů pak v pracovním prostředí střihového programu můžeme měnit jejich délku (a to s přesností na jeden snímek - 1/25 vteřiny), jejich rychlost (záběry můžeme zrychlovat nebo zpomalovat, nebo obraz zcela zastavit) a také je možné měnit i směr pohybu záznamu - záběr pouštíme pozpátku. Dále můžeme vkládat různé efekty mezi záběry (přechody) počínaje jednoduchou prolínačkou až po různé 3D prolínání záběrů

Přehrávače multimediálních formátů

- (Windows Media Player, BSPlayer, QuickTime, PowerDVD, RealPlayer)
- Nejdůležitějším hlediskem je podpora přehrávání různých formátů obrazu a zvuku, která by však měla být ve všech přehrávačích bezproblémová. Podpora multimediálních formátů se totiž řídí jak kodeky obsaženými přímo v přehrávači, tak i kodeky instalovanými v počítači, třeba pomocí některého z instalačních balíčků. Funkce pro přehrávání by měly zahrnovat nejen základní navigaci v přehrávaném souboru, ale rovněž například také rychlý posun vpřed a vzad, přeskakování kapitol, práci s titulky, playlisty atd. Hodnocení ovládání programů zohledňuje vzhled a ergonomii uživatelského rozhraní, rozmístění ovládacích prvků a dostupnost důležitých funkcí. Zvýhodněny jsou také programy přeložené do češtiny. Další body v hodnocení jsou za doplňkové funkce pro přehrávání videa a hudby.

Uplatnění multimedií

- prezentace – např. PowerPoint, e-learningové multimediální vzdělávací programy – často na DVD-ROM nebo na internetu, herní software, simulační software atd...)
- V poslední době jsou to hlavně: výukové programy, prezentační produkce, informační systémy, encyklopedie, multimediální prezentace a v neposledním případě také hry. Právě díky hrám se tato oblast výpočetní techniky velice rychle rozšířila i do domácností.

U výukových programů jde hlavně co nejpřitažlivěji naučit třeba počítat, číst, poznávat zvířata, a nyní jde hlavně o výuku cizích jazyků. Existují různé encyklopedie, kde máte mimo textu mnoho obrázků, videí, animací i zvuků.

Aplikační software

Odkaz na FTP

http://maturitait4.iunas.cz/public-files/maturitni%20otazky/pvy/22.%20aplikacni_software.doc

Co je a co není aplikační software

Aplikační software (zkráceně aplikace) je v informatice veškeré programové vybavení počítače (tj. software), které umožňuje provádět nějakou užitečnou činnost (řešení konkrétního problému, interaktivní tvorbu uživatele) Aplikace využívají pro interakci s uživatelem grafické nebo textové rozhraní, případně příkazový řádek. Mezi aplikace nepatří systémový software (jádro a další součásti operačního systému, např. služba Windows, démon).

Rozdělení aplikačního software podle účelu

Aplikační software můžeme rozdělit do různých skupin. Některé aplikace mohou patřit i do více skupin a skupiny se mohou překrývat, protože rozdělení nemusí být jednoznačné (seřazeno podle abecedy):

-Antivirové programy - AVG, Avast, ESET Smart Security, Norton

-Databázové systémy - Oracle, Firebird, Microsoft Access

-Grafické editory - Adobe Photoshop, GIMP

-Hry - :D

-Internetové prohlížeče (browsery) - Internet Explorer, Opera, Mozilla Firefox

-Kancelářské balíky - Microsoft Office, OpenOffice

-Pomocné programy – různé utility

-Poštovní programy - Microsoft Outlook, Mozilla Thunderbird

-Prezentační programy - Microsoft Powerpoint, OpenOffice Presenter

-Výukové programy - různé vyukové programy pro jiné programy

-Tabulkové procesory - Microsoft Excel, OpenOffice Calc

-Vývojové nástroje - Microsoft Visual Studio, PSPad

Softwarová licence, příklady

Softwarová licence je v informatice právní nástroj, který umožňuje používat nebo rozšiřovat software, který je chráněn zákonem. V České republice se jedná o Autorský zákon.

Tvorbu přenechávají autoři software právníkům.

Často se používá licence ve formě EULA.

Autor může licenci měnit.

EULA - Je licence pro koncového uživatele softwaru určující, co uživatel smí a nesmí dělat. Je možné, aby byl zdrojový kód open source, ale výsledný produkt už spadá pod EULA, kde se hovoří o zákazu editace a šíření tohoto programu (např. Mozilla Firefox). Některé programy v závěru EULA při instalaci zmiňují instalaci dodatečného škodlivého programu, jedná se o tzv. grayware (např. vyskakující pop-up okna, otravuje uživatele).

Některé způsoby distribuce

- Adware – programy, které mají integrovanou nepříjemnou reklamu
- Cardware – autor žádá uživatele o zaslání pohlednice
- Donationware – autor žádá uživatele o příspěvek na své konto nebo na dobročinné účely
- Demo – funkčně nebo časově (trial) omezená verze komerčního software, která se šíří zdarma
- Freeware – software, jehož užívání je naprosto zdarma
- Shareware – software, který lze volně distribuovat a zdarma vyzkoušet, pro další používání je třeba zaplatit
- Abandonware či orphanware – program, který již není výrobcem prodáván ani podporován, leč je tolerováno jeho šíření neoficiálními

kanály<

Kancelářský balík

Aplikace se může skládat z několika programů, případně je několik aplikací spojeno do skupiny, kterou označujeme jako aplikační balíky (anglicky application suite), mezi které patří například kancelářské balíky OpenOffice.org a Microsoft Office.

Co to je kontingenční tabulka v tabulkovém procesoru

Kontingenční tabulka se užívá k přehledné vizualizaci vzájemného vztahu dvou statistických znaků.

Řádky kontingenční tabulky odpovídají možným hodnotám prvního znaku, sloupce pak možným hodnotám druhého znaku. V příslušné buňce kontingenční tabulky je pak zařazen počet případů, kdy zároveň měl první znak hodnotu odpovídající příslušnému řádku a druhý znak hodnotu odpovídající příslušnému sloupci. Například prvním znakem může být pohlaví člověka a druhým znakem měsíc jeho narození. Kontingenční tabulka o 2 řádcích (žena, muž) a 12 sloupcích (leden, únor,..., prosinec) pak popisuje počty výskytů všech kombinací pohlaví a měsíce v nějakém souboru sledovaných jedinců.

Je možné, aby jeden řádek či sloupec odpovídal více možným hodnotám znaku. To se děje v případě, kdy znak nabývá některých hodnot příliš zřídka, takže je vhodné spojit více možných hodnot.

Relativní a absolutní adresování v tabulkovém procesoru

Relativní adresa je odkaz přizpůsobující se nové pozici (U Excelu např. A1, B8, T3, ...).

Absolutní adresa je odkaz směřující stále na stejné buňky (U Excelu pomocí znaku $, např. $C8 pro absolutní sloupce, C8 pro absolutní sloupce i řádky.

Relativní odkazy: Pokud vytvoříte vzorec, jsou odkazy na buňky nebo oblasti obvykle založeny na jejich umístění vzhledem k buňce, která obsahuje daný vzorec. V následujícím příkladu obsahuje buňka B6 vzorec =A5 a v aplikaci Microsoft Excel bude vyhledána hodnota jednu buňku nad a jednu buňku doleva od B6. Toto je takzvaný relativní odkaz. Pokud zkopírujete vzorec s relativními odkazy, upraví aplikace Excel automaticky odkazy ve vloženém vzorci tak, aby odkazovaly na jiné buňky vzhledem k umístě ní vzorce. V následujícím příkladu byl vzorec v buňce B6 (=A5, cožje buňka o jednu buňku nad a doleva od buňky B6) zkopírován do buňky B7. Vzorec v buňce B7 se upravil na =A6, který odkazuje na buňku, která je jednu buňku nad a doleva od buňky B7.

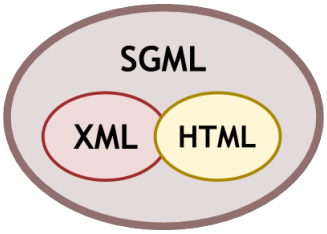
Absolutní odkazy: Pokud nechcete, aby aplikace Excel upravovala odkazy při kopírování vzorce do jiné buňky, použijte absolutní odkaz. Jestliže například vzorec násobí buňku A5 buňkou C1 (=A5*C1) a daný vzorec zkopírujete do jiné buňky, upraví se oba odkazy. Absolutní odkaz na buňku C1 vytvoříte umístě ním znaku dolaru ($) před části odkazu, které se nemě ní. Absolutní odkaz například na buňku C1 vytvoříte přidáním znaků dolaru do vzorce podle následujícího příkladu: =A5*C1

XML

- obecný jazyk SGML
- jak vypadá XML dokument, využití v praxi
- XML a layout, co to je eXtensible Stylesheet Language (XSLT styl)
- co to je definice typu dokumentu (DTD)

Obecný jazyk SGML

XML patří do skupiny tzv. **značkovacích jazyků**. Tyto jazyky umožňují označovat části textu **značkami** (tagy).



Vztah mezi SGML, XML a HTML

Předchůdcem XML byl jeden z prvních značkovací jazyků **SGML** (Standard Generalized Markup Language). Ten byl přijat v roce 1986 jako ISO norma. Asi prvním známých značkovacím jazykem byl jazyk GML, který byl vytvořen při práci na systému pro uchovávání a následné využití právních textů pro IBM. Autoři se museli vypořádat s nekompatibilitou jednotlivých systémů a programů a nejsnazší cesta vedla právě přes vytvoření nějakého obecného značkovacího jazyka. Tak vzniklo SGML.

Vlastnosti

- umožňuje definovat vlastní **značkovací jazyky** (pomocí DTD viz dále)
- umožňuje nastavit spoustu volitelných parametrů (maximální délku názvů značek, určit znaky sloužící jako oddělovače značek od textu a **mnohem mnohem** víc)
- velice obecný a složitý/komplexní

Na jeho základech stojí mnoho dalších značkovacích jazyků (HTML, XML, atd.).

Po letech používání jazyka SGML se ukázalo, že se ve skutečnosti používá jen část jeho skutečných možností. Tato část byla v roce 1996 vybrána jako základ pro nově vznikající jazyk XML.

Jak vypadá XML dokument

Na začátku každého XML dokumentu se může nacházet (*měla by, je to slušnost*) tzv. **XML deklarace**. V základu vypadá takto:

```
<?xml version="1.0" ?>
```

Deklarace sděluje programu, který bude dokument zpracovávat, některé důležité informace. V uvedeném případě se jedná o informaci, že dokument je zapsán podle XML verze 1.0 (aktuální je verze 1.1). Deklarace **musí** obsahovat alespoň informaci o použité verzi XML. Poté je možné ještě určit použité kódování:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Uvedené informace musí být uvedeny přesně v tomto pořadí. Když není hodnota `encoding` uvedena, je předpokládano použití výchozího kódování **ISO 10646**. Jedná se o 32-bitovou znakovou sadu, která je schopná pojmout všechny znaky dnes používaných jazyků. Je téměř shodná se standardem Unicode (zdroj).

Za XML deklarací se nechází **kořenový element**. Kořenový element je nejvyšším elementem dokumentu, všechny ostatní elementy jsou do něj vnořeny. XML dokument musí obsahovat alespoň jeden element (třeba jenom kořenový). Kořenový element je **jenom jeden**.

```
<?xml version="1.0" encoding="UTF-8" ?>
<trida name="it4" startYear="2008" endYear="2012">
        <tridniUcitel name="Ivana Šmejkalová" alias="SJ" />
        <zaci>
                <zak>
                        <jmeno>Mirek</jmeno>
```

```
                        <prijmeni>Balog</prijmeni>
                        <cislo>1</cislo>
                        <znamky year="2008">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>

                        <znamky year="2009">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>
                </zak>

                <zak>
                        <jmeno>Jakub</jmeno>
                        <prijmeni>Beneš</prijmeni>
                        <cislo>2</cislo>
                        <znamky year="2008">
                                <znamka lesson="tev" grade="1">
                                <znamka lesson="cjl" grade="2">
                                <znamka lesson="pvy" grade="1">
                        </znamky>
                </zak>
                <!-- atd. -->
        </zaci>
</trida>
```

Kořenovým elementem je zde element `trida`. Ten obsahuje **atributy** `name`, `startYear` a `endYear`. Do kořenového elementu jsou vnořeny další elementy - element `tridniUcitel` a element `zaci`.

Každý element je tvořen **uvozovací** a **koncovou značkou** (tagem) a svým obsahem. Koncový tag obsahuje před jménem tagu lomítko (`</tag>`). Uvozovací tag může obsahovat atributy (`name="it4"`, `startYear="2008"`, apod.)

```
<mujTag>Obsah elementu</mujTag>
```

Element nemusí obsahovat žádný obsah.

```
<mujTag></mujTag>
```

Tento zápis je ale zbytečně zdlouhavý a komplikovaný, proto nám XML umožňuje zapsat element bez obsahu tímto způsobem:

```
<mujTag />
```

Zde je uvozovací a koncový tag spojen do jednoho. Třída bez žáků by se tak zapsala takto:

```
<trida name="Prázdná třída" />
```

Tagy se **nesmí** křížit. Toto je špatný zápis:

```
<a><b></a></b>
```

Správně má být:

```
<a><b></b></a>
```

Jazyk XML je velice přísný v oblasti správného zápisu. Stačí jedna drobná chyba v zápisu a zpracování dokumentu skončí chybovou zprávou. Tímto se liší např. od HTML, kde prohlížeče spoustu chyb ignorují a snaží se uhodnout, jak to mělo být správně. Pokud je dokument správně napsán (odpovídá syntaxi (správnému XML zápisu)), říkáme, že se jedná o **well-formed** dokument.

Součástí dokumentu můžou být také instrukce pro nějaký jiný program. Můžeme tak do dokumentu zařadit odkaz na styl definující zobrazení v prohlížeči, formátovacímu programu můžeme naznačit, kde má zalomit stránku, nebo vložit do dokumentu kód některého skriptovacího jazyka. Tyto instrukce XML parser ignoruje, předá je nadřazené aplikaci záleží na ní, zda je nějak využije. Syntaxe je následující:

```
<?«identifikátor» «data»?>
```

Identifikátor je identifikátor externího programu, např. `php`. Můžeme tak do dokumentu vložit kód jazyka PHP, který nám do dokumentu vypíše dnešní datum.

```
<dokument>
    <datum>Dnešní datum je <?php echo date("d.m.Y");?></datum>
    <para>Nějaké další informace</para>
</document>
```

Pomocí instrukcí je možno připojit k XML dokumentu i externí styl:

```
<?xml-stylesheet tref="styl.css" type="text/css"?>
```

Do dokumentu také můžeme vložit libovolný znak pomocí tzv. entit. Znaková entita má tvar `&#x«kód znaku»;`, kde «kódznaku» je kód znaku ze znakové sady ISO 10646 (Unicode – UCS) zapsaný v šestnáctkové soustavě. Můžeme použít i tvar `&#«kód znaku»;`, kde je «kód znaku» zapsán v desítkové soustavě.

Využití v praxi

XML lze považovat za poměrně univerzální formát, který nalézá uplatnění v mnoha oblastech.

