POČÍTAČOVÉ SYSTÉMY

A SIETE

PETER ŠVEC, ŠTEFAN KOPRDA, JARMILA ŠKRINÁROVÁ , VLADIMÍR SILÁDI

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

itakadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

OBSAH

1 ZÁKLADNÁ KONCEPCIA ČÍSLICOVÉHO POČÍTAČA

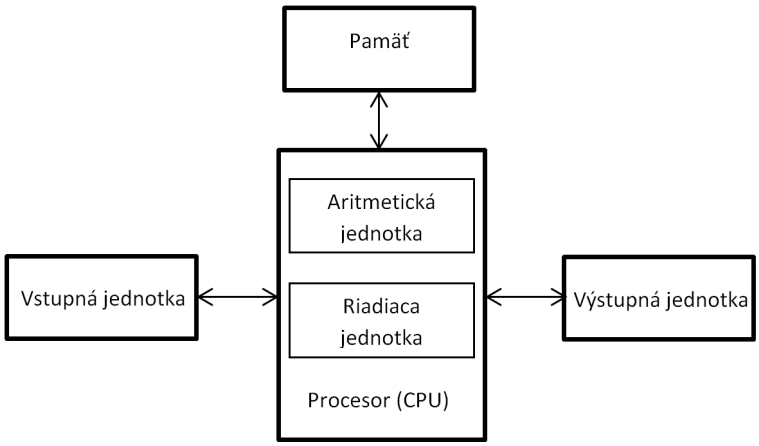
Číslicový počítač je definovaný ako zložitý univerzálny číslicový systém (automat), určený na samočinné vykonávanie požadovanej postupnosti operácií.

Základný koncept počítača, ktorý vykonáva postupnosť operácií na dosiahnutie konečného výsledku, je známy viac ako 150 rokov. Bol použitý v mechanických dekadických počítacích strojoch, ktoré navrhol a čiastočne zostrojil Charles Babbagé. Jeho analytický stroj (Analytical Engine) z r. 1834 obsahuje centrálnu procesorovú jednotku (mechanickú, s dekadickou aritmetikou), pamäť (mechanickú) a vstupnú a výstupnú jednotku (pre dierne karty), t.j. všetky základné časti moderných počítačov. Program a údaje pre tento stroj sú na diernych kartách. Samozrejme, vtedajšia mechanická technológia nedovolila úspešnú realizáciu funkčných zariadení, takže jeho myšlienky zostali viac ako 100 rokov nevyužité. Až vytvorenie elektronických obvodov v 40. rokoch 20. storočia ich umožnilo zaviesť do praxe. V tomto čase John von Neumann predložil základný princíp počítača riadeného tokom inštrukcií, ktorý sa stal základom jednej triedy súčasných počítačov. Táto trieda počítačov býva tiež označovaná ako von Neumannovské počítače. Vyznačuje sa tým, že jednotlivé inštrukcie programu sa vykonávajú postupne za sebou, tak ako sú uložené v pamäti. V súčasnosti existujú aj výpočtové systémy, v ktorých sa inštrukcie nevykonávajú v poradí, v akom sú uložené v pamäti (napr. počítače riadené tokom údajov, neurónové počítače a iné). [1]

1.1 Počítače riadené tokom inštrukcií

Základné črty von Neumannovského počítača sú:

- Pamäť je použitá na uloženie inštrukcií údajov.
- Riadiaca jednotka j e použitá na výber inštrukcií z pamäte.
- Aritmetická jednotka je použitá na vykonávanie špecifikovaných operácií nad údajmi.
- Vstupná jednotka je použitá na vstup údajov, výstupná jednotka na výstup údajov.(Alfa, 1987)



Obrázok 1 Bloková schéma von Neumanovského počítača

Pamäť je množina rovnakých buniek, z ktorých každá je samostatne identifikovateľná svojím poradovým číslom - adresou. Inštrukcie a údaje, uložené v pamäti, sú zakódované dvojkovým kódom.

Inštrukcia (príkaz pre riadiacu jednotku) určuje, aká operácia sa má vykonať a s ktorými údajmi. Inštrukcie sa vykonávajú postupne za sebou, tak ako sú uložené v pamäti. Výnimkou sú skokové inštrukcie. Implicitne sa predpokladá pripravenosť údajov, ktoré vykonávaná inštrukcia požaduje.

Dvojkovo zakódované inštrukcie sú označované ako strojové inštrukcie. Operácie, špecifikované v strojových inštrukciách, sú obyčajne iba jednoduché, napr. aritmetické a logické operácie, posuvy atď., čo poskytuje najväčšiu flexibilitu. Zložitejšie operácie potom vytvára používateľ ako postupnosť inštrukcií. Z danej množiny strojových inštrukcií (inštrukčný súbor procesora)používateľ vyberá inštrukcie na vykonanie požadovaného výpočtu. Táto postupnosť vybraných inštrukcií sa nazýva strojový program počítača. Riadiaca jednotka a aritmetická jednotka sú zvyčajne realizované ako jeden funkčný blok, ktorý sa nazýva centrálna procesorová jednotka (CPU)alebo skrátene procesor. Ak je procesor integrovaný na jedinom polovodičovom čipe, nazýva sa mikroprocesor. Procesor obsahuje niekoľko registrov, ktoré sú použité na uchovávanie špecifických operandov, použitých pri výpočte, adries a riadiacich informácií. Počet registrov a ich funkcia závisí od konkrétneho procesora, ale niektoré registre sú prítomné v každom procesore von Neumannovského počítača. Typickým registrom je programové počítadlo (register PC). Obsahuje adresu nasledujúcej inštrukcie, ktorá sa bude vykonávať.

Princíp práce počítača pozostáva z nasledujúcich krokov:

- vstupná jednotka je zariadenie, ktoré do „počítača posiela vstupné informácie, ktoré sa spracovávajú“. Vstupom môže byť klávesnica, myš, mikrofón, kamera, atď.
- základná, centrálna jednotka je napojená na vstupné a výstupné zariadenia. Je to centrum spracovávania informácií.

Skladá sa z troch častí: operačná pamäť, ALU (Aritmeticko – logická jednotka), riadiaca jednotka, radič.

- operačná pamäť - je priestor, kde sú uložené spracovávané programy a údaje. Môže sa označiť aj ako interná pamäť alebo dočasnú pamäť. Okrem internej pamäte existuje externá pamäť (pamäťové zariadenia), ktoré slúžia najmä na ukladanie, na trvalé uloženie dát.
- ALU – aritmeticko – logická jednotka, je výkonná jednotka, ktorá robí aritmetické výpočty a logické operácie.
- riadiaca jednotka, radič - je miesto, ktoré riadi celý proces toku údajov pomocou riadiacich signálov a zodpovedajúcich stavových funkcií. Na každý riadiaci signál odpovedá príslušná časť poslaním stavového signálu
- Výstupná jednotka podieľa sa na sprostredkovaní, zobrazení spracovaných údajov, dát pre človeka.

Spojenie ALU a riadiacej jednotky je označované ako procesor (mikroprocesor). Von Neumannova architektúra sa vyznačuje spoločnou pamäťou pre inštrukcie aj údaje. Z tohto vyplýva, že je potrebné zabezpečiť, aby procesor neinterpretoval údaj ako inštrukciu a naopak. Prístup procesora k pamäti je totiž rovnaký, či sprístupňuje inštrukciu alebo údaj - používajú sa tie isté adresové, údajové i riadiace signály. Prepojenie medzi jednotlivými časťami počítača je zabezpečená zbernicami. Tieto zbernice majú rôznu šírku a funkciu. Šírka zbernice sa udáva v počte bitov, ktoré zbernica dokáže naraz preniesť (8, 16, 32, 64, ...) Zbernica v počítači predstavuje sústavu vodičov, pomocou ktorých je nejaká jednotka spojená s procesorom, pamäťou a vstupno / výstupnými obvodmi. Je to akási dopravná infraštruktúra, pomocou ktorej sa dopravujú údaje medzi jednotlivými prvkami počítača. Zbernica v sebe zahŕňa aj spôsob komunikácie, komunikačný protokol. Zbernica býva vyvedená na konektor, port, pomocou ktorého je možné pripájať ďalšie zariadenia. Zbernice je možné rozdeliť podľa rôznych kritérií:

Podľa umiestnenia:

- lokálna, vnútorná zbernica - používa signály z procesora. Obvykle sa používa pre prepojenie procesora a pamätí.
- systémová zbernica - prepája signály z mikroprocesora a periférií, prídavných zariadení. Zbernica je navonok reprezentovaná normalizovaným konektorom, tzv. slotom. Do slotov sa zasúvajú karty rozširujúcich periférnych zariadení.