- **B2B, business-to-business aplikace** (výměna informací mezi obchodními partnery v elektronickém formátu)
- **Inteligentní webové stránky** (Možnost definice vlastních značek, které přesně vyznačí význam jednotlivých částí stránky, má pozitivní efekt na přesnost a rychlost vyhledávání informací. Obecný model umožňuje vývoj stránek pro mnoho zařízení s rozdílnými schopnostmi PC, mobilní telefony, WebTV nebo třeba herní konzole.)
- **Metadata** (Pro vyhledávání, ale hlavně pro klasifikaci dokumentů, je užitečné o nich znát co nejvíce metadat.)
- **Elektronické publikování** (Často potřebujeme jeden dokument v několika různých formátech jako tištěnou knihu, sadu provázaných webových stránek nebo hypertextovou příručku na CD-ROMu. Flexibilita stylových jazyků také umožňuje, aby se z jednoho zdroje generovalo několik druhů dokumentů s různým obsahem, některé údaje v technické dokumentaci jsou například tajné a vytisknou se pouze pro potřeby firmy, přičemž zákazníkům se ze stejného XML dokumentu vygeneruje okleštěná verze.)
- **Univerzální datový formát** (Pro ukládání složitějších parametrů programu se nehodí jednoduché způsoby typu INI soubor. Navíc je poměrně komplikované psát program, který bude načítat jednotlivé parametry z konfiguračního souboru. Pokud by si program ukládal data v XML, mohl by je programátor velice snadno číst pomocí některé z knihoven pro práci s XML.)

Konkrétní použití XML:

- **XHTML** – XML alternativa jazyka HTML.
- **RSS** – Rodina XML formátů, sloužící pro čtení novinek na webových stránkách.
- **SMIL** – Synchronized Multimedia Integration Language, popisuje multimedia pomocí XML.
- **MathML** – Mathematical Markup Language je značkovací jazyk pro popis matematických vzorců a symbolů pro použití na webu.

- **SVG** – Scalable Vector Graphics je jazyk pro popis dvourozměrné vektorové grafiky, statické i dynamické (animace).
- **Jabber** – Protokol pro Instant messaging.
- **SOAP** – Protokol pro komunikaci mezi Webovými službami.
- **Office Open XML, OpenDocument** – Souborový formát určený pro ukládání a výměnu dokumentů vytvořených kancelářskými aplikacemi.
- **a mnoho dalších** (formát pro ukládání TV programu, formáty pro popis 3D scén, popis editace 2D grafiky, konfigurační soubory atd.)

XML a layout

Pomocí XML značek vyznačujeme v dokumentu význam jednotlivých částí textu. Říkáme toto je název výrobku, tohle zase telefonní číslo a tohle je číslo našeho účtu. Dokumenty obsahují mnohem více informací, než kdyby se používalo prezentační značkovaní tohle je tučným písmem Arial o velikosti 12 bodů zarovnané vlevo.

V mnoha případech potřebujeme XML dokument zobrazit na nějakém běžném médiu na obrazovce, na papíře. V tomto případě už samozřejmě chceme přesně ovlivnit, jak se obsah jednotlivých značek zobrazí. XML samo o sobě žádné takové prostředky nenabízí. Existuje však naštěstí hned několik stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu, se říká **styl**.

Výhodou je, že jeden styl můžeme aplikovat na mnoho dokumentů stejného typu. Dosáhneme tak jednotného formátování. Zároveň můžeme na jeden dokument aplikovat několik různých stylů. Jedním stylem vygenerujeme postscriptový soubor pro naše DTP studio, druhým HTML kód pro zařazení na naše webové stránky a třetím třeba jen obsah dokumentu, který pošleme mailem šéfovi.

Stylových jazyků je mnoho, mezi nejznámější patří asi kaskádové styly (CSS). Ty lze použít pouze pro jednoduché formátování, které dobře poslouží pro zobrazení dokumentu na obrazovce v XML editoru nebo v prohlížeči. Pro náročnější aplikace slouží jazyk **XSL** (eXtensible Stylesheet Language).

Co to je eXtensible Stylesheet Language (XSLT styl)

XSL zahrnuje 3 jazyky:

1. XSL Transformace (XSLT): XML jazyk k převádění XML dokumentů,
2. XSL Formátovací Objekty (XSL-FO): XML jazyk pro specifikaci vizuálního formátování XML dokumentů,
3. XML Path jazyk (XPath, česky „jazyk pro cesty v XML“): jazyk k adresování částí XML dokumentu, který ale sám není XML jazykem. Je používán například v XSLT.

Transformace **XSLT** (eXtensible Stylesheet Language Transformations) slouží k převodům zdrojových dat ve formátu XML do libovolného jiného požadovaného formátu, nejčastěji HTML, jiného XML nebo libovolných jiných datových struktur.

K provedení transformace jsou potřeba dva soubory:

- První soubor obsahuje zdrojová data, která budou transformována. Struktura tohoto souboru vyjma obecných vlastností XML není blíže specifikována.
- Druhý soubor obsahuje vzorec pro transformaci a musí být napsán v jazyce XSL.

Odkazování v XML dokumentu

XML umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty navzájem. Nabízí však mnoho možností nad rámec odkazů, které známe z HTML. Můžeme vytvářet i vícesměrné odkazy, které spojují několik dokumentů dohromady. Užitečná je i možnost uložení odkazů zcela mimo dokumenty, kterých se týkají. Tímto způsobem lze vytvářet různé anotace a komentáře k již existujícím stránkám.

Co to je definice typu dokumentu (DTD)

~~DTD (Document Type Definition) je jedna z důležitých vlastností, které převzalo XML z jazyka SGML. Umožňuje definovat vlastní sadu značek (elementy a atributy), které budou XML dokumentu k dispozici. V definici můžeme určit, kde se dané elementy a atributy mohou nacházet a jaký mohou mít obsah či hodnotu. Jestliže XML dokument odpovídá všem pravidlům v určitém DTD, říkáme, že je dokument **validní**.~~

~~DTD např. umožňuje určit, že element `trida` smí obsahovat pouze elementy `tridniUcitel` a `zaci` a že jeho jedinými povolenými atributy jsou `name`, `startYear` a `endYear`. Dále můžeme určit povolené atributy pro element `tridniUcitel` a také říct že se v elementu `trida` může vyskytovat pouze jednou. A nakonec určíme, co smí obsahovat element `zaci`.~~

~~Jedná se v podstatě o pravidla pro daný XML dokument. DTD nám umožňuje říct, co smí XML dokument obsahovat a v jaké podobě.~~

Definice typu dokumentu (DTD) říká, které elementy a atributy můžeme v dokumentu použít. Navíc je zde definováno, v jakých vzájemných vztazích mohou být jednotlivé elementy použity. DTD je tedy užitečný nástroj, který nám umožní hlídat, zda mají naše dokumenty správnou strukturu. Ve světě se používá mnoho DTD, které vyhovují různým požadavkům. Mezi jedno z nejznámějších patří například DTD DocBook, které definuje elementy a atributy vhodné pro značkování technické dokumentace.

DTD se hodí pro popis formátů, které se používají především pro textové dokumenty. Neobsahuje však nástroje pro kontrolu různých typů dat jako čísla, měnové údaje, údaje o datu a čase. To je přitom velice důležité pro aplikace, které si pomocí XML posílají data spíše databázového charakteru. Pro tyto potřeby existuje několik dalších jazyků, umožňujících určit správné schéma dokumentu.

K jakému DTD se náš XML dokument hlásí oznámíme použitím tzv. **DOCTYPE**, hned za XML deklarací. Pro opakované použití bývá DTD uloženo v samostatném souboru.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE «kořenový element» SYSTEM "«URL»">
```

- kořenový element - jméno elementu, ve kterém bude obsažen celý document, v našem případě `trida`
- URL - adresa nebo jméno souboru, ve kterém je DTD uloženo

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE trida SYSTEM "denifinice-tridy.dtd">
```

Můžeme také DTD zapsat přímo do XML dokumentu:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE kniha [
« ... DTD ... »
]>
```

Syntaxe DTD

DTD obsahuje:

- deklarace elementů
- deklarace atributů
- deklarace entit

Deklarace elementů

- **prázdný element** - nemůže obsahovat text ani další elementy
  - v HTML např. `br` pro nový řádek a `hr` pro vložení horizontální čáry <!ELEMENT br EMPTY>□<!ELEMENT hr EMPTY>
- **element obsahující další elementy** - např. HTML element `<html>` může obsahovat elementy `head` a `body` <!ELEMENT html (head, body)>
  - pokud chceme vyjádřit, že element může obsahovat jenom jeden z uvedených použijeme `|` <!ELEMENT potomek (dcera | syn)>
  - článek vždy obsahuje název, ale nemusí obsahovat autora <!ELEMENT clanek (nazev, autor?)>
  - chceme vyjádřit, že se nějaký element může vyskytovat opakovaně, minimálně však jednou (např. kniha obsahuje minimálně jednu kapitolu) <!ELEMENT kniha (kapitola+)>
  - vraťme se k příkladu se článkem - nyní může článek obsahovat více autorů, nebo také žádného (tj. 0 až N autorů) - použijeme hvězdičku <!ELEMENT clanek (nazev, autor*)>
- **element smí obsahovat pouze text a ne další elementy** - použijeme speciální slovo `#PCDATA` <!ELEMENT em (#PCDATA)>

Deklarace atributů

V XML může mít každý element libovolné množství atributů. Atributy se většinou používají pro připojení různých metainformací k elementům. Deklarace atributů pro element má poměrně jednoduchý tvar:

```
<!ATTLIST «jméno elementu» «deklarace atributů»>
```

Deklarace:

```
<!ATTLIST kniha autor CDATA ...>
```

Nám umožní používat u elementu `kniha` atribut `autor`:

```
<kniha autor="Čapek">
```

Deklarace entit

V XML můžeme předdefinovat libovolné množství entit. Každá bude mít vyhrazen svůj identifikátor.

Např.