Podľa funkcie:

- dátová zbernica je určená na prenos dát. Podľa toho koľko paralelných drôtov dá- tová zbernica obsahuje rozpoznávame triedu počítača 8, 16, 32 a 64 bitového.
- adresná zbernica vysiela identifikačný kód zariadenia. Adresovaná je každá bunka RAM pamäte, ale aj iné hardwarové zariadenia. Adresa je však len jednou z podmienok prenosu dát. Na spustenie výmeny dát to však nestačí. Spolu s adresou musí proces synchronizovať riadiaca zbernica.
- riadiaca zbernica dáva konečný povel k vykonaniu akcie

Podľa smeru:

- jednosmerná zbernica - signál sa prenáša len jedným smerom,
- obojsmerné zbernica - signál sa prenáša oboma smermi.

Podľa usporiadania vodičov:

- sériová zbernica - signál sa prenáša pomocou jedného, dvoch alebo veľmi malého počtu vodičov,
- paralelná zbernica - signál sa prenáša viacerými vodičmi, ich počet je často určený výsledkom mocniny čísla dva.

Podľa synchronizácie:

- synchrónna zbernica - synchronizovaná s taktom procesora počítača. Synchrónne zbernice sú rýchlejšie a používajú sa na spojenie zariadení s rovnakou prenosovou rýchlosťou,
- multimaster zbernica - dovoľujúca tzv. busmastering, pri ktorom môže byť zbernica riadená viacerými zariadeniami.
- pseudosynchronná zbernica - s oneskorením prenosu dát. asynchrónna zbernica - jednostranne riadená.
- Asynchrónne zbernice sú pomalšie, pretože sa čaká na potvrdenie prenosu od pomalšieho zariadenia. Sú však vhodné na spojenie zariadení s rôznou prenosovou rýchlosťou.

Činnosť všetkých jednotiek počítača je synchronizovaná hodinami (definujú základný strojový takt procesora). Dnešné počítače sa v akomkoľvek prevedení a forme a aj teoretickej rovine podobajú tejto schéme. Je samozrejmé, že evolúcia počítačov sa mierne podpísala aj pod niektoré výnimky, ktoré nie sú obsiahnuté vo von Neumannovej schéme:

- Dnešné počítače dokážu spracovávať niekoľko úloh a teda aj programov naraz – multitasking.
- Počítač môže disponovať viacerými procesormi.
- Existujú aj takzvané vstupno/výstupné zariadenia, z a do ktorých môže byť program a jeho výsledky zapísané a načítané.
- Program sa nemusí zaviesť do pamäti celý, stačí len jeho najdôležitejšia časť, pričom ostatné časti sa zavedú vo chvíli keď sú potrebné
- Program sa nemusí nahrávať do pamäte cez procesor – DMA (Direct Memory Access – priamy prístup do pamäte), šetrí procesorový čas, ktorý sa vďaka tomu môže venovať vykonávaniu programu.

Počítače von Neumannovej sa radia do skupiny počítačov riadených tokom inštrukcií.

ZAPAMÄTAJTE SI!

Zapamätajte si čo je pamäť, inštrukcia, aritmetická jednotka, riadiaca jednotka a aké zbernice poznáme.

1.1.1 Princetonská a Harvardská architektúra

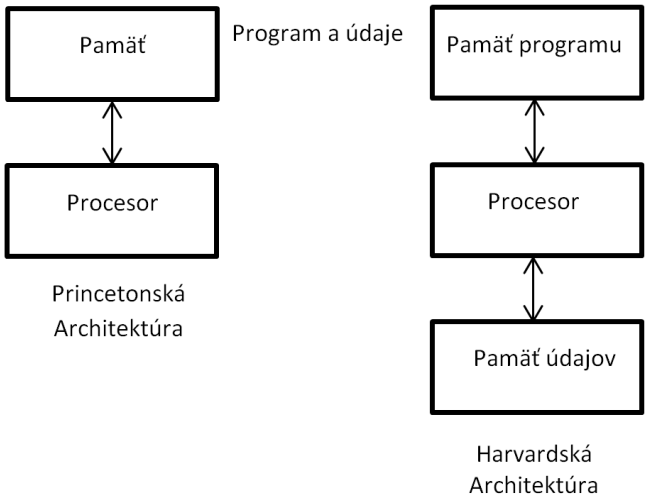
Von Neumannov počítač je predstaviteľom tzv. Princetonskej architektúry počítačov, ktorá sa vyznačuje spoločnou pamäťou pre inštrukcie i údaje. Z tohto vyplýva, že je potrebné zabezpečiť, aby procesor neinterpretoval údaj ako inštrukciu a naopak. Prístup procesora k pamäti je totiž rovnaký, či sprístupňuje inštrukciu alebo údaj - používajú sa tie isté adresové, údajové i riadiace signály. Takéto usporiadanie pamäte potom umožňuje používať aj samomodifikujúce sa programy. Program počas svojho behu môže meniť sám seba. Treba si však uvedomiť, že takáto

situácia môže nastať aj neželane, či už nesprávnym programom, alebo vplyvom poruchy. Niektoré súčasné typy procesorov už vykonávajú kontrolu správnosti prístupu k pamäti vlastnými technickými prostriedkami. V prípade, že sa niektorý prístup k pamäti vyhodnotí ako nesprávny (napr. procesor sa pokúša zapisovať do oblasti pamäte, ktorá je vyhradená na uloženie inštrukcií), automaticky sa generuje výnimka. Túto situáciu potom rieši operačný systém počítača.

Hardvardská architektúra (podľa Howarda Aikena) na rozdiel od von Neumanovej predpokladá existenciu dvoch oddelených pamätí. V prvej sú uložené programy a v druhej sú uložené dáta.

Programový kód a dáta sú uložené v oddelene adresovaných oblastiach pamäte. Môže sa prekrývať čítanie a vykonávanie inštrukcii. Priechodnosť inštrukcii a dát možno zvýšiť: minimalizovaním času potrebného na vykonanie inštrukcie; rozdelením jednotlivých inštrukcii na menšie úseky, prekrývanie cyklov.

Hardvardská architektúra umožňuje paralelné spracovanie dát (niekoľko paralelne zapojených ALU, keď fyzické oddelenie inštrukcii a dát umožňuje súčasný prístup k obom. - RALU rekonfigorovateľné ALU v ktorých načítanie dát sa vykoná 1 impulzom).



Obrázok 2 Organizácia pamäte počítačov s Princetonskou a Harvardskou architektúrov. (http://aplo.fiit.stuba.sk/aps/frames/generuj.php?id=12)

Porovnanie vlastnosti oboch koncepcií:

VON NEUMANNOVÁ KONCEPCIA

Výhody:

- rozdelenie pamäte pre kód programu a dáta určuje programátor,
- riadiaca jednotka procesora pristupuje do pamäte pre dáta a pre inštrukcie jednotným spôsobom,
- používa sa jedna zbernica - jednoduchšia výroba.

Nevýhody:

- spoločné uloženie dát a kódu môže mať pri chybe za následok prepísanie vlastného programu,
- jedna zbernica vytvára úzke miesto, spomaľovanie výpočtov.

HARVARDSKA KONCEPCIA

Výhody:

- program nemôže prepísať sám seba a ani dáta,
- pamäte môžu byť vyrobené odlišnými technológiami,
- každá pamäť môže mať inú veľkosť,
- dve zbernice umožňujú jednoduchý paralelizmus, dá sa pristupovať k inštrukciám aj k dátam súčasne.

Nevýhody:

- dve zbernice požadujú vyššie nároky na vývoj riadiace jednotky procesora a zvyšujú aj náklady na výrobu celého počítača.
- nevyužitá časť pamäte dát sa nedá použiť pre program a obrátene.

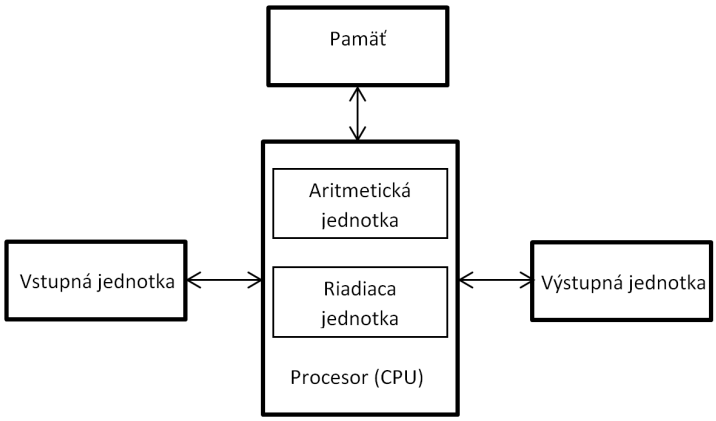
CVIČENIE – OTÁZKY!

Vymenujte výhody a nevýhody obidvoch používaných architektúr.

Nakreslite blokovú schému Princetonskej architektúry.

1.2 Počítače riadené tokom inštrukcií

Tieto počítače nevykonávajú inštrukcie postupne za sebou, tak ako sú uložené v pamäti, ale vykoná sa práve tá inštrukcia, ktorá má pripravené údaje. Ak má viac inštrukcií pripravené svoje údaje, tieto inštrukcie sa vykonajú paralelne. Treba poznamenať, že je potrebné vždy vytvoriť toľko kópií vstupných údajov, koľko je inštrukcií, ktoré ich budú potrebovať. Počítače riadené tokom údajov predstavujú osobitnú triedu paralelných počítačov. Sú to počítače novej generácie s vysokou výkonnosťou. Ďalšou významnou vlastnosťou je, že programu sa prispôsobuje štruktúra technických výpočtových prostriedkov. Na obr. 3 je principiálna bloková schéma počítača, riadeného tokom údajov.
V pamäti inštrukcií sa nachádzajú všetky inštrukcie programu. Arbitračná sieť zisťuje, ktorá inštrukcia (inštrukcie) je pripravená na vykonanie (t.j. ktorá má pripravené všetky svoje vstupné údaje). Túto inštrukciu potom vyberie a pridelí ju na vykonanie niektorému voľnému procesoru z poľa procesorov. Keď procesor vykoná príslušnú inštrukciu, pošle výsledok do distribučnej siete. Distribučná sieť pridelí výsledok všetkým inštrukciám, ktoré v pamäti inštrukcií na tento údaj čakajú.



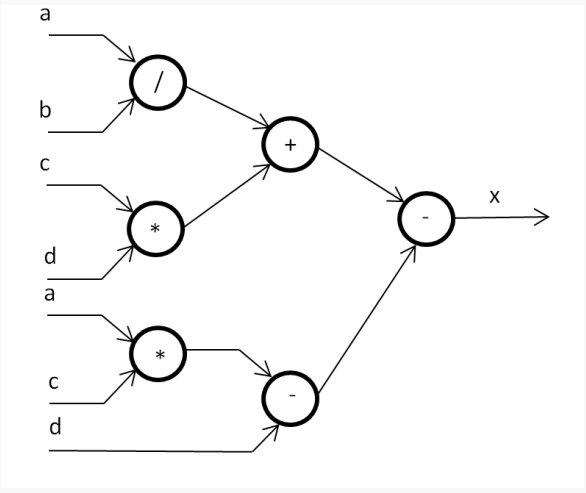
Obrázok 3 Bloková schéma počítača riadeného tokom údajov (http://aplo.fiit.stuba.sk/aps/frames/generuj.php?id=12))

Program pre počítač riadený tokom údajov sa obyčajne zobrazuje ako orientovaný graf, v ktorom uzly reprezentujú asynchrónne aktívne členy (operátory, inštrukcie, úlohy) a hrany reprezentujú komunikačné cesty na, prenos a smerovanie správ(údajových balíkov, operandov), generovaných uzlami počas ich aktivácie alebo prijímaných z externého prostredia počas výpočtu. Tento graf sa nazýva graf programu.

ÚLOHA – RIEŠTE!

Nakreslite graf programu pre realizáciu výpočtu: x = (a/b + c*d) - (a*c - d)

Riešenie:



1.3 Klasifikácia počítačov podľa aplikačného určenia

- Univerzálne počítače predstavujú prevažnú väčšinu systémov, určených na riešenie vedecko-výskumných, inžinierskych, administratívnych a iných úloh. Niektoré triedy

úloh riešia s väčšou, iné triedy úloh s menšou efektívnosťou. Tieto počítače používajú štandardné univerzálne procesory s inštrukčným súborom, ktorý je z hľadiska spracovania informácie úplný a štandardné prostriedky na komunikáciu s hlavnou pamäťou a vstupno/výstupným podsystémom.

- Problémovo orientované počítače sú určené na riešenie problémov z danej triedy, napr. na spracovanie signálov. Používajú procesory s prispôsobeným alebo špeciálne navrhnutým súborom inštrukcií a špecializované periférne zariadenia na efektívne riešenie úloh danej triedy.

- Aplikačne špecifické počítače sú svojou architektúrou orientované na riešenie jedného problému (napr. riadenie nejakého technologického procesu). Z danej úlohy vyplýva špecifikácia architektúry systému a jeho implementácia. Neznamená to, že počítače tejto triedy nevyužívajú štandardné procesory. Obyčajne však obsahujú špeciálne, na mieru vytvorené technické prostriedky a programové vybavenie. Do tejto triedy sa zaraďujú aj tzv. vnorené (embedded) systémy, v ktorých je počítač neoddeliteľnou súčasťou riadeného zariadenia.

1.4 Klasifikácia počítačov podľa architektonickej koncepcie

Podľa architektonickej koncepcie existuje viacero klasifikácií počítačov. *Flynnova klasifikácia* vychádza z počtu súčasne spracúvaných tokov inštrukcií a tokov údajov v počítači. Na základe tohto kritéria Flynn klasifikuje 4 triedy počítačov:[3]

- SISD (Single Insíruction stream Single Data stream). Jeden tok inštrukcií, jeden tok, údajov. Táto architektúra predstavuje architektúru Voji Neumannovho počítača, kde jeden procesor interpretuje jeden prúd inštrukcií (strojový program) a v zodpovedajúcom procese sa spracúva jeden prúd údajov.

- SIMD (Single Instruction stream Multiple Data stream). Jeden tok inštrukcií, viac tokov údajov. Táto architektúra má významné použitie pri tvorbe paralelných počítačov. Uplatňuje sa najmä pri tvorbe systémov, ktoré vykonávajú vektorové a maticové operácie, t.j. súčasne sa vykonáva jedna operácia s viacerými údajovými štruktúrami rovnakého typu. Jeden program sa vykonáva súčasne s viacerými prúdmi údajov vo viacerých procesných elementoch (v tzv. pasívnych procesoroch, ktoré dostanú inštrukciu z riadiacej jednotky a vykonajú ju). Tieto systémy dosahujú podstatne lepšie výsledky ako architektúra SISD pri riešení úloh uvedenej triedy.
- MISD (Multiple Instruction stream Single Dala stream). Niekoľko tokov inštrukcií, jeden tok údajov. Táto trieda predstavuje určitú teoretickú koncepciu, ktorá sa v počítačoch ako v celkoch neaplikuje a zatiaľ sa neobjavil ani jej praktický význam.

- MIMD (Multiple Instruction stream Multiple Data stream). Viacnásobný tok inštrukcií, viacnásobný tok údajov. Architektúra MIMD predstavuje veľmi širokú triedu paralelných systémov, do ktorej zaraďujeme multiprocesorové a multipočítačové systémy s paralelným spracovaním. Ide o paralelné spracovanie s oddelene prebiehajúcimi paralelnými procesmi,

riadenými samostatnými procesormi. Multiprocesorový systém je paralelný počítač, obsahujúci niekoľko procesorov, ktoré majú vlastnú alebo zdieľanú pamäť a na komunikáciu s okolím používajú spoločné vstupné a výstupné zariadenia. Multipočítačový (distribuovaný) systém sa skladá z niekoľkých počítačov s možnosťou vzájomnej komunikácie, ktoré sú schopné aj samostatnej činnosti. Patria sem aj počítačové siete.

CVIČENIE – ANALYZUJTE!

Analyzujte pojem architektonická koncepcia

1.5 Procesor, základná doska, pamäť

Procesorom rozumieme elektronický obvod, schopný spracovávať dáta na základe riadiacich inštrukcií tvoriacich program .

Procesor je jadrom, základnou súčiastkou každého počítača. Procesory najstarších generácií počítačov boli tvorené množstvom samostatných (diskrétnych) súčiastok, vytvárajúcich príslušné elektronické obvody. S príchodom integrovaných obvodov podliehajú miniaturizácii aj obvody procesorov. V r. 1971 prichádza prvý jednoduchý procesor, riešený ako jediná súčiastka a označuje sa ako mikroprocesor. Tento obvod vyvinula firma Intel pre elektronické kalkulačky a dostal označenie i4004. Mikroprocesorom teda rozumieme programovateľný obvod, ktorý dokáže podľa určitého programu spracovávať dáta a vykonávať úlohy. Obvod je riešený formou jedinej súčiastky. Základom súčiastky je monokryštalická kremíková platnička s plochou niekoľkých mm2 , nazývaná chip. Mikroprocesor z hľadiska technológie výroby Z obvodového hľadiska sú základným funkčným prvkom mikroprocesorov unipolárne tranzistory, vyrábané P-MOS, N-MOS či CMOS technológiou. Moderné procesory vychádzajú z NMOS, príp. CMOS technológie, využívajú však mnohé špeciálne polovodičové efekty ako sú plávajúce hradlá, tunelingy „horúcich“ elektrónov a pod. Štruktúry mikroprocesorov sa vyrábajú litografickými technológiami. Veľmi dôležitým parametrom je rozlíšenie litografickej štruktúry, ktoré udáva, aký najmenší rozmer dokážeme v štruktúre vyrobiť. Rozlíšenie sa udáva v desatinách mikrometra či v nanometroch. Čím sú rozmery menšie, tým sa dosiahne menšia parazitná kapacita aj parazitná indukčnosť štruktúr a tým sa procesoru umožňuje pracovať na vyššej frekvencii. Zmenšovanie štruktúry umožňuje používať aj nižšie napájacie napätie a v neposlednom rade aj zvyšovať hustotu súčiastok na chipu.