```
<!ENTITY quot "&#34;">
```

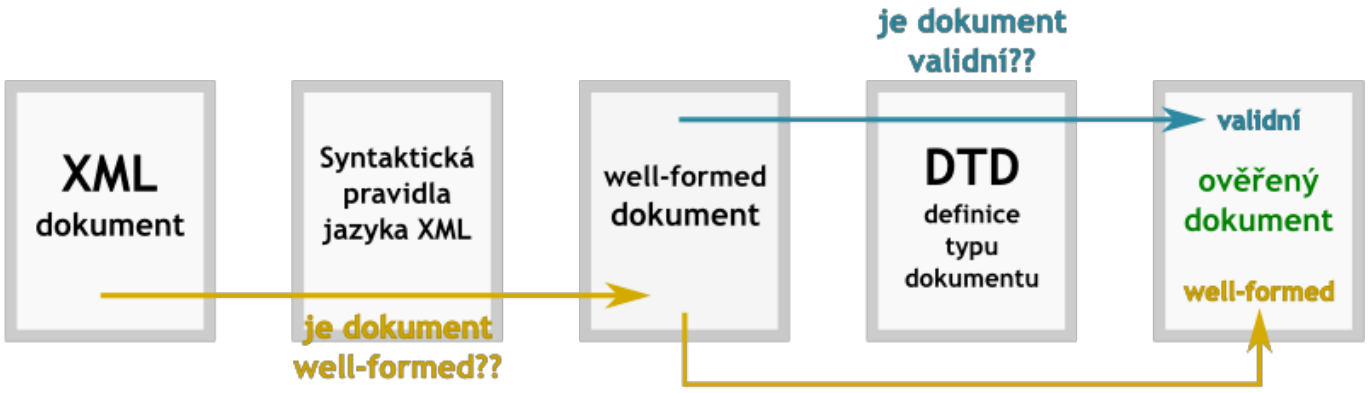
nebo

```
<!ENTITY logo SYSTEM "/obr/logo.eps" NDATA "EPS">
```

Takto definovanou entitu nemůžeme přímo vložit do textu dokumentu a ani by to nemělo smysl. Můžeme na ní však odkázat pomocí atributu, který má typ `ENTITY` nebo `ENTITIES`. Od roku 1999 má naše firma logo, které je na obrázku `<picture name="logo"/>`

Kontrola dokumentu podle DTD

Jak už bylo řečeno, pokud XML dokument odpovídá struktuře popsané pomocí DTD, říkáme o XML dokument, že je **validní**, jinak je **invalidní**. Validita dokumentu je tzv. 2 stupeň správnosti dokumentu. Prvním stupněm je tzv. syntaktická správnost, tj. zda-li je dokument napsán bez syntaktických chyb (neuzavřené tagy a hodnoty, překřížené tagy, překlepy v názvech elementů, atributů, apod.). Pokud je dokument syntakticky správný, říkáme, že je tzv. **well-formed**.



Pro zpracování XML dokumentu se používá tzv. **parser**. Ten načítá XML dokument a poskytuje nám snadný způsob, jak přečíst data z XML dokumentu, a zároveň kontroluje jeho syntaktickou správnost a v případě přítomnosti DTD i validitu dokumentu. Parser bývá k dispozici ve dvou provedeních:

- knihovny pro různé programovací jazyky, které můžeme použít v našich programech
- jednoduché programy, kterým na vstup předáme XML dokument a na výstupu obdržíme případný přehled chyb v dokumentu.

Můžeme také využít parser zabudovaný v prohlížeči, který XML podporuje (většina webových prohlížečů, apod.).

Počítačové periferie

- porty počítače – paralelní, sériové
- vstupní zařízení – klávesnice, polohovací zařízení, snímací zařízení, audiovizuální zařízení
- výstupní zařízení – zobrazovací zařízení, tisková zařízení, audiovizuální zařízení

Periferie

Periferie je obvykle zařízení rozšiřující možnosti použití počítače - je to tedy tzv. počítačová periferie. Počítačová periferie konkrétně slouží ke vstupu a výstupu dat z počítače.

Rozdělení počítačových periferii

- výstupní
  - tiskárna
  - monitor
  - reproduktor
- vstupní
  - klávesnice
  - myš
  - scanner
  - tablet
  - mikrofon
  - webová kamera
  - snímač čárového kódu
  - snímač Braillova písma
  - radiotelevizní karta
- kombinovaná vstupně-výstupní
  - zařízení pro ukládání dat
    - flash paměť
    - vyměnitelný disk
      - pružný disk
      - pevný disk
      - optický disk
        - CD disk
        - DVD disk
      - magnetooptický disk
  - síťová karta
  - zvuková karta

Periferie pro automatizaci

- Periferie řídicího systému

Porty počítače

Paralelní port

- Dnes již moc nepoužívaný port, který sloužíval pro komunikaci se staršími tiskárnami, skenery či dalšími specializovanými zařízeními. Na dnešních počítačích se s ním sice setkáme, ale je to jen kvůli zpětné kompatibilitě, kdybychom stále používali zařízení, které by ho vyžadovalo. Říká se mu LPT.

Sériový port RS-232.

Paralelní port LPT.

Sériový port

- Opět věc dnes již dávno překonaná. Avšak ještě nedávno se přes tento port připojovaly modemy, myši a skoro všechna zbylá externí zařízení. Sice jej nalezneme na dnešních počítačích také, ale opět je zde z čistě nostalgických důvodů. Říká se mu RS-232.

Zdířka pro napájecí kabel

- Není sice port jako takový, ale je nejdůležitější, protože se přes tuto zdířku počítač připojuje do elektrické sítě.

VGA port

- (Video Graphics Array – video-grafické rozhraní) Je to jeden z nejdůležitějších portů, slouží k připojení grafické karty k CRT monitoru či LCD displeji. Přenos dat zde probíhá analogovým způsobem.

HDMI

- je zkratka anglického označení High-Definition Multi-media Interface nekomprimovaného obrazového a zvukového signálu v digitálním formátu. Může propojovat zařízení jako například satelitní televizní přijímač, DVD přehrávač nebo A/V receiver s kompatibilním výstupním zařízením, jako například plazmový televizor.

DVI port

- (Digital Visual Interface – digitální vizuální rozhraní) Toto rozhraní slouží speciálně k připojení LCD displejů, jelikož je zde přenos dat realizován digitálně (tzn. nulami a jedničkami), což je pro LCD výhodnější.

S-Video

- Slouží pro připojení televizoru, jako výstupního zobrazovacího zařízení.

PS/2 port

- Standardní port sloužící výhradně pro připojování klávesnice a myši; proto na dnešních počítačích najdeme právě dva. Většinou jsou pro snadnější orientaci také barevně rozlišeny, zelený je pro myš a fialový pro klávesnici.

USB port

- (Universal Serial Bus - univerzální sériová sběrnice) Novodobý univerzální a nejrozšířenější komunikační port. Právě tento standard dnes již téměř kompletně nahradil své předchůdce, sériový a paralelní port. Je to dáno jeho univerzálností, praktickými rozměry a vysokou rychlostí. Dnes se přes něj připojuje skoro vše: myši a klávesnice, digitální foťáky, přenosné MP3 přehrávače atd.

Firewire

- port Extrémně rychlý komunikační port, používaný převážně na připojení digitálních videokamer.

Audiovstupy/výstupy

- Audiozdířek můžeme mít různý počet (záleží na naší zvukové kartě). I zde se uchytil jistý standard, co se jejich barevného rozlišení týče. Většinou se setkáme s klasickými konektory typu jack (3,5 mm) v provedení: zelený (výstup na přední reproduktory), modrý (nahrávací vstup line-in) a růžový (vstup pro mikrofon). Pokud vaše „zvukovka“ podporuje režim více reproduktorů (např. režim 5.1), bývá zvykem, že výstup na zadní reproduktory je černý a kombinovaný výstup na subwoofer a centrální reproduktor bývá oranžový.

Optický port

- Na zadní stěně PC můžeme nalézt i alternativní speciální výstupy audia jako např. výstup pro optický nebo koaxiální audiokabel.

Game port

- Do této zdířky se připojuje joystick, volant či midizařízení.

Síťový port

- RJ-45 Připomíná o trochu větší telefonní přípojku. Využívá se pro vzájemné propojení (tzv. zesíťování) více počítačů. Ty pak mohou spolu komunikovat. V dnešní době je v domácnostech přes něj čím dál častěji také realizováno tzv. pevné připojení do internetu.

Vstupní zařízení

Klávesnice

- Je vnější zařízení systému PC. Prostřednictvím klávesnice uživatel zadává textové informace, které dále počítač zpracovává.

Některé klávesnice obsahují nestandardní klávesy. Např. aktivují internet apod.

- Je rozdělena do několika logických částí:
  - Alfanumerická: největší část a slouží pro běžné psaní textu
  - Numerická část:zcela vpravo, která obsahuje pouze čísla a matematická znaménka
  - Horní část:oblast funkčních kláves, respektive F1 až F12, v každém programu má tlačítko specifickou funkci
  - Ovládací klávesy: mezi alfanumerickou a numerickou částí, ovládají kurzor
- Každá klávesnice může obsahovat ještě další nestandardní prvky.
- Dále se rozlišují různé typy klávesnic.
  - Standardní: 84 kláves
  - Nestandardní: 101 nebo 102 kláves

Myš

- Jedná se o polohovací zařízení k počítači. Pomocí pohybu naší ruky přenáší pohyb šipky na monitoru. Disponuje obvykle dvěma až třemi tlačítky a ovládacím kolečkem. Vše slouží k snadnějšímu ovládání myši. Typy myší:

- Kuličková myš: nejpoužívanější, méně spolehlivá, jelikož silně reaguje na nečistoty
- Bezdotyková myš: snímání probíhá infračerveným paprskem, není nutná podložka
- Bez datového kabelu: u bezdotykových myší nebývá kabel, přenos probíhá rádiovým signálem

Skener

- Slouží ke snímání obrazu (fotky, obrázky, texty apod.) do počítače.
- Princip: obrazová předloha je osvětlována po řádcích a odražené světlo je snímáno pomocí elektronických prvků citlivých na světlo - CCD. Množství světla je tak převedeno na elektrický signál. U barevných skenerů je předloha snímána třemi barvami.
- Dle způsobu snímání předlohy se dělí:
  - Stolní skenery: zařízení v podobě ležaté krabice
  - Ruční skenery: malá domácí zařízení, uživatel ručně přejíždí přes předlohu, nízká kvalita

Mikrofon

- Vstupní audiozařízení, které do počítače nahrává hlasový vstup.
- Podobně lze k počítači připojit obdobná zařízení, jako např. zesilovač, věž apod.

Výstupní zařízení

Monitor

- Prostřednictvím monitoru s námi počítač komunikuje. Zobrazuje aktuální stav počítače, který je zasílán prostřednictvím grafické karty, která převádí signál z počítače na videosignál.
- Typy:
  - CRT (klasická vakuová obrazovka)
  - LCD (tekuté krystaly)
  - plazmová obrazovka
  - a další, méně obvyklé typy (OLED, SED, atd.)

- Parametry
  - Úhlopříčka
  - Rozlišení obrazovky(px)
  - Obnovovací (vertikální) frekvence(80-100Hz)
  - Doba odezvy
    - Doba odezvy se udává v jednotkách milisekund (ms) – doba, za kterou se bod na LCD monitoru rozsvítí a zhasne, pro pracovní využití je vyhovující doba 2,5 ms (obvykle výrobci udávají parametr podobný, ze šedé do šedé barvy, tudíž skutečná odezva je horší)
  - Vstupy (HDMI, DVI, D-SUB)
  - Ostatní (spotřeba, hmotnost, hloubka monitoru, pozorovací úhly)

Tiskárny

- Umožní vyprodukovat informace do tištěné podoby.
- Typy
  - *Inkoustová tiskárna* (kvalitní a rychlý tisk, vyšší provozní náklady, pomalá, zejména pro domácí užití)
    - Princip tisku je založen na tom, že inkoust je na papír vymršťován velkou rychlostí v podobě kapek.
    - Typy:
      - termální
      - piezoelektrické
      - voskové
  - *Jehličková tiskárna* (nízká kvalita tisku, nevhodné pro tisk grafiky, hlučnost, pomalá,nízká cena tisku)
    - používají k tisku tiskovou hlavu, která se pohybuje ze strany na stranu po listu papíru a přes barvící pásku naplněnou inkoustem se otisknou jehličky na papír. Má to stejnou funkci jako klasický psací stroj, s tím rozdílem, že můžeme vybírat různé druhy písma, nebo popřípadě tisknout obrázky.
    - Tyto tiskárny jsou oproti laserovým, nebo inkoustovým tiskárnám výrazně pomalejší, ale i nadále se využívají například u kasy v supermarketu, kde není třeba vysoké kvality tisku.
  - *Laserová tiskárna* (nejkvalitnější, dražší pořizovací náklady, levný provoz)
    - pracuje na podobném principu jako kopírka. Laserový paprsek vykresluje obrázek na světlocitlivý válec, na jehož povrch se poté nanáší toner. Ten se uchytí jen na osvětlených místech, obtiskne se na papír a na závěr je k papíru tepelně fixován.

Reproduktory

- Jsou připojeny ke zvukové kartě a může je nahrazovat např. minivěž apod.

Dataprojektor

- Slouží pro prezentaci více osobám (školení apod.). Je připojen k videokartě počítače a vysílá na zeď nebo plátno.

Interaktivní tabule

- Systém pracující podobně jako dataprojektor, ale k dispozici je tzv. interaktivní ukazovátko. To funguje podobně jako myš.

Webkamera

- Slouží k přenosu obrazu druhé osobě. Lze ji zahrnout i do vstupního zařízení.

Externí odkazy

Paralelní port Sériový port Počítačová tiskárna Monitor

Počítačová síť

Dotazované požadavky u maturity:

- co to je počítačová síť, důvod vzniku
- rozdělení počítačových sítí podle velikosti, rozlohy a účelu (LAN, MAN, WAN, VPN)
- rozdělení počítačových sítí podle topologie [bus (sběrnice), star (hvězda), ring (kruh)]
- rozdělení počítačových sítí podle technologie komunikace a řízení přístupu (Ethernet (verze), Token Ring)
- rozdělení počítačových sítí podle komunikačního protokolu (TCP/IP, IPX/SPX, AppleTalk)
- rozdělení počítačových sítí podle technologie zapojení [Wired (metalické), optical (optické), wireless (bezdrátové)]

Definice sítě:

- Síť sleduje komunikaci koncových uzlů a může fungovat několika způsoby.
- Může jít o soustavu vzájemně propojených sítí,či může jít o množinu vzájemně propojených aktivních prvků (směrovačů).
- **Počítačová síť** (computer network) je souhrnné označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači.
- Umožňuje tedy uživatelům komunikaci podle určitých pravidel, za účelem sdílení využívání společných zdrojů nebo výměny zpráv.

V poslední době jsou všechny sítě postupně spojovány do globální celosvětové sítě internet, která používá sadu protokolů TCP/IP.

- **Počítačová síť se skládá z aktivních a pasivních síťových prvků.**
  - **Mezi pasivní síťové prvky** patří kabeláž a konektory.
  - **Mezi aktivní síťové prvky** patří síťová karta, switch, router, firewall, apod.

Existuje několik typů sítí dle rozsahu,velikosti a účelu:

PAN

- Mají nejmenší rozlehlost, používají se pro propojení osobních elektronických zařízení.
- Jejich cílem není co nejvyšší přenosová rychlost, ale spíše odolnost proti rušení, nízká spotřeba energie či snadná konfigurovatelnost.
- Dosah většinou jen několik metrů (Wi-Fi, Bluetooth, IrDa).

LAN

- Většinou je tato síť používána v rámci jedné budovy či několika budov, vzájemně spojenými.
- Má za úkol sdílení zařízení a služeb (síťové tiskárny, paměťová media…) a umožňovat vzájemnou komunikaci mezi uživateli.
- Používá se kroucená kabeláž, optické vlákno, bezdrátová síť.
- Rychlost 10Mb/s až 10Gb/s.

MAN

- Používají se k propojení sítí LAN v rámci městské zástavby.
- Přenáší data, hlas a obraz.
- Důraz je kladen na vysokou rychlost a spolehlivost.
- Mohou být soukromé i veřejné.

WAN

- Jsou užívány na velkou vzdálenost mezi koncovými uzly či stanicemi – lokální nebo metropolitní sítě.
- Síť obsahuje páteřní rozvody přenosu, které určují rychlost celosvětové sítě – typickým příkladem je Internet.
- Využívají se optická vlákna i bezdrátová technologie.
- Rychlosti 65kb/s až 10Gb/s.

Topologie sítí:

Sběrnicová:

- Tato topologie patří k nejstarším, všechny stanice jsou připojeny na jedno přenosové médium. Dnes už se příliš nepoužívá.
- Výhody: nezávislost stanic na výpadku jiné stanice, levné náklady takového řešení, snadné všesměrové vysílání.
- Nevýhody: při přerušení kabelu úplný výpadek sítě, nutnost vyřešení přístupu stanic k médiu (kdo bude vysílat).

Kruhová

- Označuje logické zapojení, při němž je každý uzel spojen se dvěma dalšími tak, aby společně vytvořily kruh.
- Výhody: nevznikají kolize, jednoduchý přenos dat.
- Nevýhody: data musí projít přes všechny uzle, což zvyšuje riziko poruchy, přerušením kruhu vzniká problém.

Hvězdicová

- Tato topologie je dnes jednoznačně nejpoužívanější. Její myšlenka spočívá v tom, že existuje centrální prvek, který spojuje všechny prvky. Dříve tím centrálním prvkem býval počítač, dnes je aktivní prvek (rozbočovač nebo směrovač).
- Výhody: lehce rozšiřitelná struktura, výpadek libovolné stanice neznamená výpadek celé sítě.
- Nevýhody: velké množství kabelů, při výpadku centrálního prvku nefunguje celá síť.

Stromová

- Vychází z hvězdicové topologie. Používá se především v rozsáhlých počítačových sítích ve velkých firmách.
- Jednotlivé hvězdice často představují jednotlivá oddělení firmy, patra budovy nebo celé budovy.
- Výhody: lze oddělovat provoz

Metody řízení přístupů lze rozdělit na:

Řízené (deterministické)

- Je definován jednoznačný algoritmus určující, v jakém pořadí mohou stanice na medium přistupovat.
- Na medium nebude nikdy přistupovat více stanic současně. Řízené metody se dále ještě rozdělují na centralizované a decentralizované.
- Centralizované – je vyhrazena jedna stanice, která ostatním přiděluje přístupy.

Decentralizovaná (distribuovaná)

- stanice si předávají pověření (metoda logického kruhu), nebo je pravidelně vysílán rezervační rámec a jednotlivé stanice si rezervují pořadí vysílání.

Neřízené (nedeterministické)

- V algoritmu přístupu na kanál hraje roli náhoda -náhodně volené časové prodlevy, tím vznikají kolize.
- Nemusí vždy vést k výsledku. Metody: Aloha, CSMA

- Metoda CSMA poslouchá nosnou a pokud nikdo nevysílá, může začít vysílat ona. Ani CSMA nedokáže zabránit kolizím.

Síť může fungovat několika způsoby:

Spojovaně a nespojovaně

Spojovaná komunikace:

- Nejprve se strany musí domluvit, že spolu vůbec chtějí komunikovat (mohou se domluvit i na dalších parametrech vzájemné komunikace).
- Při navázání spojení je nalezena a vyznačena trasa přenosu.
- Poté probíhá vlastní komunikace – přenáší se celé bloky dat (pakety) po trase, která byla nalezena při navazování spojení.
- Na konci je spojení třeba zase ukončit (zrušit vyznačenou trasu).

Nespojovaná komunikace:

- Komunikující mezi sebou nenavazují žádné spojení (neověřují si, že druhá strana chce komunikovat, nehledá se žádná jedna trasa mezi komunikujícími.
- Komunikace probíhá skrze zasílání samostatných zpráv.
- Na konci není třeba nic ukončovat

Spolehlivě a nespolehlivě

- **Spolehlivý přenos** – ten, kdo data přenáší, považuje za svou povinnost postarat se o nápravu v případě chyb (případně si vyžádá nový přenos).
- **Nespolehlivý přenos** – nepovažuje za svou povinnost postarat se o nápravu (poškozená data zahodí a pokračuje dál, či ani nezkoumá, zda jsou poškozená)
    - Na principu přepojování paketů či přepojování okruhů.

Přepojování paketů:

- Používají se hodně ve světě počítačů, požívají jej prakticky všechny datové sítě, méně jsou používány ve světě spojů (veřejné datové sítě).
- K jednotlivým přenosům se využívá vždy celá dostupná přenosová kapacita pro všechny odesílatele i příjemce.
- Přenášená data musí být opatřena identifikací odesílatele a příjemce.
- Nelze přenášet jednotlivé byty, smysl má pouze blokový přenos (přenos paketů, rámců, buněk…).
- Standardně jde o přenos „best effort“

Přepojování okruhů:

- Používá se hodně ve světě spojů (například veřejná telefonní síť), avšak ve světě počítačů je používána málo (sériové komunikace například).
- Týká se přidělování dostupné či disponibilní přenosové kapacity sítě.
- Z přenosové kapacity se vyjme tolik, o kolik si komunikující strany řeknou – je přidělena do výlučného použití.

(pokud ji nevyužijí, nemůže být přepuštěna někomu jinému, kdo by ji potřeboval), je jim garantována a je také uživatelům naúčtována.

Komunikující mají mezi sebou přímé spojení.

- Stylem „best effort“ či s garancí kvality služeb (Quality of Service).

„Best effort“

- Způsob přenosu, kdy je „měřeno všem datům stejně“.
- Přenos má negarantovaný charakter (síť se snaží vyhovět všem požadavkům na přenos, dokud její zdroje stačí, pokud zdroje přestávají stačit, jsou požadavky kráceny všechny stejně).

S garancí kvality služeb (QoS)

- Přenosová síť dokáže rozlišovat mezi jednotlivými přenosy a nabízet jim různou kvalitu přenosu.
- Rezervace zdrojů – potřebné zdroje jsou rezervovány jen pro jeden příslušný přenos a nemůže je využívat nikdo jiný.

Nemusí představovat žádnou garanci, ale jen určitou přednost formou prioritizace (některé přenosy mají přednost před jinými).

Blokovým či proudovým způsobem

Blokový přenos:

- Data se přenáší po blocích (každý blok se přenáší jako celek)

Existuje několik druhů označení bloků:

- **Paket** (packet) – blok dat, přenášený na úrovni síťové vrstvy (velikost je proměnná, ale shora omezená).
- **Rámec** (frame) – blok dat, přenášený na úrovni linkové vrstvy (velikost je proměnná, ale shora omezená).
- **Buňka** (cell) – malý blok fixní velikosti, obvykle přenášený na úrovni linkové vrstvy.
- **Diagram** – paket přenášený nespojovaným způsobem.
- **Zpráva** (message) – blok dat, přenášený na úrovni aplikační vrstvy.

Proudový přenos:

- Komunikující strany si předávají data jako proud bitů/bytů (po jednotlivých bitech, bytech či znacích, nemusejí být shromažďována do větších celků – bloků, nemusejí být explicitně adresována).

Adresace IPV4

- IP adresa, konfigurace síťového adapteru, routing

Charakteristika IP (Internet Protokolu)

- Je to univerzální přenosový protokol síťové vrstvy, který se snaží fungovat „nad vším“ – to znamená nad libovolnou přenosovou technologií
- Je jediným přenosovým protokolem TCP/IP na síťové vrstvě
- Používá virtuální pakety – nemají ekvivalent v Hardware, musí se zpracovávat v Softwaru, říká se jim IP diagramy
- Zajišťuje přenos diagramů skrz internet a realizuje směrování
- Je implementován v hostitelských počítačích a ve směrovačích
- Funguje nespolehlivě a nespojovaně
- Dnes se používá verze číslo 4 (IPv4) a v jejím rámci se používají 32bitové IP adresy

Adresace

- Každý uzel musí mít unikátní IP adresu, aby jej bylo možné rozlišit.
- IP adresy jsou:
  - Fyzicky „jednolité“ – každá má 32 bitů
  - Logicky dvousložkové – mají síťovou část (s adresou sítě, identifikující síť jako celek), a adresu uzlu v rámci sítě (relativní část)
    - Hranici mezi složkami tvoří bitová pozice
      - Síťovou část tvoří vyšší bity, relativní adresu uzlu tvoří zbývající nižší bity IP adresy, dnes je hranice volitelná
- Adresy nemohou být přidělovány libovolně, musí být respektováno rozdělení na sítě a uzly ve stejné síti musí mít IP adresy se stejnou síťovou částí, kdežto uzly v různých sítích musí mít IP adresy s různými síťovými částmi
- IP adresy patří rozhraním, ne uzlům; musí se přidělovat po celých blocích, se stejnou síťovou částí, bez ohledu na to, kolik se jich využije a pokud jsou přidělené jedné síti, nelze je použít v jiné
- Autoři předpokládali, že bude existovat:
  - Malý počet opravdu velkých sítí, které vyžadují malou síťovou část a naopak velkou část pro relativní adresu uzlu (třída A)
  - Střední počet středně velkých sítí, které by měli mít srovnatelně velkou síťovou i relativní část (neboli třída B)
  - Velký počet malých sítí, které vyžadují velkou síťovou část a postačí jim malá část pro relativní adresy (neboli třída C)
- Existují tři třídy adres:
  - Třída A - pro velmi velké sítě, rozděluje 32bitů na 8 a 24
  - Třída B - pro středně velké sítě, rozděluje na 16 a 16
  - Třída C - pro malé sítě, rozděluje na 24 a 8
  - Autoři se tímto rozdělením snažili zmenšit plýtvání s IP adresami
- Symbolický zápis IP adres
  - IP adresu lze chápat jako jedno velké (32bitové) binární číslo, které se však špatně zapisuje i čte
  - Používá se jednotný způsob zápisu:
    - Obsah každého bytu je vyjádřen jako desítkové číslo
    - Jednotlivé části jsou spojeny tečkou (193.84.57.34)
- Způsob přidělování IP adres:
  - Existuje zásada, že žádná IP adresa nesmí být přidělena dvakrát
  - Existuje centrální autorita, která přiděluje adresy, centrální autoritou je IANA, která přiděluje cele bloky IP adres regionálním přidělovatelům
    - RIPE (Evropa)
    - APNIC (Asie a Pacific)
    - Internic (ARIN, v USA)
  - Na úrovni IANA byl vyčerpán prostor IPv4 1. 2. 2011

Routování (neboli směrování)

- Je to volba směru pro další předání paketu či diagramu
- Zahrnuje:
  - Výpočet optimální cesty
  - Uchovávání směrových informací (což znamená vedení směrovacích tabulek)
  - Předávání paketů (neboli forwarding)
  - Udržování směrových informací – aktualizuje údaje pro vypočtení cesty a reaguje na změny

- Dále s routováním souvisí:
  - Celková koncepce směrování
  - **Statické směrování**
    - Obsah směrovacích tabulek má statický charakter a nemění se (ruční konfigurace směrovačů a jejich směrovacích tabulek, nereaguje to na změny v síti)
    - Používá se jen výjimečně:
      - Pro definování takzvaných implicitních cest
      - Pro zavedení směrů, které nejsou inzerovány
      - Pro implementaci speciálních směrovacích politik
      - Jako obrana proti nekorektním směrovacím informacím, ...
  - **Dynamické směrování**
    - Obsah směrovacích tabulek má dynamický charakter a mění se (často je základ konfigurace vytvářen staticky, ruční konfigurací směrovačů; ostatní údaje se průběžně aktualizují)
    - Existují dvě základní varianty:
      - **Vector-distance rating**
        - Sousední směrovače si předávají celé své směrovací tabulky
        - Je hůře škálovatelní a méně stabilní, přestává se používat
      - **Link-state routing:**
        - Směrovače si předávají jen údaje o průchodnosti cest k sousedům
        - Lépe škálovatelné, používá se
  - Přímé a nepřímě doručování
    - **Přímé doručování:**
      - Odesilatel a příjemce se nachází ve stejné IP síti
      - Odpadá rozhodování o volbě směru, o doručení se dokáže postarat linková vrstva
    - **Nepřímé doručování:**
      - Odesilatel a příjemce se nacházejí v různých IP sítích
      - Odesilatel musí určit nejvhodnější odchozí směr
  - Metody optimalizace směrovacích tabulek
  - Řešení směrování v opravdu velkých systémech jako jsou autonomní systémy (může si sám stanovit vlastní směrovací politiku)

Nastavení routovacího zařízení

- Vyresetujeme zařízení do továrního nastavení
- Poté zjistíme, přes jakou IP adresu se dostaneme do routovacího zařízení (routr, L3Switch,...).
- Nastavíme si na počítači IP adresu z IP sítě routru a výchozí bránu nastavíme jako IP adresu routru.
- Přes prohlížeč se připojíme do menu.
- V menu nastavíme IP adresu na WAN rozhraní a na LAN rozhraní. Na WAN rozhraní musí být IP adresa z IP sítě, ke které se připojujeme. Na LAN rozhraní si nastavíme nějakou IP adresu z rozsahu neveřejných IP adres.
- Zapneme DHCP server aby nám uděloval IP adresy automaticky v lokální síti a můžeme nastavit aj výchozí DNS server (př: 8.8.8.8 – Google, 195.113.176.229 –Školní ,....).
- Dále zapneme NAT, který nám maskuje IP adresy z lokální sítě za jednu IP adresu z veřejné sítě(za IP adresu na WAN rozhraní).
- NAT
  - Překlad síťových adres je funkce, která umožňuje překládání adres. Což znamená, že adresy z lokální sítě přeloží na jedinečnou adresu, která slouží pro vstup do jiné sítě (např. Internetu), adresu překládanou si uloží do tabulky pod náhodným portem, při odpovědi si v tabulce vyhledá port a pošle pakety na IP adresu přiřazenou k danému portu. NAT je vlastně jednoduchým proxy serverem. NAT může být softwarového typu (Nat32, Kerio Winroute firewall), nebo hardwarového typu (router s implementací nat).

- **Příklad:**
- Zapojení 3 routrů, aby každý routoval a na každý routr bude připojený jedno PC.
- Nejprve zapojíme do každého routru na LAN port PC, poté z Routru 1(R1) přes WAN do LAN R2 a z WAN R2 do LAN R3.Zmenežujeme zařízení viz. nastavení. Následně nastavíme pro každý routr s počítačem jinou IP síť :
  - 1) PC1,R1- 192.168.1.0/24 (LAN rozhraní R1)
  - 2) PC2,R2- 192.168.2.0/24 (WAN rozhraní R1 a LAN rozhraní R2)
  - 3) PC3,R3- 192.168.3.0/24 (WAN rozhraní R2 a WAN rozhraní R3)

- Nastavíme routovací tabulky pro R1 a R3
  - 1)R1- 192.168.3.0/24 Výchozí brána 192.168.2.x(co nastavíme na LAN rozhraní R2)
  - 2)R3- 192.168.1.0/24 Výchozí brána 192.168.3.x(co nastavíme na WAN rozhraní R2)

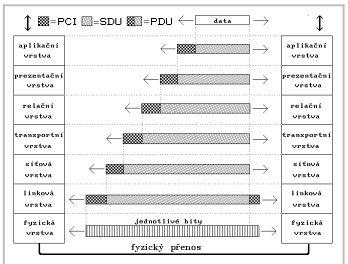
Komunikační model

Model ISO/OSI je *referenční komunikační model* označený zkratkou slovního spojení "International Standards Organization / Open Systen Interconnection" (Mezinárodní organizace pro normalizaci / propojení otevřených systémů). Jedná se o doporučený model definovaný organizací ISO v roce 1983, který rozděluje vzájemnou komunikaci mezi počítači do *sedmi souvisejících vrstev*. Zmíněné vrstvy jsou též známé pod označením *Sada vrstev protokolu*.

Úkolem každé vrstvy je poskytovat služby následující vyšší vrstvě a nezatěžovat vyšší vrstvu detaily o tom jak je služba ve skutečnosti realizována. Než se data přesunou z jedné vrstvy do druhé, rozdělí se do paketů. V každé vrstvě se pak k paketu přidávají další doplňkové informace (formátování, adresa), které jsou nezbytné pro úspěšný přenos po síti.

Vrstvový model ISO/OSI

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.



Model ISO/OSI

Podle referenčního modelu není dovoleno vynechávat vrstvy, ale některá vrstva nemusí být aktivní. Takové vrstvě se říká *nulová*, nebo *transparentní*.

Komunikaci mezi systémy tvoří:

- komunikace mezi vrstvami jednoho systému, řídí se pravidly, která se obvykle nazývají *rozhraní* (interface),
- komunikace mezi stejnými vrstvami různých systémů, řídí se *protokoly*.

1. Fyzická vrstva

Definuje prostředky pro komunikaci s přenosovým médiem a s technickými prostředky rozhraní. Dále definuje fyzické, elektrické, mechanické a funkční parametry týkající se fyzického propojení jednotlivých zařízení. Je hardwarová.

2. Linková vrstva

Zajišťuje integritu toku dat z jednoho uzlu sítě na druhý. V rámci této činnosti je prováděna synchronizace bloků dat a řízení jejich toku. Je hardwarová.

3. Síťová vrstva