Zvyšovanie hustoty súčiastok jednak umožňuje zvyšovať počet tranzistrov na chipu pri nezmenených rozmeroch, ale aj zmenšovať plochu chipu pri rovnakom počte tranzistorov. Menšia plocha chipu umožňuje na jedinom kremíkovom plátku (wafferi) vytvoriť viacero chipov, čo sa významne premieta do znižovania výrobných nákladov na jeden chip. Mikroprocesor ako základ každého počítača sa nazýva aj Central Processing Unit – CPU.

Pamäť je jednou zo základných častí číslicového počítača. Slúži na uchovávanie programov a údajov, s ktorými programy pracujú. Ideálna pamäť má vysokú kapacitu, je rýchla (čo najrýchlejšia – aby zbytočne nespomaľovala prácu procesora) a pritom je cenovo dostupná.

Tieto požiadavky sú navzájom protichodné a zatiaľ ich nie je možné všetky splniť súčasne. Preto sa v číslicových počítačoch používa niekoľko druhov pamätí, ktoré tvoria hierarchický systém.

Pamäť v procesore je tvorená registrami, ktoré procesor obsahuje. Je to najrýchlejšia pamäť, pretože je možné s ňou pracovať tak rýchlo, ako pracuje procesor. Okrem toho odpadá pri práci s ňou potreba použiť zbernicu počítača na prenos údajov. Samozrejme táto pamäť má veľmi malú kapacitu – maximálne niekoľko desiatok registrov.

Cache pamäť slúži ako vyrovnávacia pamäť, pretože umožňuje vyrovnávanie rýchlosti prenosu údajov medzi procesorom a hlavnou pamäťou. Je teda podstatne rýchlejšia ako hlavná pamäť, ale zároveň je rádovo menšia. Princíp vyrovnávacej pamäte spočíva v tom, že časť údajov sa presunie z hlavnej pamäte do vyrovnávacej a tým sa zvýši rýchlosť prístupu ku informáciam zo strany procesora. Je možné použiť viacero pamätí typu cache, pričom sa líšia rýchlosťou a veľkosťou. Potom hovoríme o vyrovnávacej pamäti prvej, druhej, prípadne tretej úrovne (L1, L2 resp. L3 cache).

Hlavná (primárna) pamäť počítača sa tiež nazýva operačná pamäť („ramka“). Tvorí základ pamäťového podsystému. Obsahuje práve vykonávaný program a spracúvané údaje. Kapacita operačnej pamäte sa v súčasnosti bežne pohybuje v oblasti stoviek megabajtov, prípadne až v oblasti gigabajtov. Tiež je možné sledovať neustálu snahu o zrýchľovanie hlavnej pamäte.

Sekundárna pamäť slúži na uchovanie informácií, ktoré sa momentálne nepoužívajú. Do tejto oblasti patria predovšetkým pevné disky. Táto pamäť tiež slúži na permanentné uloženie údajov, keďže predošlé typy pamätí pri výpadku napájania zvyčajne svoj obsah strácajú.

Záložná pamäť – ako napovedá už jej názov, slúži predovšetkým na archiváciu informácií, prípadne na ich prenos. Sem patria USB kľúče, CD-ROM/RW, DVD-RW/ROM, diskety, magneto-optické disky, v minulosti sa dosť využívali kazetovo-páskové jednotky.

Medzi jednotlivými pamäťami sa prenášajú údaje po blokoch. Veľkosť blokov je v jednotlivých prípadoch rôzna. Pri prenose procesor – cache je to niekoľko bajtov (1 – 4 B), cache – hlavná pamäť: 8 – 128 B, hlavná pamäť – sekundárna pamäť: 512 B. Tieto hodnoty sú samozrejme odlišné pre rôzne počítače.

Celý pamäťový podsystém je riadený pomerne zložitými riadiacimi obvodmi. Ich úlohou je zabezpečiť, aby bol potrebný údaj vždy „poruke“, čiže v cache pamäti (aby sa obmedzilo čakanie procesora na minimum). Ak požadovaný údaj v cache nie je, dochádza ku chybe bloku a je potrebné načítať do cache pamäte ten správny blok z hlavnej pamäte.

Rozdelenie pamätí:

Z historického hľadiska

- kondenzátorové
- feritové
- polovodičové

Podľa spôsobu prístupu k údajom:

- pamäte s ľubovoľným (náhodným) prístupom (Random Access Memory – RAM)
- pamäte s priamym prístupom (Direct Access Storage Media – DASM)
- pamäte s asociatívnym prístupom (Content Access Memory – CAM)

Podľa smeru prístupu

- RWM (Read Write Memory)
- xROM (Read Only Memory)

Veľmi často sa pamäte typu RWM označujú ako RAM, hoci skratka RAM nehovorí nič o možnostiach čítania či zápisu do pamäti, ale len o rýchlosti prístupu k danej pamäťovej bunke. V skutočnosti aj pamäte ROM sú väčšinou typu RAM!

Podľa energetickej závislosti

- energeticky závislé pamäte: ich obsah sa po prerušení napájania stráca. Sem patria najmä SRAM a DRAM (cache a hlavná pamäť)
- energeticky nezávislé pamäte: obsah pamätí je nezávislý na napájaní. Sú to všetky typy ROM pamätí, pevné disky, CD/DVD-ROM

Podľa spôsobu udržania informácie

- statické (SRAM): informácia je po zápise zachovaná až do jej prepísania. Ako základný stavebný prvok sa používajú bistabilné klopné obvody, napríklad typu D (pre každý bit jeden). Používajú sa napríklad v cache pamätiach
- dynamické (DRAM): tieto pamäte využívajú na uloženie jedného bitu informácie parazitné kapacity, v ktorých buď je alebo nie je uložený náboj. Tento náboj sa postupne vybíja, preto je potrebné pravidelne obnovovať obsah pamäte (refresh). Obnovovanie sa uskutočňuje čítaním pamäte. DRAM pamäte sa najčastejšie využívajú ako hlavná pamäť, pretože nie sú príliš drahé a môžu mať pomerne veľkú kapacitu. Ak sú tieto pamäte synchrónne, označujú sa SDRAM. Novšie typy, umožňujúce dvojnásobne rýchly prenos informácií sa označujú ako DDR SDRAM (Double Data Rate) alebo len DDRAM

Virtuálna pamäť

V číslicových počítačoch rozlišujeme pamäť na fyzickú a virtuálnu. Pod fyzickou pamäťou sa myslí operačná (hlavná) pamäť – „ramka“. Pod pojmom „virtuálna pamäť“ sa často chápe len „swapovanie“ pamäte. V skutočnosti virtuálna pamäť predstavuje taký spôsob menežovania pamäte, kedy operačný systém prezentuje nespojité časti pamäte ako jeden spojitý celok. To znamená, že aplikácia „má dojem“, že jej údaje sú uložené v jednom súvislom bloku pamäte, pričom fyzicky môžu byť údaje uložené na rôznych miestach v pamäti, prípadne dokonca aj v rôznych druhoch pamäti (RAM, disk ...). Swapovanie je len jednou z techník používaných pri práci s virtuálnou pamäťou.

2 POKROČILÉ ARCHITEKTÚRY POČÍTAČOV

Procesor (CPU – Central Processing Unit) je hardvérové zariadenie určené na spracovanie a vykonávanie inštrukcií. Zatiaľ čo základná doska zabezpečuje komunikáciu medzi zariadeniami, procesor celý systém oživí - vydáva jednotlivým zariadeniam príkazy a riadi ich na základe inštrukcií programu. Niektoré inštrukcie spracováva sám, pri spracovaní iných využíva ďalšie komponenty.(napr. operačnú pamäť, disky, zbernice atď.). V súčasnosti používané procesory označujeme ako mikroprocesory na vyjadrenie kontrastu s prvými procesormi, ktorých veľkosť sa blížila veľkosti obytnej miestnosti. Fyzicky ide o súčiastku, ktorú tvorí kremíková doštička obsahujúca na malej ploche (niekoľko cm2) milióny tranzistorov(v minulosti) v súčasnej dobe najlepšie mikroprocesory obsahujú 1 až 3 miliardy tranzistorov. Tieto na základe spínania a vypínania riadia ostatné komponenty systému.

2.1 Procesory CISC a RISC

V dnešnej dobe sa ustálilo delenie počítačov do dvoch základných kategórií podľa typu použitého procesora:

- CISC - počítač so zložitým súborom inštrukcií (Complex Instruction Set Computer)
- RISC - počítač s redukovaným súborom inštrukcií (Reduced Instruction Set Computer)

Či už je toto delenie akokoľvek nepresné, treba ho považovať za obvyklé. V dnešnej dobe totiž prakticky neexistujú procesory, ktoré by znášali "čisté" rysy RISC a CISC, vždy ide o kompromis medzi oboma smermi. Z hľadiska historického vývoja je dôležité sa venovať okamihu, keď sa vývoj procesorov rozdelil do dvoch vetiev. Vývoj procesorov, ktoré následne dostali označenie CISC, smeroval na konci 70. rokov k nezadržateľnému rastu ich zložitosti a tak sa objavili prvé pokusy o celkové zjednodušenie štruktúry procesorov. Vznikla tak celá nová kategória procesorov, dnes označovaná ako RISC.

VÝKLAD

Vznik procesorov RISC

V 70. rokoch výskumy frekvencie inštrukcií ukázali, že programátori a kompilátory používajú strojové inštrukcie veľmi nerovnomerne. Počet najfrekventovanejších inštrukcií je obmedzený na veľmi úzku časť inštrukčného súboru.

Na univerzite v Berkel vznikol v roku 1982 procesor RISC, od ktorého názvu je odvodený celý smer vývoja počítačov. V roku nasledujúcom bol na Stanfordskej univezitě realizovaný procesor M ips (Mikroprocesor without Interlocked pipelining stages - procesor bez vzájomne sa blokujúcich sekciou prúdového spracovanie). Z týchto projektov potom vzniklo niekoľko generácií priemyselne vyrábaných RISC procesorov.

Prvé sériové počítače RISC boli na trh dodávané až v polovici 80. rokov, ale vyrábali sa technológií sa stupňom integrácie MSI a boli veľmi rozmerné. Skutočný rozvoj nastal až s využitím vyššieho stupňa integrácie v sériovej výrobe.

Vývoj mikroprocesorov typu RISC je neoddeliteľný od vývoja radu CISC. Vyplýva to z dlhoročnej skúsenosti vývojárov s vývojom a realizáciou CISC procesorov. Dnes neexistuje žiadny čistý RICS alebo CISC procesor. Každý moderný procesor v sebe uplatní rysy z oboch kategórií. Je skôr vecou rozhodnutia výrobca a jeho marketingu, kam bude ich produkt zaradený. Môže sa rozhodovať podľa toho, akých vlastností má procesor viac, ale často je to ťažké.

Vlastnosti architektúry RISC

Bezpochyby najtypickejšie vlastnosti charakterizujúce počítača RISC je malý inštrukčný súbor. Pre túto vývojovú vetvu si ale vývojári stanovili celý rad ďalších vcelku zásadných kritérií, charakterizujúcich metodiku návrhu nielen procesora, ale celého počítača. Procesoru je potrebné prenechať len tú činnosť, ktorá je nevyhnutne potrebná a preniesť ďalšie potrebné funkcie do architektúry počítače, programového vybavenia a kompilátora.

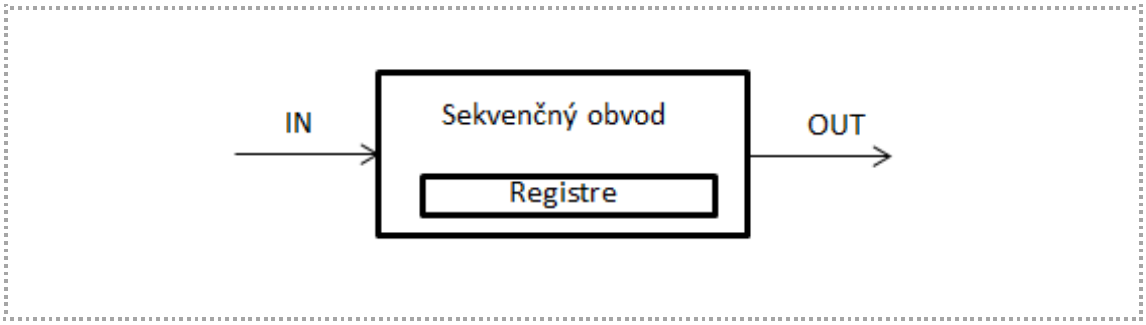
Výsledkom návrhu konštrukcie procesora RISC sú najmä tieto vlastnosti:

- V každom strojovom cykle by mala byť dokončená jedna inštrukcie (pozor, to neznamená, že jej vykonanie trvalo jeden stroj. cyklus).
- Mikroprogramové radič môže byť nahradený rýchlejším obvodovým radičom, používať zreťazené spracovanie inštrukcií.
- Celkový počet inštrukcií a spôsobov adresovanie je malý.
- Dáta sú z hlavnej pamäte vyberaná a následne ukladané výhradne len pomocou dvoch inštrukcií LOAD a STORE.
- Inštrukcie majú pevnú dĺžku a jednotný formát, ktorý vymedzuje vý znam jednotlivých bitov.
- Je použitý vyšší počet registrov.
- Zložitosť sa z technického vybavenia presúva čiastočne do optimalizujú ceho kompilátora.

Všetky uvedené vlastnosti tvoria dobre premyslený a previazaný celok. Napr. navýšenie počtu registrov súvisí s obmedzením komunikácie s pamäťou na dve inštrukcie LOAD a STORE. Ak ostatní inštrukcie nemôžu používať pamäťové operandy, je potrebné mať v procesore uložené viac dát. Ďalší súvislosť je zrejmá medzi zreťazeným spracovaním a formátom inštrukcií. jednotná dĺžka inštrukcií dovoľuje rýchlejší výber inštrukcií z pamäte a tým zaisťuje lepšie plnenie frontu inštrukcií (pozri ďalšia kapitola). Jednotný formát potom zjednodušuje dekódovanie inštrukcií.

Výsledné počítače vytvorené podľa týchto pravidiel prinášajú výhody ako pre užívateľov, tak i pre výrobcov. Skracuje sa vývoj procesora a spravidla už prvý realizovanej čipy fungujú správne. Architektúra RISC má aj svoje nedostatky, napriek tomu sa väčšina výrobcov CISC procesorov uchýlila pri výrobe procesorov k realizácii čoraz väčšieho počtu vlastností architektúry RISC.

Medzi nevýhody RICS architektúry patrí napríklad nutný nárast dĺžky programov, tvorených obmedzeným počtom inštrukcií a tiež vďaka jednotnej dĺžke všetkých pokynov. Spomalenie, ktoré by z toho malo nevyhnutne plynúť, sa ale v praxi nepotvrdilo. Je potrebné si uvedomiť, ako malé percento inštrukcií muselo byť rozpísané a zreťazené spracovanie inštrukcií zásadným spôsobom urýchľuje prácu procesora.



Obrázok 4 Procesor CISC ako sekvenčný obvod

ÚLOHA 2 – RIEŠTE!

Vytvorte príklad možností krokov spracovávania inštrukcie.

Riešenie:

Krok	Význam
1.	VI Výber inštrukcie
2.	DE Dekódovanie
3.	VA Výpočet adresy
4.	VO Výber operanda
5.	PI Prevedenie inštrukcie
6.	UV Uloženie výsledkov

2.2 Prúdové spracovanie inštrukcií

Predstavme si procesor ako sekvenčný obvod podľa obrázku 4. Vstupom sú strojové inštrukcie a dáta z pamäte. Z výstupu sa dáta ukladajú späť do pamäte. Každá inštrukcia teda musí prejsť celým obvodom a kým sa neuložia výsledky, nemožno začať vykonávať inštrukciu nasledujúce. Prevedenie inštrukcia musí prejsť vždy rovnakými fázami. V zjednodušenej variante si môžeme

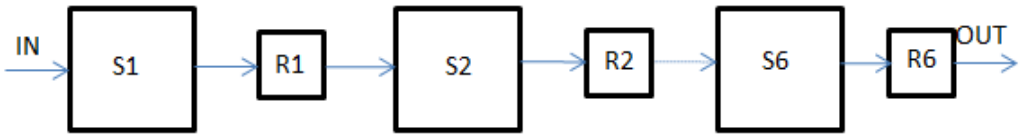
predstaviť, že inštrukcie sa musia vybrať z pamäte, dekódovať, vypočítať adresa operandu, pripraviť dáta, inštrukciu vykonať a nakoniec uložiť výsledky. Pokiaľ bude pre vykonanie jedného elementárneho úkonu potrebný jeden cyklus, môžeme vykonávanie inštrukcií názorne zobraziť v tabuľke 1. Inštrukcia I2 sa nezačne vykonávať skôr, než je uložený výsledok inštrukcie I1. Ak bude jeden časový cyklus označený ako TM, budeme potrebovať pre vykonanie jednej inštrukcie 6 cyklov. Prevedenie každé ďalšie inštrukcie potom teda bude vyžadovať opäť 6 cyklov. Zatiaľ však nepožadujeme, aby mali všetky cykly rovnakú dĺžku.