Definuje protokoly pro směrování dat, jejichž prostřednictvím je zajištěn přenos informací do požadovaného cílového uzlu. V lokální síti vůbec nemusí být pokud se nepoužívá směrování. Je hardwarová ale když směrování řeší PC s dvěma síťovými kartami je softwarová.

4. Transportní vrstva

Definuje protokoly pro strukturované zprávy a zabezpečuje bezchybnost přenosu (provádí některé chybové kontroly). Řeší například rozdělení souboru na pakety a potvrzování. Je softwarová.

5. Relační vrstva

Koordinuje komunikace a udržuje relaci tak dlouho, dokud je potřebná. Dále zajišťuje zabezpečovací, přihlašovací a správní funkce. Je softwarová.

6. Prezentační vrstva

Specifikuje způsob, jakým jsou data formátována, prezentována, transformována a kódována. Řeší například háčky a čárky, CRC, kompresi a dekompresi, šifrování dat. Je softwarová.

7. Aplikační vrstva

Je to v modelu vrstva nejvyšší. Definuje způsob, jakým komunikují se sítí aplikace, například databázové systémy, elektronická pošta nebo programy pro emulaci terminálů. Používá služby nižších vrstev a díky tomu je izolována od problémů síťových technických prostředků. Je softwarová.

TCP/IP

Segment TCP

- Data posílána jedním počítačem na druhý protokolem TCP jsou rozdělena na segmenty. Ty jsou vkládány do IP datagramů. Je-li použit delší datagram než je maximální povolení délka pro přenos na fyzické vrstvě, dochází k fragmentaci IP datagramu. Fragmentace ovšem zatěžuje více, a proto je lepší posílat takové datagramy, aby nemusela být nutná. Fragmentace může být známkou útoku na data, a

proto není na některých firewallech povolena.

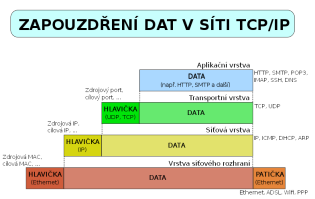


Schéma zapouzdření aplikačních dat na vrstvách TCP/IP.

IP packet (packet)

- blok dat přenášený v počítačových sítích založených na přepojování paketů, kde je možné přenášet data i při výpadcích některých spojů. Některé typy síťových spojů přenos paketů nepodporují (například point-to-point) a data se v nich přenášejí jako proud bajtů, znaků nebo bitů.

Ethernet frame (rámec)

- blok dat, který je uvozen *synchronizační sekvencí*, za kterou následují *přenášená data* a na konci je zakončen *trailerem*. Příjemce je pomocí synchronizační sekvence schopen rámec přijmout a dokud ji nezachytí, přijímaná data ignoruje (například pokud začne přijímat až uprostřed přenášeného rámce). Přenášená data se typicky dělí na *hlavičku*, která obsahuje služební údaje nutné pro přenos rámce a *tělo*, které obsahuje přenášená data. Součástí přenášených dat bývá též *kontrolní součet*, který umožňuje poškozený rámec zahodit (dále nezpracovávat).

1. vrstva - síťová

IP - Internet Protocol

- nejzákladnější protokol, neobsahuje potvrzování (počítač neví jestli data, které vyslal, přijmul vzdálený počítač). Zabezpečuje správné doručování dat k jednotlivým počítačům v síti.

ARP - Address Resolution Protocol

- Převádí 32 bitovou IP adresu na 48 bitovou MAC adresu.

RARP - Reverse Address Resolution Protocol

- Naopak převádí MAC adresu na IP adresu. Tento protokol používají bezdiskové pracovní stanice, které neznají svojí IP adresu.

ICMP - Internet Control Message Protocol

- Používá se k signalizaci chyb a různých nestandardních situací (ale pouze potřebám signalizace, ICMP sám nezajišťuje jejich nápravu).

IGMP - Internet Group Management Protocol

- Podporující tzv. skupinové vysílání (multicasting).

2. vrstva - transportní

TCP/UDP

- Musíme zavést další rozdělení - *port*. Na jednom počítači lze provozovat několik programů, které poskytují své služby. Aby se rozlišilo na kterou službu program přistupuje musí být nějak rozlišeny. A to takzvaným portem. Například služba www serveru HTTP má standartně port 80 atd. Maximálně může být najednou spuštěno 65 tisíc portů (programů). SOCKET = IP adesa + port.

TCP - Transmission Control Protocol

- Je potvrzovaný. TCP vytváří takzvané virtuální spojení. Toto spojení trvá po dobu než aplikace spojení ukončí.

UDP - User Datagram Protocol

- Nepotvrzovaný protokol. Od IP se liší jen tím, že má navíc port. Mužu tak poslat konkrétnímu programu dotaz. Moc se nepoužívá, spíše jen na služební komunikaci. Např. routery když každých 30 sec. hlásí kdo je připojen.

3. vrstva - aplikační

Obsahuje protokoly (aplikace), které se už přímo využívají ke komunikaci po síti.

FTP/TFTP - File Transfer Protocol/Trivial FTP

- Slouží k přenosu souborů mezi počítači spojenými do sítě. TFTP je jednoduší varianta k FTP.

HTTP/HTTPS - Hyper Text Transfer Protocol

- Slouží k přístupu na www stránky. HTTPS je zabezpečený (šifrovaný) přenos www stránek.

TELNET - Telecommunication Network

- Vytváří terminálový provoz. Můžeme pracovat se vzdáleným počítačem stejně jako bychom seděli u terminálu bezprostředně k němu připojeném. Protože komunikace probíhá nešifrovaně představuje jeho používání bezpečnostní riziko. Náhradou za TELNET je SSH (Secure Shell) který komunikuje šifrovaně.

POP3 - Post Office Protocol

- Slouží k přijímání elektronické pošty poštovním klientem.

SMTP - Simple Mail Transfer Protocol

- Slouží k odesílání elektronické pošty poštovním klientem

Rozdíl mezi ISO/OSI a TCP/IP

V celkovém přístupu autorů

- *ISO/OSI:*
  - všechno musíme vymyslet sami (nebo alespoň převzít to, co vymysleli jiní, a udělat z toho vlastní standard)
  - příklad: ISO vydává Ethernet jako svůj standard ISO 8802.3
- *TCP/IP:*
  - to co je rozumné převezmeme a využijeme
  - soustředí se na "provázání" vlastních řešení s cizími - řeší např. jak provozovat IP nad Ethernetem

Ve způsobu tvorby nových řešení:

- *ISO/OSI:*
  - od složitého k jednoduššímu
  - řešení vznikají od začátku jako "dokonalá"
  - nejprve navymýšlí vzdušné zámky, pak musí slevovat
  - nejprve vznikne standard, pak se zkoumá praktická realizovatelnost
- *TCP/IP:*
  - od jednoduššího ke složitějšímu
  - řešení vznikají nejprve jako "skromná", postupně se obohacují
  - nejprve se řešení ověří, a teprve pak vzniká standard

V pohledu na počet vrstev a způsob jejich fungování

- jaké služby mají být nabízeny a na jaké úrovni mají být poskytovány (kde má být zajišťována spolehlivost)
- jak mají služby fungovat
  - spolehlivost/nespolehlivost,
  - spojovanost/nespojovanost,
  - princip maximální snahy vs. garance kvality služeb, ...
- zda má být ponechána možnost volby
  - mají aplikace právo si vybrat např. mezi spolehlivým a nespolehlivým přenosem?

Externí odkazy

- Několik prezentací na toto téma, podle kterého učil Klečka POSka !!!!
- Wikipedia - ISO/OSI
- Wikipedia - TCP/IP

Metalické technologie počítačových sítí

Pasivní a aktivní prvky

- Počítačové sítě se skládají ze dvou základních částí. Pasivní část, zahrnující různá vedení, která propojují jednotlivá zařízení sítě (např. metalické a optické vedení).
- Proto, aby se mohli uživatelé dostat k informacím na lokální server či do Internetu, jsou potřeba aktivní prvky. Ty zahrnují síťové přepínače (switche), routery, access pointy, firewally, různá bezpečnostní zařízení zajišťující vyšší prioritu pro vyřízení některých požadavků.

Pasivní prvky

- Mezi pasivní prvky se řadí především datové rozvaděče, které fyzicky přenášejí data do počítače.
- Pasivní spojení počítačových sítí se skládá z mnoha částí, které musíme všechny spojit do jednoho celku. Jedná se o:
- Kabel UTP (křížený dvoupár) kategorie 5, 5e, 6 a 7. Nižší typy kategorii kabelů se v dnešní době již nepoužívají.
- Zásuvky UTP, zakončovací konektory příslušné stejné kategorie a doporučuje je i výrobce.
- 19“ rozvaděče, plechové skříně, které se stávají prostorem pro centra zakončení a rozvodu.
- Patch panely, zakončovací panely do 19“ rozvaděče.
- UTP propojovací kabely, které se používají na propojení v rozvaděči a pro spojení mezi zásuvkou a zařízením.

- **Kroucená dvojlinka** - druh kabelu, který je tvořen páry vodičů, které jsou po své délce pravidelným způsobem zkrouceny a následně jsou do sebe zakrouceny i samy výsledné páry.
- **Koaxiální kabel** - je asymetrický elektrický kabel s jedním válcovým vnějším vodičem a jedním drátovým nebo trubkovým vodičem vnitřním.
- **ITU-T G.hn** - technologie využívá existující domácí sítě (síťová kabeláž, koaxiální kabeláž, ADSL apod.) a podporuje provoz sítě přes elektrické přípojky, telefonní linky a koaxiální kabely s datovým tokem až do 1 Gbit/s v lokální síti.

Aktivní prvky

- S pasivní částí počítačových sítí samozřejmě přímo spolupracují aktivní části.
- Aktivní síťové prvky jsou všechny zařízení, které slouží ke vzájemnému propojení v počítačových sítích. Aktivní síťový prvek je všechno to, co nějakým způsobem aktivně působí na přenášené signály - tedy je zesiluje a různě modifikuje.

Mezi aktivní prvky se řadí především: opakovač, hub, switch, bridge nebo router. Patří zde však i další zařízení jako například síťová karta, tiskový server nebo host adapter.

Přenos FullDuplex a HalfDuplex

Full Duplex (plný duplex)

- U plného duplexu může obousměrná komunikace probíhat současně. Příkladem takové komunikace může být běžný telefonický hovor, kdy obě zúčastněné strany mohou hovořit zároveň.
- Plný duplex v Ethernetu funguje tak, že dva páry vodičů v ethernetovém kabelu jsou využívány pro odesílání rámců a dva páry jsou využívány pro příjem.

Half duplex(poloduplex)

- Poloduplex nebo anglicky half-duplex je režim střídavé obousměrné komunikace (např. v počítačové síti nebo rádiotelefonní síti).
- V daném okamžiku může probíhat pouze v jednom směru, směr přenosu se ale může měnit.
- Příkladem takové komunikace je vysílání radiostanic (vysílaček) přes opakovač; typické pro half-duplexní spojení je používání signalizace „přepínám“.

Kabeláž a zásuvky

- **Strukturovaná kabeláž** je obecné označení metalických prvků, které umožňují propojení jednotlivých uživatelů v rámci počítačové sítě.

- Je to univerzální systém, který:
- podporuje přenos digitálních i analogových signálů,
- u něhož se přípojné body instalují i tam, kde momentálně nejsou potřeba,

- který používá datové kabely se čtyřmi kroucenými páry,
- u kterého se předpokládá dlouhá technická i morální životnost,
- jehož správná funkčnost je pro firmu stejně tak důležitá jako fungování elektrických rozvodů a dalších prvků firemní infrastruktury.

Telekomunikační zásuvky:

- Slouží pro připojení koncových uživatelských zařízení - např. stolní počítač, notebook, analogový nebo ISDN telefon, VoIP telefon či síťová tiskárna. Nejčastěji se tyto zásuvky dodávají v dvouportovém provedení - tj. u jednoho uživatele slouží jeden port pro připojení počítače nebo notebooku, druhý port pak pro připojení telefonu. 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:

- Jedná se o měděné kabely obsahující čtyři kroucené páry, které vzájemně propojují již zmíněné telekomunikační zásuvky a patch panely.

Patch kabely:

- Jedná se o propojovací kabely, jejichž užití bylo již naznačeno výše; umožňují totiž připojení uživatelských zařízení do počítačové sítě na straně telekomunikačních zásuvek a připojení jednotlivých portů patch panelů do aktivních zařízení na straně rozvaděče.

U všech výše zmíněných komponent je již od roku 1991, kdy vznikl první standard pro strukturovanou kabeláž, přesně definován způsob jejich použití, jsou dány jejich elektrické vlastnosti a je přesně specifikováno fyzické rozhraní, které umožňuje jejich vzájemné propojení do jednoho celku

Kategorie prvků (podle výkonnosti):

- Kategorie 3 (Cat. 3) - je nejnižší kategorií. U prvních sítí se komponenty kategorie 3 používaly pro přenos hlasu i dat. Dnes se již prvky kategorie 3 ve většině případů používají pouze pro telefonní rozvody (např. propojovací ISDN panely, kabely k telefonní ústředně či propojovací šňůry k telefonnímu přístroji). Maximální přenosová rychlost, které bylo možné dosahovat na kabelážích kategorie 3, byla 10 Mb/s (protokol 10Base-T).

- Kategorie 4 (Cat. 4) - tato kategorie se již téměř nepoužívá. Byla spojována především se společností IBM a jejími prvky pro sítě Token Ring. Kategorie 4 byla silně zastoupená především v USA, v evropských standardech nebyla nikdy zmíněna.

- Kategorie 5 (Cat. 5) - tato kategorie byla schválena v roce 1995. Nyní je již nahrazena kategorií 5E – tzn. stejně jako v případě kategorie 3 a 4, se jedná již o historickou kategorii. Maximální přenosová rychlost, které bylo možné dosahovat na komponentech kategorie 5 byla 100 Mb/s (tzv. Fast Ethernet, protokol 100Base-T).

- Kategorie 5E (Cat. 5E) – vychází z kategorie 5 a má i stejnou šířku pásma (tj. 100 MHz). Z důvodu cenové dostupnosti je v této chvíli kategorie 5E stále nejrozšířenější kategorií ve strukturované kabeláži. Komponenty kategorie 5E umí přenést i Gigabit Ethernet v podání protokolu 1000BaseT. Nicméně přenosová rychlost 1 Gb/s je limitní rychlostí pro všechny komponenty kategorie 5E.

- Kategorie 6 (Cat. 6) - tato kategorie byla schválena v roce 2002. Pracuje s dvojnásobnou šířkou pásma než kategorie 5E (tj. až 250 MHz). Vyšší kvalita komponent s větší šířkou pásma zajišťuje vynikající spolehlivost přenosu Gigabit Ethernetu (1 Gb/s) u kabelážních systémů kategorie 6 ve srovnání s kategorií 5E a zároveň i podporu dalších protokolů (např. kromě již zmíněného 1000Base-T i 1000Base-TX nebo částečně i nového protokolu 10GBase-T).

- Kategorie 6A (Cat. 6A) – toto je nejnovější kategorie, která vznikla v dubnu 2008. V této chvíli je plně specifikována pouze v americké normě ANSI/TIA/EIA 568B.2-10. S kategorií 6A se počítá především pro plnohodnotný přenos protokolu 10GBase-T na všechny vzdálenosti (rychlost 10 Gb/s), které jsou v metalické kabeláži běžné. Oproti kategorii 6 pracují komponenty kategorie 6A s dvojnásobnou šířkou pásma – tj. 500 MHz, která poskytuje komponentům této nové kategorie již zmíněnou vyšší datovou propustnost. Kompletní schválení těchto nových prvků i v ostatních standardech (tj. ISO/IEC a CENELEC) se očekává ve druhé polovině roku 2009. I když se zpočátku počítalo s nasazením kategorie 6A především v páteřních spojích nebo datových centrech, mnozí výrobci (např. Solarix, Signamax či RiT) nabízí svá 10G řešení i pro kabeláže běžných LAN sítí – tj. až k uživateli na stůl.