Keby sa nám teda podarilo osamostatniť jednotlivé časti sekvenčného obvodu z obrázka 4 na samostatné obvody, aby každému obvodu zodpovedala jedna fáza spracovania inštrukcie, mohli by sme si ho predstaviť ako postupnosť na seba nadväzujúcich zreťazených jednotiek (v podstate ide o princíp výrobnej linky, ako ho poznáme z mnohých iných odvetví).

Tabuľka 1 Postup prevádzania inštrukcií procesorom CISC

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
VI	I1						I2						...
DE		I1						I2					
VA			I1						I2				
VO				I1						I2			
PI					I1						I2		
UV						I1						I2	

Medzi jednotlivé fázy spracovania však musí byť pre medzivýsledky zaradený register, ktorý slúži ako odovzdávacie miesto medzi po sebe idúcimi obvodmi. Z tohto odovzdávania medzivýsledkov plynie isté malé meškanie. To možno však v celkovom prínose zreťazenie zanedbať. Ešte sme však nevyslovili jeden dôležitý predpoklad, aby uvedený princíp zreťazenia mal čo najväčší prínos, musia byť všetky fázy spracovania rovnako časovo náročné. Inak je jasné, že najpomalší článok zreťazenia bude brzdiť všetky ostatné. Keď sa nám podarí rozdeliť vykonávanie inštrukcie na jednotlivé úkony, pričom časový cyklus potrebný pre každú fázu spracovania bude rovnaký, môžeme si spracovávanie inštrukcií I1 ÷ I7 znázorniť v tabuľke 2. Je vidieť, že pre vykonanie 7 inštrukcií nám stačí 12 cyklov. V tabuľke 1 sme počas rovnakého počtu cyklov vykonali iba 2 inštrukcie. Teoreticky to znamená, že v nekonečnom čase nám N-úrovní zreťazenia zrýchli vykonávanie inštrukcií N x. V praxi je však celkový prínos zreťazenia limitovaný mnohými ďalšími hľadiskami. Jednak je obmedzená rýchlosť na vstupe a výstupu zreťazenej jednotky a taktiež dochádza počas vykonávania k problémom s používaním obmedzeného množstva pomocných obvodov (registre, zbernica, atď.). Spomenuli sme aj spomalenie odovzdávaním medzivýsledkov. Ako najkritickejšie sa však ukazuje problém plnenia zreťazenej jednotky a tým vzniknuté fronty rozpracovaných inštrukcií. Najmä podmienené skoky znehodnocujú front rozpracovaných inštrukcií (inštrukcie sa spracovávajú sekvenčne a ak sa vykonávanie programu presunie na inú adresu podmieneným skokom, môžeme už rozpracované inštrukcie zahodiť). A čím je hĺbka zreťazenia väčšia, tým sa zvyšujú aj straty (tu je vidieť, ako môže byť väčšia hĺbka zreťazenie kontraproduktívne). Na základe predchádzajúceho vieme, že počet skokových inštrukcií v programoch je veľmi vysoký, prakticky každá šiesta inštrukcia predstavuje podmienečný skok. Na základe známych technický parametrov možno hľadať optimálnu hĺbku zreťazenia.



Obrázok 5 Zreťazené spracovanie procesorom RISC

Tabuľka 2 Zreťazené prevádzanie inštrukcií procesorom RISC

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
VI	I1	I2	I3	I4	I5	I6	I7	...				
DE		I1	I2	I3	I4	I5	I6	I7				
VA			I1	I2	I3	I4	I5	I6	I7			
VO				I1	I2	I3	I4	I5	I6	I7		
PI					I1	I2	I3	I4	I5	I6	I7	
UV						I1	I2	I3	I4	I5	I6	I7

2.3 Virtualizácia

Vznik termínu virtualizácia siaha do histórie a to do 60-tych rokov 20. storočia. Je spojený so systémom IBM M44/44X, na ktorom bolo možné simulovať viaceré virtuálne stroje pomocou hardvérových a softvérových prostriedkov. (HOOPES, 2009 s. 4) Virtualizácia je metodika, ktorá porušuje štandardné paradigma výpočtovej architektúry oddelením operačného systému a jeho aplikácií od fyzického hardvéru, na ktorom bežia. V dôsledku čoho môžu IT organizácie dosiahnuť väčšie využitie zdrojov a flexibilitu. Virtualizácia umožňuje viac virtuálnych strojov, často s rôznorodými operačnými systémami, bežať izolovane vedľa seba, na rovnakom fyzickom stroji. Každý virtuálny stroj má svoj vlastný súbor s operačným systémom a aplikáciami a je mu pridelená časť hardvéru (CPU, pamäť, sieťové rozhrania, disk, grafická karta), na ktorom je operačný systém a jeho aplikácie spustené. Operačný systém vidí pre neho vyčlenenú množinu hardvéru a nevie o iných hosťovaných operačných systémoch bežiacich na rovnakej fyzickej hardvérovej platforme. Po roku 1990 sa začalo intenzívnejšie pracovať na využívaní virtualizácie pri potrebe znižovať náklady, dbať na bezpečnosť a ponúkať lacnejší hardvér. Dnes je virtualizácia jednou zo základných technológií v čele správy dátového centra, ktorá pomáha podnikom riešiť ich problémy s rozvojom, bezpečnosťou, riadením IT infraštruktúry a zároveň znižuje finančné náklady podniku. (RULE, a iní, 2007 s. 60)

Virtualizácia je framework alebo metodológia delenia zdrojov počítača na niekoľko výpočtových prostredí aplikovaním jedného alebo viacerých prístupov alebo technológií, hlavne rozdelenia hardvéru a softvéru, zdieľania výpočtového času, čiastočné alebo úplné simulovanie výpočtového prostredia, emulácie, kvality služieb a iných.

Prečo virtualizovať:

- Úspora prevádzkových nákladov zlepšením energetickej efektívnosti, zredukovaním počtu fyzických počítačov a zvýšením pomeru serverov na správcu.
- Oddelenie systému od fyzického hardware-u prináša možnosť jednoducho preniesť systém na iné fyzické zariadenie bez nutnosti akýchkoľvek zmien v jeho konfigurácii.
- Živá migrácia umožňuje “za jazdy” preniesť bežiaci systém na iný fyzický systém bez výpadku prevádzkovaných služieb.
- Rýchle nasadenie systémov - nasadenie a spustenie nových systémov je vo virtualizovanom prostredí mnohonásobne rýchlejšie a jednoduchšie.
- Prenositeľnosť dát - diskové dáta virtualizovaných systémov je možné jednoducho prenášať medzi rôznymi fyzickými systémami, resp. virtualizačnými platformami.
- Transparentné sieťové služby - virtualizácia umožňuje za určitých okolností prenášať systémy medzi fyzickými systémami bez nutnosti rekonfigurácie sieťových nastavení.

Úrovne virtualizácie

Plná virtualizácia

Virtualizácia založená na tomto prístupe emuluje všetky dostupné fyzické zdroje hostitelského systému. Hypervisor, ktorý spostredkúva virtualizáciu prekladá všetky systémové volania v klientovi, ktoré žiadajú prístup k hardvéru (v tomto prípade virtuálnemu) hostujúcemu systému – tvorí akýsi most, prostredník medzi fyzickým a virtuálnym prostredím.

Paravirtualizácia

Paravirtualizácia sa vyznačuje tým, že vykonáva len čiastočnú abstrakciu na úrovni virtuálneho počítača, tj. ponúka virtuálne prostredie, ktoré je podobné tomu fyzickému, na ktorom virtuálny počítač provozujeme. Virtualizácia v tomto prípade nie je úplná, niektoré vlastnosti napr. procesoru môžu byť obmedzené a operačný systém môže rozpoznať, že beží vo virtuálnom prostredí. Na druhú stranu skutočnost, že virtuálny a fyzický hardware se príliš nelíši, umožňuje, aby virtuálny počítač v maximálnej miere využíval vlastnosti základného fyzického prostredia (nemusia sa emulovať všetky komponenty virtuálneho počítača).

Kontajnerová virtualizácia

Tiež nazývaná ako virtualizácia na úrovni operačného systému. Táto technika je často využívaná v hostingovými firmami, ktoré poskytujú virtuálne privátne servery kvôli jej "ľahkej" podobe – nie je nutné doinštalovať žiaden dodatočný softvér. Virtuálny počítač je tvorený izolovaným výpočtovým prostredím, ktoré beží v užívateľskom priestore a zdieľa fyzické prostriedky hostujúceho systému.

Virtualizácia aplikácií

Jedná sa o "najľahšiu" formu virtualizácie, nakoľko nevyžaduje virtualizovanie cieľového operačného systému, pre ktorý bola daná aplikácia vytvorená. Virtualizácia aplikácií využíva abstrakciu cieľovej platformy. Tento spôsob je podobný plnej virtualizácii, ktorá emuluje

hardvér. Aplikačná virtualizácia emuluje potrebné softvérové vybavenie, knižnice, prostriedky a systémové volania.

Prístup k virtuálnemu stroju

Priamo k operačnému systému bežiacemu na virtuálnom počítači sa dá pripojit štandardnými prostriedkami, SSH, VNC, atď. Na grafický výstup virtuálneho počítača sa dá pripojiť buď cez natívny protokol virtualizačného software-u (ako napr. vo VirtualBoxe) alebo cez nejaký remote desktop protokol, zpravidla VNC (napr. v Qemu), alebo RDesktop.

Virtualizačné riešenia

VMware

VMware je vývojárom virtualizačného software-u pre počítače založené na architektúre x86, ako je VMware Workstation alebo VMware ESX Server.

Hypervisor VMware ESX sa radí do rodiny natívnych hypervisorov, ktorý beží priamo na hardvéri fyzického serveru. VMware ESX poskytuje abstrakciu hardvérových prostriedkov a ich zdieľanie medzi virtuálnymi počítačmi. Z hľadiska implementácie ho možno taktiež zaradiť do skupín plnej virtualizácie, ale aj do množiny paravirtualizácie.

VMware Workstation je virtualizačný program (software), ktorý umožňuje spustiť na jednom počítači viac virtuálnych strojov. Každý takýto stroj používa svoj vlastný operačný systém. Nie je pritom obmedzení operačným systémom hostitelského počítača, takže môžeme spustiť na 32-bitovom operačnom systéme systém 64-bitový.

Citrix

Citrix Systems je americká firma, ktorá je zameraná na software predovšetkým určený na virtualizáciu a vzdialený prístup k aplikáciám.

Citrix XenServer je riešenie pre vykonávanie úloh virtualizovaných aplikacií na ľubovolnom počte serverov. XenServer umožňuje ovládať virtuálne ako aj fyzické servery a zvyšuje tak dynamiku celého dátového centra.

KVM

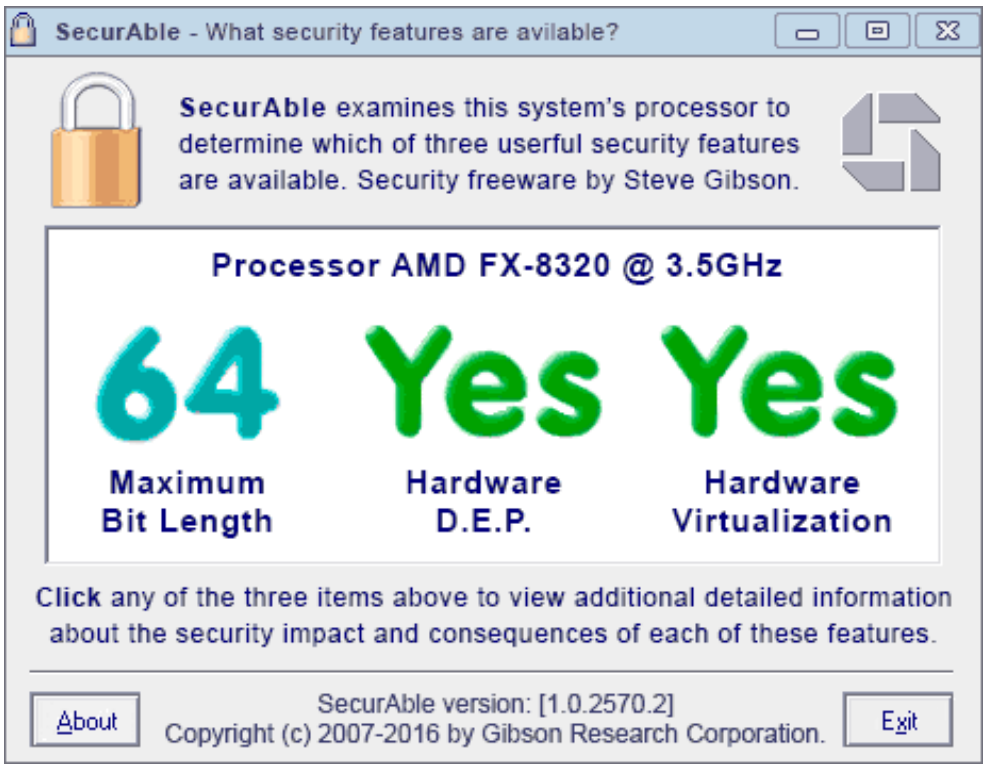
KVM (Kernel-based Virtual Machine) je modul jadra pre plnú virtualizáciu. V súčasnosti podporuje okrem x86-kompatibilných architektúr aj PowerPC, AMD64 a ARM. Jednou z funkcií je možnost živej migrácie, kedy je možné virtuálny systém preniesť na iný fyzický počítač bez nutnosti vypínať virtuálny stroj. Taktiež je možné prevádzkovať viac virtuálnych strojov. Každý z virtuálnych strojov má svoj virtualizovaný hardware: sieťovú kartu, disk, grafickú kartu, atď.

Príklad virtualizácie systému BIOS

Ako skontrolovať - je podporovaná a či je povolená virtualizácia

Pre tých, ktorí sa boja prihlásiť do systému BIOS, overte, či procesor podporuje technológiu virtualizácie alebo nie, a či je v BIOS-e aktivovaný, je to možné aj v programe SecurAble. Nástroj

je bezplatný, nevyžaduje inštaláciu. Program si môžete stiahnuť kliknutím na oficiálna stránka SecurAble



Obrázok 6 SecurAble (https://www.grc.com/securable.htm)

Možnosti zabezpečenia:

- Hodnota parametra Maximálna dĺžka bitov označuje maximálnu možnú bitovú rýchlosť 32-bitového alebo 64-bitového systému.
- Hodnoty Hardvér D.E.P - Technológia zodpovedná za bezpečnosť sa implementuje, aby zabránila spusteniu škodlivého kódu.

- Možnosť Virtualizácia hardvéru - parameter môže poskytnúť štyri hodnoty:

áno - virtualizačná technológia podporovaná procesorom - povolená;

žiadny - procesor nepodporuje virtualizáciu;

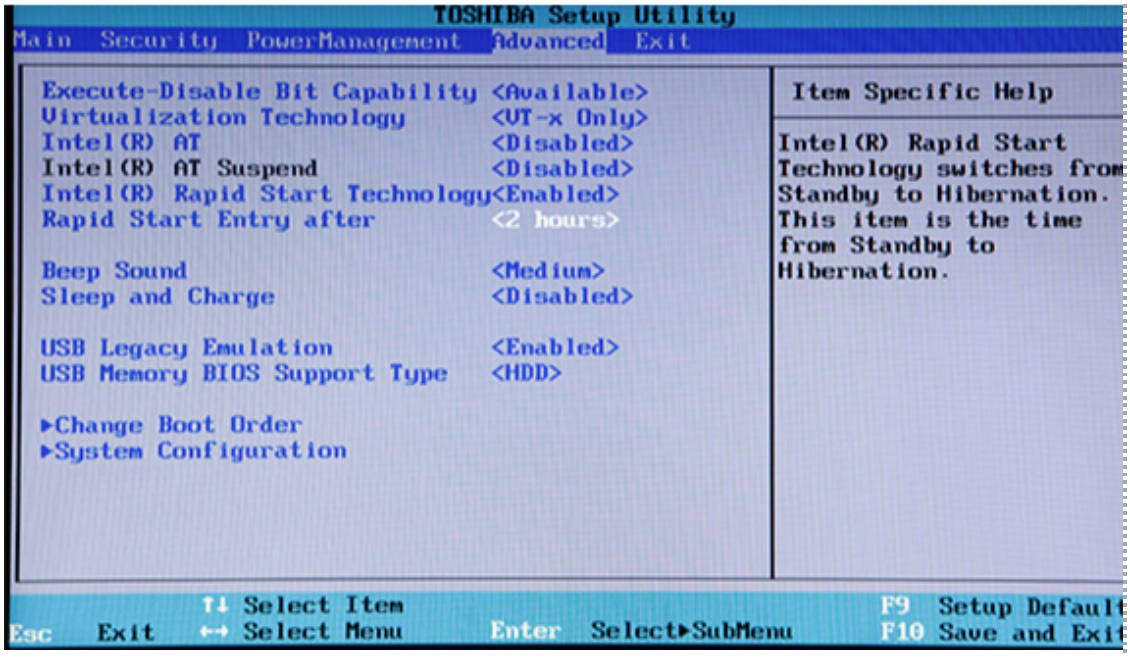
Zamknuté - povolené a podporované, ale nemôžu byť vypnuté v systéme BIOS;

Uzamknuté vypnuté - Podpora technológie je k dispozícii, ale je vypnutá a nie je možné ju zapnúť v systéme BIOS.