- Kategorie 7 (Cat. 7) - tato kategorie byla poprvé zmíněna již v roce 1997, nicméně schválení se dočkala až v roce 2002, a to navíc pouze pro

kabel a nikoli pro spojovací hardware (tj. zásuvky, patch panely atd.). Pracovní frekvence kategorie 7 je nyní 600 MHz.

- Kategorie 7A (Cat. 7A) - současná kategorie 7 z důvodu malého odstupu šířky pásma od komponentů kategorie 6A (500 MHz vs. 600 MHz) bude postupně nahrazena touto novou kategorií s dvojnásobnou šířkou pásma - tj. 1000 MHz..

Definice generické kabeláže pro čtyři třídy vedení :

- Vedení třídy A: specifikováno do 100 KHz
- Vedení třídy B: specifikováno do 1 MHz
- Vedení třídy C: specifikováno do 16 MHz
- Vedení třídy D: specifikováno do 100 MHz

Vztah mezi kategoriemi a třídou v závislosti na délce kanálu

- Typ kabelu Třída A Třída B Třída C Třída D Třída D+ Třída E
- CAT 3 2 Km 200 m 100 m - - -
- CAT 4 3 Km 260 m 150 m - - -
- CAT 5 3 Km 260 m 160 m 100 m - -
- CAT 5E 3 Km 260 m 160 m 100 m 100 m -
- CAT 6 3 Km 260 m 160 m 100 m 100 m 100 m

Barvy kabelů dle typu ethernetu: (zleva doprava)

Fast ethernet: oranžovobílá, oranžová, zelenobílá, modrá, modrobílá, zelená, hnědobílá, hnědá,

Ethernet: oranžovobílá, oranžová, zelenobílá, zelená,

PC do PC: (křížený) zelenobílá, zelená, oranžovobílá, modrá, modrobílá, oranžová, hnědobílá, hnědá,

PoE - power over ethernet - napětí přes ethernet:

- Hlavním účelem technologie PoE je ušetřit kabely při napájení zařízení. Napětí je jednoduše vedeno ethernetovým kabelem a zjednodušit připojování přístrojů; zapojuje se jen 1 datový konektor místo 2 (data+napájení)
- Může být využíváno jako záložní zdroj energie při výpadku napájecí sítě v okolí přístroje, centrální zdroj PoE je obvykle napájen zálohovaně
- Umožňuje správci sítě snadný dálkový restart napájeného přístroje na konci kabelu vypnutím a zapnutím napájení pomocí příkazu na portu (síťový *LAN přepínač s napájecími porty).
- Funguje jen na ethernetu!

Způsoby PoE

- Pro PoE se využívá v zásadě dvou možných řešení:
- Napájení po volných nevyužitých párech v datovém kabelu (režim B). Napájecí páry jsou 4,5 a 7,8.
- Napájení „fantómovým“ napětím mezi dvojicí aktivních párů vodičů, po kterých se současně přenášejí i data (režim A). Napájecí (a datové) páry jsou zde 1,2 a 3,6.

- Zapojení párů v konektoru RJ-45.
- Na vysvětlení lze dodat, že osm vodičů v kabelu je rozděleno do 4 párů, které jsou samostatně krouceny. Vodičům jsou dle normy přiřazena

čísla 1-8 a do párů jsou rozděleny takto:

- 1,2
- 3,6
- 4,5
- 7,8

Čísla současně udávají pořadí kontaktů na konektoru RJ45.

Switching

rozdělení:

- Adaptive switching

- Cut-through switching

Adaptive switching:

- Adaptive switching je v informatice metoda přeposílání datových rámců v počítačových sítích. Pracuje-li switch v adaptivním režimu, nakládá s daty podle metody cut through. Zvýší-li se však množství chyb v přenosu na některém portu switche, dojde ke změně nastavení a dále se používá metoda store and forward.

Z toho vyplývá, že tato metoda dokáže efektivně optimalizovat výkon switche v závislosti na podmínkách, ve kterých switch pracuje. Jsou-li komunikace bezchybná, zaručí metoda cut through jejich doručení příjemci s nejmenší možnou latencí. Avšak v případě, že dochází při přenosu k chybám, postará se metoda store and forward o to, aby síť zbytečně nezahlcovaly vadné rámce.

Cut-through switching:

- Cut-through switching je v informatice metoda přeposílání datových rámců v počítačových sítích, pomocí které je dosahováno nejmenší latence (zpoždění). Switch začne s odesíláním přijímaného ethernetového rámce ještě před tím, než ho přijme celý. Dochází tak k výraznému snížení latence způsobené průchodem rámce switchem.

jak Cut-through switching pracuje:

- Metoda cut-through začne s odesíláním ve chvíli, kdy je známa MAC adresa příjemce. Vzhledem k tomu, že adresa příjemce je v ethernetovém rámci hned na začátku, je zpoždění způsobené průchodem rámce skrze switch pouze 7+1+6 oktetů (preambule, SFD, MAC adresa příjemce). Cut-through znatelně snižuje latenci síťového provozu mezi odesílatelem a příjemcem, avšak doručeny jsou i poškozené rámce. Data jsou switchem při metodě cut-through přijímána a odesílána jako kontinuální proud dat. Proto může být tato metoda použita jen tam, kde je rychlost výstupního rozhraní menší nebo rovna rychlosti vstupního rozhraní, protože by po chvíli nebyla k dispozici data pro odeslání. Problém latence se snižuje se zvyšující se velikostí přenosových rychlostí.

Bezdrátové a optické technologie počítačových sítí

- pasivní a aktivní prvky
- bridge
- volné a licencované pásmo
- standard IEEE 802.11x

Optické technologie počítačových sítí

Optika je dražší, ale rychlá, spolehlivá a trvanlivá komunikační infrastruktura. Základním rozdílem mezi metalickými a optickými kabely je, že u metalických kabelů jsou data přenášena za využití elektrických signálů, zatímco v optických kabelech je signál přenášen světelnými impulzy.

Pasivní prvky optiky

Optické kabely:

- Skleněné nebo plastové vlákno, které prostřednictvím světla přenáší signály
- Může dosahovat rychlosti přenosu až 111 Gb/s (typické rychlosti 10 nebo 40 Gb/s)
- Imunní vůči elektrickému rušení
- Obsahují minimálně 2 optická vlákna (pro každý směr jedno), která jsou obalená sekundární ochranou a plastovým obalem.
- Rozdělujeme je na mnohovidivé optické kabely a jednovidové optické kabely.
  - Mnohavidové optické kabely - původní světelný paprsek je rozložen do více světelných paprsků - dochází k odrazu a lomu od pláště vlákna a následnému zkreslení dat.
    - Rychlost přenosu u vícevidových linek se pohybuje okolo 10 Mbit/s až 10 Gbit/s na vzdálenosti do 600 metrů
  - Jednovidové optické kabely - původní světelný paprsek prochází jedním optickým vláknem bez lomů a ohybů. Je tedy rychlejší.
    - minimální zkreslení, lze použít na desítky kilometrů bez opakovače
    - druhý nejpoužívaější typ optický vláken
    - dražší oproti ostatním
- Výhody oproti metalickému vedení:
  - velká šířka pásma
  - nízký útlum (delší opakovací úseky, menší počet zesilovačů na optické trase)
  - odolnost proti elektromagnetické interferenci a přeslechům
  - bezpečnost přenosu (signál nelze jednoduše vyvázat)
  - elektrická izolace
- Nevýhody:
  - Každý ohyb optických vláken působí nepříznivě na šíření světla.
  - V případě špatného napojení vlákna na konektor dochází ke značným ztrátám.
  - Vyšší náklady na instalaci (drahé nářadí a přístroje).

Aktivní prvky optiky:

- Media konvertory: Media konvertory patří mezi aktivní prvky, které mění typ signálu - tzv. „převodníky médií“, kde je signál převeden na jiný typ signálu, aniž by se datově změnil. Ve většině případů se tedy jedná o převodníky kde vstupem/výstupem je optické vlákno ať již singlemode, multimode a výstupem/vstupem je 1000/100/10 Mbps ethernet se standardním konektorem RJ45 připojitelným na UTP/STP metalickou kabeláž.

Výhody optiky:

Oproti metalickým kabelům mají optické sítě následující výhody:

- Velká šířka pásma - optické nosné vlny odpovídají frekvencím 1013-1016 Hz z čehož plyne obrovský potenciál přenášených rychlostí - přenosové pásmo je možné v některých případech zvětšovat na již položeném kabelu dodatečně (nasazením nových technologii),
- Nízký útlum - přenos na velké vzdálenosti bez nutnosti aktivních "opakovačů",
- Odolnost proti elektromagnetické interferenci - u optických vláken neexistují přeslechy a díky použité technologii lze použít v silně zarušeném elektromagnetickém prostředí,

- Bezpečnost přenosu - přenášené světlo nevyzařuje do okolí, těžko se dá vyvázat a v případě vyvázání dojde k poklesu signálu na koncovém zařízení a to tak může vyvázání detekovat,
- Dostupnost výroby vláken - vlákna se vyrábějí z křemíku, který není strategickou surovinou, jelikož je ho všude dostatek,
- Přenos na velké vzdálenosti - vzhledem k nízkému útlumu je možný dosah desítky km bez zapojení aktivních prvků, tak jako je tomu u metalických sítí. S rozvojem nových optických technologií se dále vzdálenosti mohou zvyšovat,
- Menší průměr a nižší hmotnost kabelů.

Bezdrátové technologie počítačových sítí

Bezdrátová síť je typ počítačové sítě, ve které je spojení mezi jednotlivými účastníky sítě uskutečňováno pomocí bezdrátové komunikace, nejčastěji elektromagnetických vln. Tato implementace se nachází na fyzickém vrstvě síťové struktury. Bezdrátová síť se používá v domácnostech, telekomunikačních sítích a ve společnostech, kde by zavádění kabelů do budovy a spojování jednotlivých místností bylo příliš drahé, či je instalace kabelů z historického hlediska nemožná (zámky, hrady a jiné památky)

Wireless PAN

Bezdrátové osobní sítě (WPAN) spojují jednotlivá zařízení v relativně malé oblasti. Která je obecně pro osobu připojenou do této sítě snadno dosažitelná. Například pomocí bluetooth nebo infračerveného světla můžeme připojit sluchátka k laptopu a tím si vytvořit malou osobní bezdrátovou síť (WPAN). ZigBee také podporuje WPAN aplikace. Osobní Wi-Fi sítě se stali samozřejmostí (2010), jako vybavení integrované do celé škály elektronických zařízení pro běžné spotřebitele. Nástroje „My WiFi“ od Intelu a „Virtual Wi-Fi“ z Windows 7 umožňují osobní bezdrátové sítě snadno a jednoduše nastavit a konfigurovat.

Wireless LAN

Místní bezdrátová síť (WLAN) spojuje dvě a více zařízení na střední vzdálenosti pomocí bezdrátové distribuční metody, obvykle poskytuje přes přístupový bod připojení k internetu. Použití rozprostření signálu nebo OFDM technologie umožňuje uživatelům pohybovat se v rámci signálem pokryté oblasti a stále být připojen do sítě.

Produkty používající WLAN standard IEEE 802.11 jsou zaregistrované pod obchodní značkou Wi-Fi. Stálé bezdrátové technologie implementují point-to-point spojení mezi počítači nebo sítěmi, které jsou umístěny na dvou vzdálených lokacích. Obvykle se používá směrový mikrovlnný nebo modulovaný laserový paprsek mezi dvěma místy, které na sebe mají volný výhled. To se obvykle používá ve městech pro propojení dvou a více budov bez instalace drátového spojení.

Wireless WWAN

Velká bezdrátová síť (WWAN) je bezdrátová síť, která typicky pokrývá velké oblasti, jako například mezi sousedícími vesnicemi a městy, nebo městem a předměstím. Tyto sítě mohou být použity k připojení poboček kanceláří nebo jako veřejný přístupový systém. Bezdrátové spojení mezi přístupovými body je obvykle point-to-point mikrovlnná linka používající parabolický reflektor na frekvenci 2,4 GHz, na menší vzdálenosti se používá omnidirektionální anténa. Typický systém obsahuje vstupní brány základních stanic, přístupové body a bezdrátové přemostění signálu. Ostatní konfigurace jsou síťové systémy, kde každý přístupový bod předává signál dál. Pokud zkombinujeme tyto systémy s obnovitelnými energetickými zdroji jako jsou solární energie nebo větrná energie mohou fungovat jako stand alone systémy.

Wireless MAN

Bezdrátové metropolitní sítě (WMAN) jsou typ bezdrátové sítě, které spojuje několik bezdrátových lokálních sítí. WiMAX je typ bezdrátové MAN a je popsána standardem IEEE 802.16.

Mobilní síť

S vývojem smartphounů a telefonních sítí běžně přenášíme data do a z mobilních zařízení:

- Globální systém pro mobilní komunikaci (GSM): GSM síť je rozdělena mezi tři hlavní systémy: přepínací systém, systém základní stanice a operační a podpůrný systém. Mobilní telefon se připojí do základního systému, který se poté připojí do operačního a podpůrného systému. Ten poté propojí mobil s přepínací stanicí, kde je hovor přesměrován tam, kam je potřeba. GSM je nejvíce používaný standard a je to majoritní standard pro mobilní telefony.
- Personal Communications Service (PCS): PCS je rádiová frekvence, kterou mohou používat mobilní telefony v Severní Americe a Jižní Asii. Sprint je první společnost, která zprovoznila PCS síť.
- D-AMPS: Digital Advanced Mobile Phone Service, vylepšená verze AMPS. Byla nahrazena novější GSM sítí.

Aktivní prvky:

- Bridge
  - Odděluje provoz dvou segmentů sítě, do své paměti RAM si sám sestaví tabulku MAC adres a portů, za kterými se dané adresy nacházejí.
  - Pracuje na linkové vrstvě
  - Výhody:
    - není ho potřeba konfigurovat
    - snižuje velikost kolizní domény
    - transparentní k protokolům z vyšších vrstev
    - levnější než router
  - Nevýhody:
    - neomezuje rozsah všesměrového vysílání
    - vyšší latence, než opakovače z důvodu čtení MAC adresy
    - dražší než opakovače
    - přemosťováním různých MAC protokolů dochází k chybám

- Antény
  - Zařízení k příjmu nebo k vysílání rádiových signálů
  - Vlastnosti antén
    - směrovost antény – jedná se o schopnost antény vyzařovat/přijímat elektromagnetické vlny v požadovaném směru
    - vyzařovací úhel antény
    - šířka přenášeného pásma – udává šířku přenášeného frekvenčního pásma
  - Druhy
    - Směrové
    - Sektorové
    - Panelové
    - OMNI (všesměrové)

Přenos v bezdrátových sítích

Přenos je uskutečněn pomocí modulování informací do elektromagnetických vln o určité frekvenci, které jsou následně šířeny etherem.

- Přenášená informace se moduluje do elektromagnetických vln
- Český telekomunikační úřad vymezil tyto pásma:
  - Volné pásmo - nevede se evidence jednotlivých spojů, ČTÚ vymezil tyto pásma:
    - 2,4 GHZ
    - 5 GHZ
    - 10GHZ
  - Licencované - je třeba nejdříve od regulačního orgánu (ČTÚ) zajistit přidělení kmitočtových "kanálů" v požadované lokalitě instalace a povolení k provozu.
    - 3,5 GHZ
    - 11GHZ
- Point to Point - PtP - Jeden prvek komunikuje s druhým a naopak. Např. přenos signálu z domu na dům.
- Point to Multi Point - PtMP - Jeden aktivní prvek komunikuje s více prvky najednou. Např. standardní domácí wi-fi.

Standard IEEE 802.11

- Je Wi-Fi standard pro lokální bezdrátové sítě
- Standardy 802.11b a 802.11g používají 2,4 GHz pásmo. Proto se mohou zařízení křížit s mikrovlnnými troubami, bezdrátovými telefony, s Bluetooth nebo s dalšími zařízeními používajícími stejné pásmo.
- Standard 802.11a používá 5 GHz pásmo a není tedy ovlivněn zařízeními pracujícími v pásmu 2,4 GHz.
- IEEE 802.11a 1999 (Max. rychlost) 5GHZ 54Mbit/s
- IEEE 802.11n 2009 2,4 nebo 5GHZ 600Mbit/s
- IEEE 802.11ac 2013 5GHZ 1000Mbit/s

Wi-Fi

- Je v informatice označení pro několik standardů IEEE 802.11 popisujících bezdrátovou komunikaci v počítačových sítích.
- Tato technologie využívá bezlicenčního frekvenčního pásma, proto je ideální pro budování levné, ale výkonné sítě bez nutnosti pokládky kabelů.

Struktura bezdrátové sítě

- Bezdrátová síť může být vybudována různými způsoby v závislosti na požadované funkci.
- Ve všech případech hraje klíčovou roli identifikátor SSID (Service Set Identifier), což je řetězec až 32 ASCII znaků, kterými se jednotlivé sítě rozlišují.

Ad-hoc sítě

- V ad-hoc síti se navzájem spojují dva klienti, kteří jsou v rovnocenné pozici (peer-to-peer). Vzájemná identifikace probíhá pomocí SSID. Obě strany musí být v přímém rádiovém dosahu, což je typické pro malou síť nebo příležitostné spojení, kdy jsou počítače ve vzdálenosti několika metrů.
- První spuštěný klient tvoří jakýsi imaginární access point, který pak řídí další komunikaci ostatních klientů.

Infrastrukturní sítě

- Cenově náročný.
- Typická infrastrukturní bezdrátová síť obsahuje jeden nebo více přístupových bodů (AP – Access Point), které vysílají své SSID. Klient si podle názvů sítí vybere, ke které se připojí. Několik přístupových bodů může mít stejný SSID identifikátor a je plně záležitostí klienta, ke kterému se připojí. Může se například přepojovat v závislosti na síle signálu a umožňovat tak klientovi volný pohyb ve větší síti.

1. Fresnelova zóna

- Prostor (elipsoid) ležící na přímce mezi vysílačem a přijímačem
- Stínění ve Fresnelově zóně nemá za následek podstatné snížení síly signálu, pokud ale signálu v ceste stojí libovolný předmět, generují se rušivé odrazy. To ma za následek snížení kvality přenosu dat a v konečném důsledku i ztrátu rychlosti a zvýšení odezvy v bezdrátové síti.

Principy datových komunikací

- duplex / poloduplex
- synchronní / asynchronní
- multiplexy (časový, statistický, ...)

Duplexní spojení

Duplexní spojení je taková komunikace (popř. přenos dat) mezi dvěma subjekty, při které mohou data putovat oběma směry současně. Jiným způsobem provozu je simplexní spojení, při kterém v daném časovém okamžiku putují data jen jedním směrem.

- **Half-duplex (poloviční duplex)**
  - Obě strany mohou přijímat i vysílat, avšak nikoli současně – v každý jednotlivý okamžik probíhá přenos pouze jedním směrem (podobné simplexu). Při této komunikaci jsou však na rozdíl od simplexu využívány dvě frekvence. Na jedné frekvenci se vysílá, na druhé přijímá.
  - Příkladem takové komunikace je vysílání radiostanic (vysílaček) přes opakovač; typické pro half-duplexní spojení je používání signalizace „přepínám“.

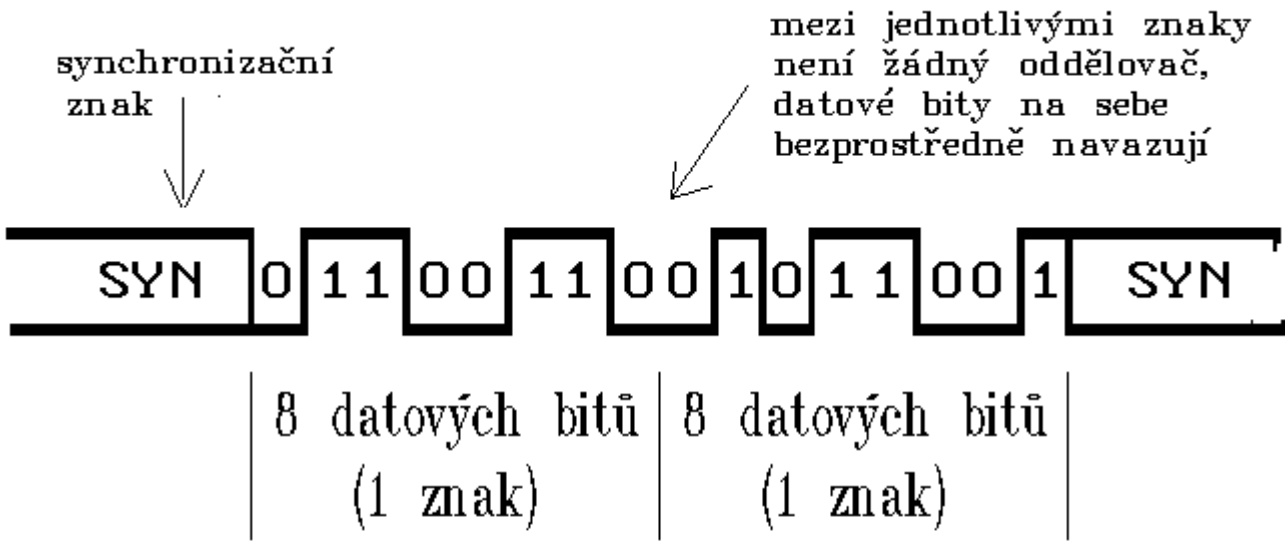
- **Full-duplex (plný duplex)**
  - U plného duplexu může obousměrná komunikace probíhat současně. Příkladem takové komunikace může být běžný telefonický hovor, kdy obě zúčastněné strany mohou hovořit zároveň.
  - Plný duplex v Ethernetu funguje tak, že dva páry vodičů v ethernetovém kabelu jsou využívány pro odesílání rámců a dva páry jsou využívány pro příjem.

Synchronní a asynchronní komunikace

Způsoby komunikace můžeme rozdělit na synchronní a asynchronní. Rozdíly nalezneme v čase kdy komunikace probíhá. Tedy jestli komunikace probíhá v reálném čase či zda v ní účastníci na sebe reagují s určitým zpožděním.

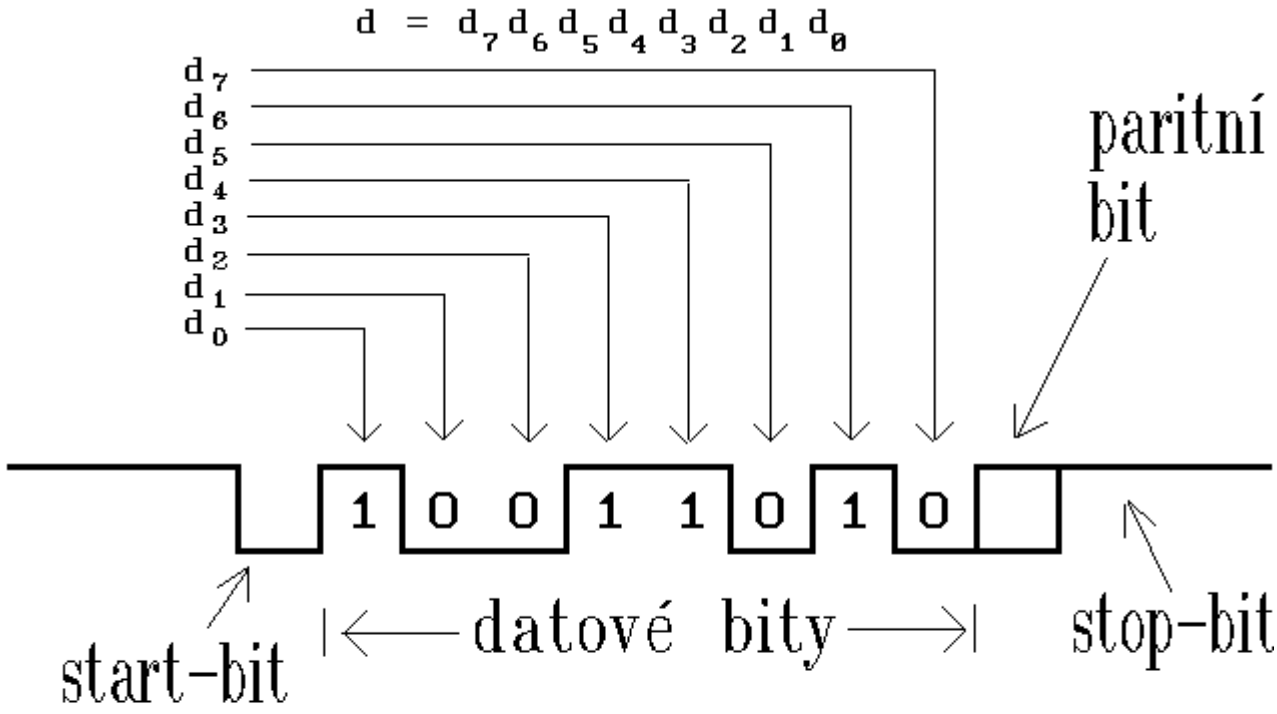
- Způsoby komunikace můžeme rozdělit na synchronní a asynchronní. Rozdíly nalezneme v čase kdy komunikace probíhá. Tedy jestli komunikace probíhá v reálném čase či zda v ní účastníci na sebe reagují s určitým zpožděním.

- **Synchronní komunikace**
  - Při synchronním přenosu jsou obvykle přenášeny celé bloky znaků. Datové bity jednotlivých znaků přitom následují těsně po sobě, bez jakýchkoli časových odstupů, a nejsou prokládány žádnými start- či stop-bity
  - Začátek bloku je indikován jedním nebo několika speciálními synchronizačními znaky (tzv. znaky SYN)
  - Synchronní přenos je obecně rychlejší než asynchronní, neboť není zatížen režií připadající na start- a stop-bity. Jeho technická a programová realizace však bývá poněkud složitější než u přenosu asynchronního.
  - Mezi nástroje synchronní komunikace patří například **chat a videokonference**



Obr. 2.2.: Synchronní přenos znaků

- **Asynchronní komunikace**
  - Při asynchronním sériovém přenosu mohou být jednotlivé znaky (přesněji značky) přenášeny s libovolnými časovými odstupy mezi sebou.
  - Příjemce pak ovšem nemůže předem vědět, kdy začíná další znak, a proto musí být schopen jeho příchod podle vhodného příznaku rozpoznat. Tímto příznakem je tzv. start-bit (též rozběhový prvek, viz obrázek 2.1), kterým začíná každý asynchronně přenášený znak. Příchod start-bitu je pro příjemce současně i možností správně si nastavit své měřítko času (přesněji svou časovou základnu). To je nutné proto, aby příjemce správně určil časové okamžiky, kdy má vyhodnocovat stav jednotlivých datových bitů, které po start-bitu následují.
  - Nevyžaduje zapojení účastníků ve stejném čase
  - Mezi představitele asynchronní komunikace se řadí především **fóra,virtuální výukové prostředí (Moodle)**

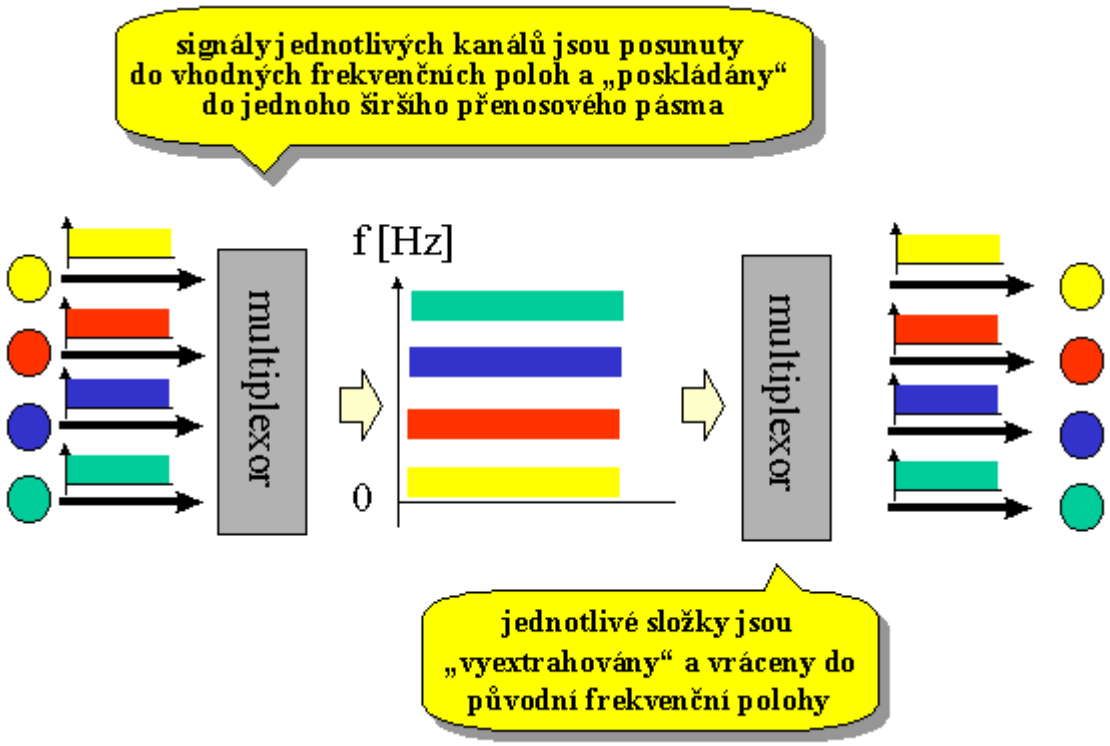


Obr. 2.1.: Asynchronní přenos znaku

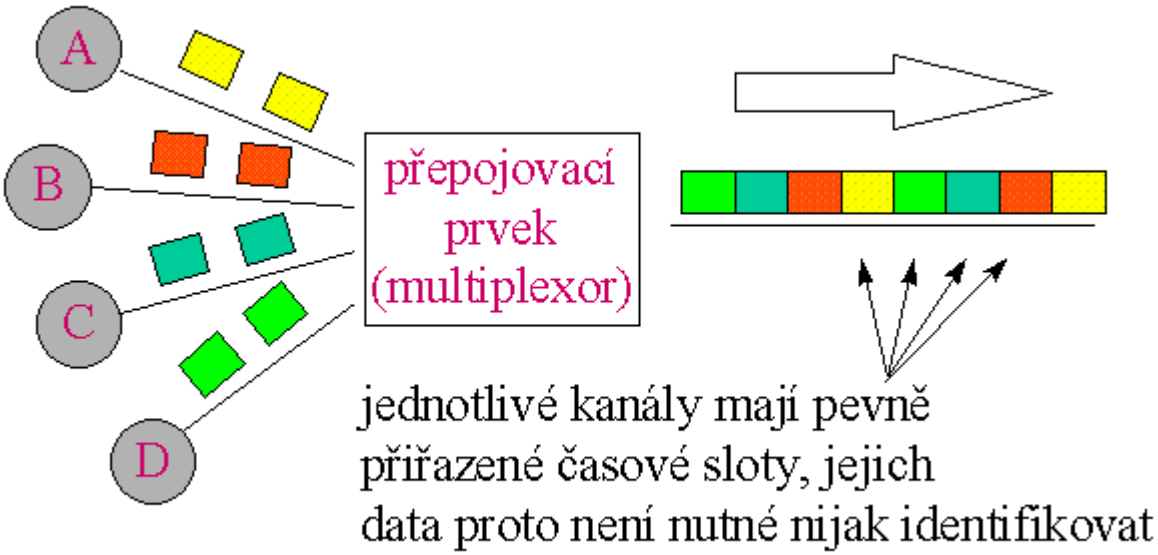
Multiplexování

je termín popisující proces, ve kterém je **více analogových signálů nebo digitálních datových toků kombinováno do jednoho signálu**. Cílem je co možná nejefektivnější využití daného přenosového média. Zařízení provádějící multiplexování se nazývá se **multiplexor** (MUX) a zařízení provádějící převod signálu zpět na jednotlivé signály se nazývá **demultiplexor** (DEMUX).

- **Frekvenční multiplex**
  - Je mnohonásobné využití dostupné kapacity přenosového vedení, radiového nebo optického pásma pomocí frekvenčního (vlnového) dělení celkové šířky pásma na více nezávislých kanálů přiřazovaných pro jednotlivá spojení.
  - Každý kanál je charakterizován svou polohou ve frekvenčním plánu danou nosným kmitočtem a svou frekvenční šířkou.
  - Uživatelský signál se před zařazením do frekvenčního multiplexu musí nejdříve frekvenčně a výkonově upravit a pak pomocí modulátoru, modulační techniky a nosného signálu umístit do určeného pásma a sloučit s ostatními kanály do výsledného signálového průběhu.
  - Na opačném konci přenosového řetězce musí dojít k zesílení a ekvalizaci signálu (signál je přenosem částečně zkreslen), rozdělení na jednotlivé kanály a k demodulaci, čímž je zpět získán původní uživatelský signál.
  - Tato technika se také používá pro oddělení směrů přenosu na jednom vedení. Využívá se jak v analogových, tak i digitálních přenosových systémech.

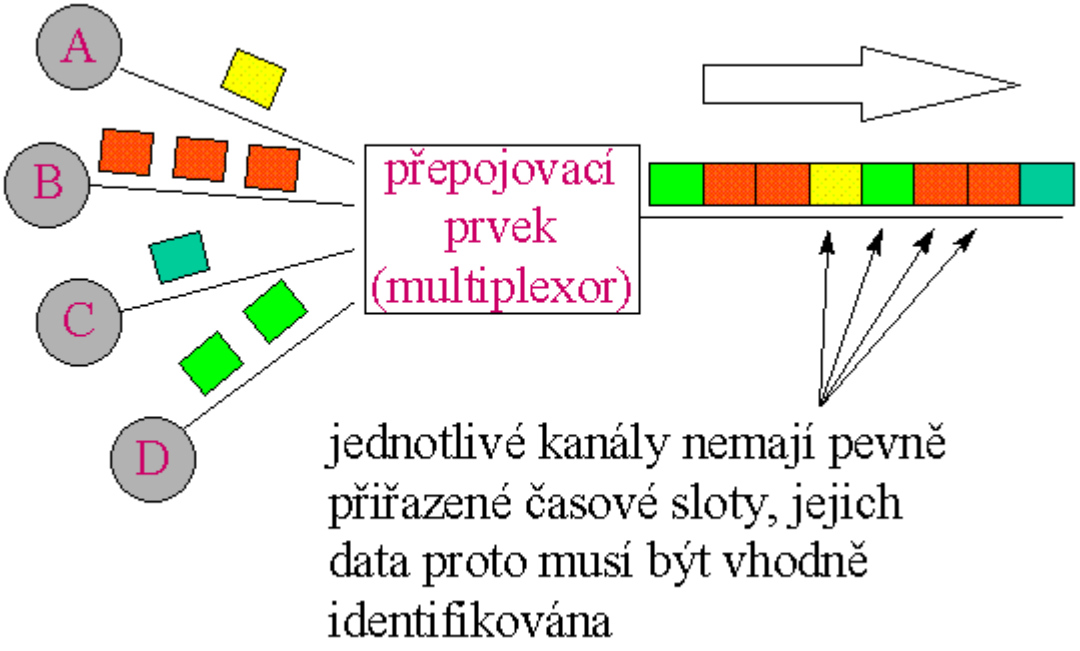


- **Časový multiplex**
  - Jednotlivé signály jsou odděleny tím, že se každý z nich vysílá (přenáší) pouze krátký pevně definovaný časový okamžik.
  - Prakticky ve všech případech se používá rámcové struktury, která je rozdělena na stejně velké timesloty (TS), časové intervaly pro vysílání, pro každý signál jeden.
  - Tento rámec se v čase neustále opakuje a tedy každý signál se přenáší stále se stejnou pravidelností.
  - Pokud budeme mít čtyři signály A,B,C a D, bude vysílání pomocí časového multiplexu vypadat následovně: **ABCD**ABCD**ABCD**ABCD... a tak stále dokola.
  - Hlavní problém spočívá v tom, jak rozlišit kde timeslot pro konkrétní signál začíná a kde končí. Nejčastějším řešením je přidání speciálního timeslotu s přesně definovaným obsahem na začátek rámce, který označuje „toto je začátek rámce“. Protože jsou všechny timesloty stejně dlouhé (časově), stačí na straně přijímače najít tento startovní bod a potom odpočítat potřebnou dobu do začátku hledaného TS.
  - Ve výše uvedeném případě by se tedy vysílal rámec vypadající takto: **SABCD**SABCD**SABCD**SABCD...kde S označuje „startovní“ timeslot



- **Statistický multiplex**

- Statistický multiplex je řešením, které přenosovou kapacitu společné přenosové cesty přiděluje podle momentální potřeby - a dokáže tudíž pružně reagovat na to, kdy jeden dílčí kanál potřebuje více prostoru, a druhému momentálně stačí menší datový tok.
- Jednoduše řešeno, pořad, který momentálně potřebuje, díky složitosti obrazové scény velké rozlišení, si „půjčí“ potřebnou šíři kanálu od programu jiného, kde jsou momentálně nízké nároky na zpracování obrazu.
- Statistický multiplex je jedním z důvodů zpoždění digitálního vysílání oproti analogovému, protože před odesláním signálu je nutné nejdříve spočítat šířku pásma, která bude jednotlivým programům přidělena.



Služby přenosu souborů

http://www.explorer.cz/cz/publish/43/Ftp-prenos-souboru.html
http://4iz110.vse.cz:41004/4iz110/cv4/cv4.html

FTP

- protokol, ale i název služby
- určen pro předávání souborů ze serveru a na server
- pracuje na **aplikační vrstvě**
- jedná se o jeden z nejstarších protokolů a zároveň nejčastěji používanou službu pro předávání dat na Internetu
- funguje na principu **klient - server**
- používá **interaktivní** styl komunikace, umožňuje řízení přístupu (přihlašování login/heslo)
- **porty**
  - **TCP/21** - slouží k řízení, jsou jím přenášeny příkazy
  - **TCP/20** - slouží k vlastnímu přenostu dat

Přenos dat

- přenos dat je **8-bitový**
- **2 režimy** přenosu dat
  - **textový** - dochází ke **konverzi konců řádků**, pokud je zdrojový a cílový systém rozdílný - DOS/Windows používá jako konce řádků posloupnost znaků `CR LF`, unixové systémy (Linux, nové verze MacOS) používají pouze znak `LF`
  - **binární** - v binárním režimu není do dat nijak zasahováno (typicky obrázky, komprimované soubory (RAR, ZIP, ...), atp.)

Nejčastější použití

- **sdílení dat** (hudba, videa, vlastní tvorba, apod.) - *v tomto ohledu ztrácí FTP svou dřívější pozici, protože většina uživatelů (především méně technicky zdatných) používá na použití jednodušší a přístupnější sdílecí servery (ulož.to, czshare.com, ...)*
- **správa** účtů internetových **stránek** (nahrávání zdrojových souborů na server - PHP skriptů, HTML stránek, obrázků,...)

Jak protokol funguje

- FTP server poskytuje data pro ostatní počítače. Klient se k serveru připojí a může provádět různé operace (výpis adresáře, změna adresáře, přenos dat atd.). Operace jsou řízeny sadou příkazů, které jsou definovány v rámci FTP protokolu, proto kdokoliv může vytvořit klienta pro jakékoliv prostředí nebo operační systém.

- Stanoví se pravidla komunikace mezi klientem a serverem. Jedná se o spojovanou spolupráci. Klient naváže spojení se serverem, předá uživatelské jméno a heslo. Po navázání spojení může probíhat práce s adresáři a soubory a přenos dat mezi klientem a serverem. Spojení se obvykle ukončí z podnětu klienta. Úsek komunikace mezi otevřením a uzavřením spojení se nazývá *relace*.
- V případě protokolu FTP se užívá na straně serveru dvou standardních portů (20 a 21). Server na nich očekává a vyřizuje požadavky klientů. Pakety na portu 21 slouží k řízení komunikace - *řídicí kanál*. Port 20 slouží k vlastnímu přenosu dat - *datový kanál*. Porty na straně klienta jsou dynamicky přidělované a přiděluje je OS.
- Řídicí kanál se po dobu relace otevře jen jednou a zůstává otevřen, zatímco datový se po přenosu dat (souboru) uzavře a pro každý datový přenos je nutné kanál znovu otevřít.
- Podle toho, kdo otevírá datový kanál, se rozlišuje aktivní a pasivní režim spojení. Přenos souborů probíhá v textovém nebo binárním režimu v závislosti na typu souboru nebo potřebách uživatele. Chování serveru nebo klientského programu je určeno konfigurací. O komunikaci lze vést na obou stranách spojení protokol.

Anonymní a neanonymní služba přenosu souborů

- **Anonymní servery** umožňují přihlášení uživatele pomocí předem daného a pro všechny uživatele společného uživatelského jména - např. `anonymous` nebo `ftp`. K zadání hesla obvykle stačí adresa elektronické pošty. Tyto servery umožňují prohlížení veřejně přístupných adresářů a stahování souborů na lokální počítač. Zpravidla nepovolují zápis do struktury souborů serveru.

- **Neanonymní servery** vyžadují účet uživatele pro daný server. Účet se skládá se z přihlašovacího jména a hesla. Server určuje oprávnění uživatele k souborům a adresářům. Seznam účtů vytváří správce služby.

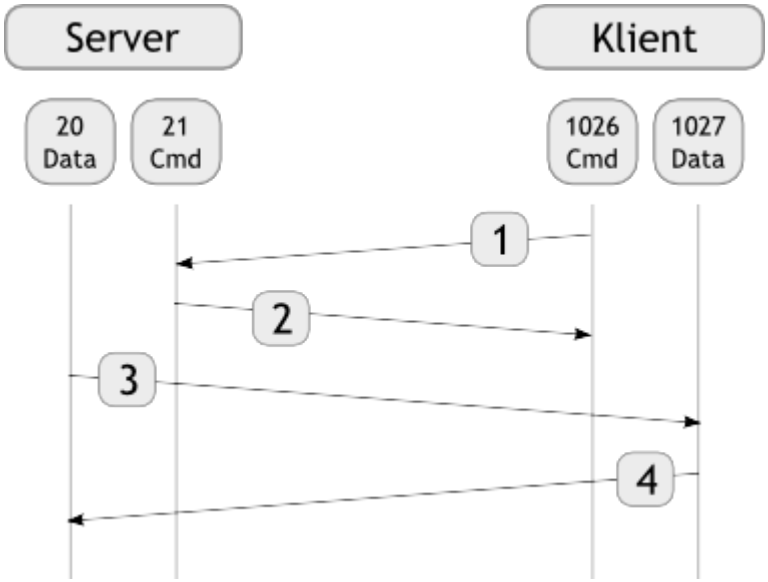
Aktivní a pasivní režim spojení

- připojení k FTP serveru je možné realizovat v aktivním nebo pasivním režimu. Pasivní režim je bezpečnější, ale ne vždy je technicky realizovatelný
- druh spojení je určen podle toho, kdo otevírá datový kanál (klient / server)

Aktivní režim

- jestliže datový kanál **otevírá server**, jde o aktivní režim spojení
- na portu **TCP/20** jsou přenášena data (*data connection*)
- připojení na přenos dat navazuje server a **klient naslouchá**
- klient oznámí serveru příkazem `PORT` svou IP adresu a port, kde bude očekávat příchod datových paketů serveru
- **problém** zpravidla nastává, když se klient připojuje z privátní sítě a jeho adresa je překládána pomocí NATu

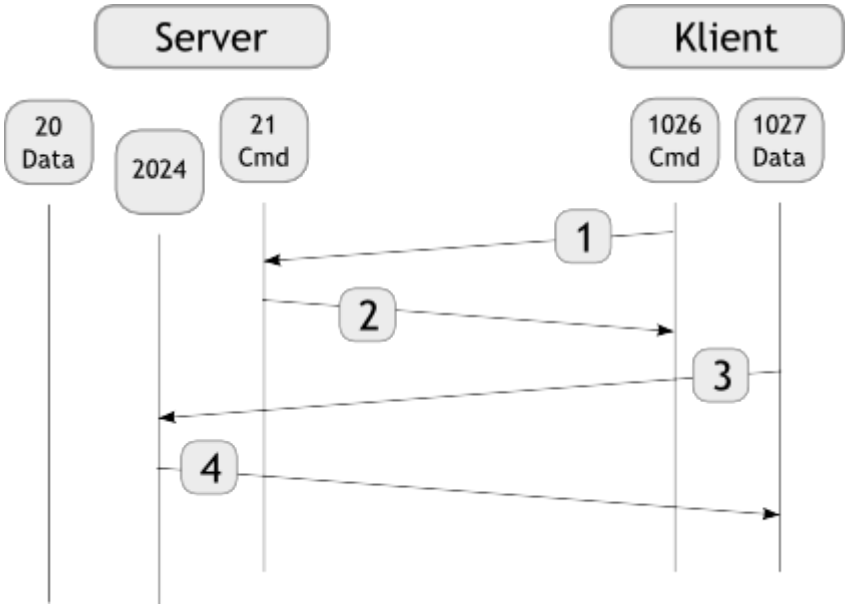
```
PORT 146,102,64,219,4,150
200 command successfully executed
```



Pasivní režim

- připojení pro přenos dat (*data connection*) **navazuje klient**, server naslouchá
- při sestavování *data connection* pošle server klientovi svou IP adresu a TCP port, na kterém server naslouchá a čeká na přenesení dat
- klient nejdříve požádá server příkazem `PASV`, aby přešel do pasivního režimu
  - pokud server souhlasí, pošle na řídícím kanálu odpověď s uvedením své IP adresy a portu, na němž datové spojení očekává

```
PASV
227 Entering Passive Mode (195,113,144,229,220,154)
```



Programy

Serverové programy

- existuje velké množství programů, které fungují jako **FTP server**
- např. **VSFTP**, **ProFTP** a mnoho dalších

Klientské programy

- pro práci s FTP existuje **velké množství** programů - jak **textových** (pro příkazový řádek), tak i **grafických**
- **textově orientované** - např. `lftp`, **MS FTP** - výchozí textový klient ve Windows (příkaz `ftp`) a řada dalších
- **graficky orientované** - v současné době nad textovými klienty převládají
  - čistokrevní FTP klienti - např. **FileZilla**, gFTP (na Linuxu), apod.
  - FTP klienti jako součást správce souborů - např. **Total Commander**, **Průzkumník** ve Windows (program `explorer.exe`), ...
  - FTP integrované do webového prohlížeče - Opera, Firefox, ... - většinou umožňují pouze čtení souborů z FTP serveru - ale už nikoli vytvářet, mazat, nahrávat a editovat soubory a adresáře

Výhody

- serverová část je jednodušší, než běžný HTTP server (neplatí pro odlehčené HTTP servery)
- na rozdíl od HTTP má protokol širší možnosti (nastavení práv, mazání, upload, ...)

Nevýhody

- hesla a soubory jsou ve standardním protokolu zasílána jako běžný text (nejsou šifrovaná)
  - snižuje bezpečnost (ohrožuje jméno, heslo, ale i přenášená data)
  - existují rozšíření FTP protokolu, která tento nedostatek odstraňují (např. FTPS)
- FTP server má delší odezvy
  - nemožnost sloučit přenos více (malých) souborů do jednoho zvyšuje časovou režii i zátěž serveru
- v některých sítích je povolen pouze protokol HTTP (tj. povoleno pouze prohlížení webových stránek) - v takových sítích není možné protokol FTP použít

Zabezpečení

- FTP protokol v současné době už **není** považován za **bezpečný** ☐ byla pro něj definována některá rozšíření
- možnosti zabezpečení komunikace:
  1. **VPN** - privátní tunel, jedná se o zabezpečenou komunikaci i když FTP zabezpečeno není (šifrován je vlastní tunel, nikoli samotná FTP komunikace)

2. **FTPS** - obdoba HTTPS (předají si klíče, pak šifrují)
3. **SFTP**- postaveno nad jiným zabezpečeným protokolem (nejčastěji SSH)
4. **SCP** - oproti SFTP má omezené možnosti (není tak komplexní)
   - programy - např. WinSCP, Putty (terminál),...

FTPS (FTP s podporou SSL/TLS)

- klient se připojuje na port 21, zahajuje nešifrovanou komunikaci a žádá o aktivaci TLS (SSL) před tím, než budou poslána citlivá data
- protokol TLS umožňuje aplikacím komunikovat po síti způsobem, který zabraňuje odposlouchávání či falšování zpráv
- pomocí kryptografie poskytuje TLS svým koncovým bodům autentizaci a soukromí při komunikaci Internetem
- typicky je autentizován pouze server (tedy jeho totožnost je zaručena), zatímco klient zůstává neautentizován
- to znamená, že koncový uživatel (ať člověk či aplikace - např. webový prohlížeč) si může být jist s kým komunikuje
- Další úroveň zabezpečení, při níž oba konce „konverzace“ mají jistotu s kým komunikují, je označována jako vzájemná autentizace
- Vzájemná autentizace vyžaduje nasazení infrastruktury veřejných klíčů (PKI) pro klienty

- TLS zahrnuje **3 základní fáze**
  - dohodu účastníků na podporovaných algoritmech
  - výměnu klíčů založenou na šifrování s veřejným klíčem a autentizaci vycházející z certifikátů
  - šifrování provozu symetrickou šifrou

SFTP (Secure FTP) a SCP (Secure Copy)

- **SFTP**
  - pro přenos dat obvykle využívá protokol **SSH-2**
  - je ale navržen tak, aby ho bylo možné používat i nad jiným protokolem
  - narozdíl od SCP má široké možnosti pro doplňující operace se soubory - umožňuje pokračovat v přerušených přenosech, vypisovat adresáře i odstraňovat soubory na vzdáleném počítači

- **SCP**
  - jednoduchý protokol
  - pro šifrování a autentizaci využívá protokol SSH

- v praxi je rozšířena aplikace **WinSCP** - jedná se o souborový manažer, který je založený na knihovnách Putty (SSH klient) a umožňuje práci v režimu SFTP a SCP.

Pozor! Následující část nesouvisí s protokolem FTP.

SSH

- SSH (Secure Shell) je zabezpečený komunikační protokol
- používají TCP/IP (port 22)
- byl navržen jako náhrada za telnet a další nezabezpečené vzdálené shelly (rlogin, rsh apod.), které posílají heslo v nezabezpečené formě a umožňují tak jeho odposlechnutí při přenosu pomocí počítačové sítě
- šifrování přenášených dat, které SSH poskytuje, slouží k zabezpečení dat při přenosu přes nedůvěryhodnou síť.
- SSH umožňuje bezpečnou komunikaci mezi dvěma počítači, která se využívá

pro zprostředkování přístupu k příkazovému řádku, kopírování souborů, ale také k jakémukoliv obecnému přenosu dat

- také zabezpečuje autentizaci obou účastníků komunikace, transparentní šifrování

přenášených dat, zajištění jejich integrity a volitelnou bezeztrátovou kompresi

- **Výhody**
  - narozdíl od svých předchůdců používá SSH zabezpečený (šifrovaný) komunikační kanál