Inscription Locked Off nie je vždy verdikt - bIOS reštartovať môže situáciu napraviť.

Technológia virtualizácie je zodpovedná za zapnutie virtualizácie hardvéru v systéme BIOS. Ak chcete vypnúť možnosť alebo zapnúť virtualizáciu v systéme BIOS, pošleme počítač na reštart. Keď sa objavia prvé príznaky bootovania, kliknite na klávesovú skratku "F2" alebo "Delete" (rôzne verzie systému BIOS) a vyhľadajte nápovedu v spodnej časti obrazovky na začiatku.

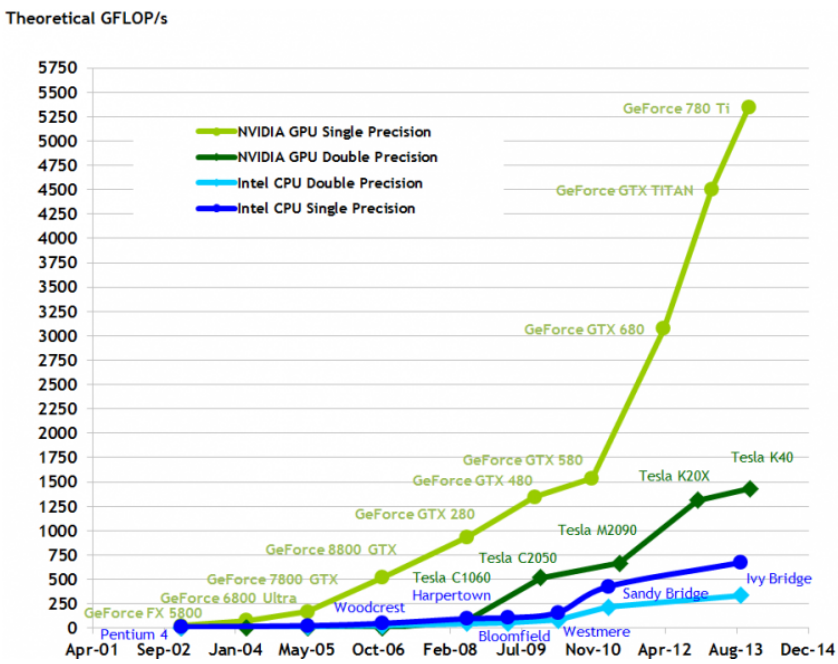
Prejdite do sekcie "Advanced BIOS - Funkcie", nachádzame možnosť "Virtualizácia" alebo "Rozšírené" → "Konfigurácia CPU", možnosť "Intel Virtualization Technology".



Obrázok 7 BIOS a virtualizácia

2.4 Všeobecné výpočty na GPU

Grafické procesory sa na rozdiel od klasických procesorov, zameraných na rýchlosť operácií, orientujú na výkon (veľké množstvo jednoduchých operácií). Ako je vidieť z obrázku 8 výkon grafických procesorov rastie rýchlejšie ako u CPU a už je niekoľkokrát väčšia. Výkon je meraný v miliónoch operácií s pohyblivou radovou čiarkou za sekundu.



Obrázok 8 Porovnanie výkonu GPU a CPU

Ďalším negatívom CPU je jeho prílišná zložitosť (urýchlenie behu jedného vlákna). Vďaka nízkej pamäťovej priepustnosti musí byť na CPU obsiahnutá tiež veľká cache-pamäť. Grafické procesory naopak obsahujú viac integrovaných pamäťových radičov na jedno jadro a toto zaručuje vysokú pamäťovú priepustnosť. Nové procesory obsahujú teraz oveľa viac jadier. Grafické procesory obsahujú stovky jadier na ktorých môže bežať až 12000 súčasne. GPU v porovnaní s procesormi teda excelujú pri nasadeniach, ktoré možno dobre paralelizovat - typicky sú to okrem niektorých vedeckých výpočtov práve časovo veľmi náročné a pritom výpočtovo "jednoduché" operácie ako rendering alebo enkódovánie videá, či manipulácie a práce s veľkými obrázkami.

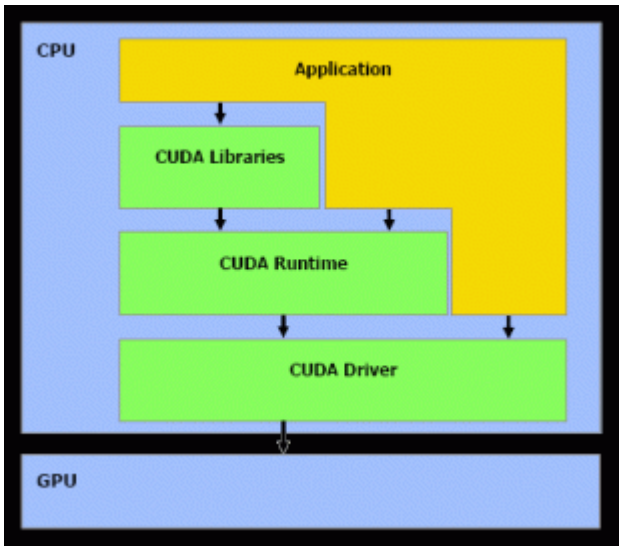
Obrázok 9 Štruktúra jadier GPU a CP

GPGPU, CUDA

Vysoký výkon grafických procesorov podnietil vznik nových odvetví - GPGPU, ktoré sa snažia využívať tento výkon nielen na grafické účely. Spoločnosť NVIDIA vytvorila programovacie prostredie CUDA (Compute Unified Device Architecture), ktoré by malo pomôcť GPU výpočtom. CUDA umožňuje programovať negrafické aplikácie pre grafické procesory pomocou jazyka C.

Spoločnosť ATI / AMD tiež vytvorila svoju technológiu pre paralelné využitie GPU - ATI Stream. CUDA je najstaršou zverejnenou technológiou pre paralelné výpočty na GPU a možno s ňou pracovať na všetkých grafických akcelerátoroch série G80 a vyššie. CUDA môže byť tiež emulovaný na klasických procesoroch.

CUDA aplikácia môže pristupovať k akejkoľvek časti API CUDA. Či už sa jedná o knižnice CUBLAS (Basic Linear Algebra Subprograms) alebo CUFFT (Fast Fourier Transform), tak i runtime API alebo driver API.



Obrázok 10 Prístup k programu API CUDA

GPGPU možno účinne využívať hlavne v aplikáciách, ktoré možno veľmi dobre paralelizovať. Tiež sa hlavne využíva k časovo náročným, ale jednoduchým operáciám (rendering, spracovanie veľkých obrázkov, práca s HD filmy).

U vhodných algoritmov po prepise programu pre GPU dôjde k výraznému zrýchleniu. GPGPU možno využiť napríklad aj v automobilovom priemysle pre návrh automobilov za pomocou GPU raytracing pod platformou CUDA. Alebo tiež v zdravotníctve napríklad pre efektívnejšie nachádzanie prvotných štádií rakoviny. National Cancer Inštitút (inštitút pre výskum rakoviny) takto zrýchlil 12x algoritmus pre výpočet proteínov. Ďalšie možné využitie je napríklad vyhľadávanie tvárí v obrázkoch.

3 UCHOVÁVANIE, ZÁLOHOVANIE A ARCHIVÁCIA DÁT

Myšlienka zapisovania a uchovávania písaných záznamov súvisí so vznikom symbolov potrebných na zápis - vznikom písma.

Uchovanie informácie súvisí rozvojom komunikácie medzi ľuďmi. Aby sa informácie mohli šíriť aj na väčšie vzdialenosti, bolo treba nájsť iný spôsob, ako ústne podanie a uchovanie v pamäti človeka(https://melisko.webnode.sk/).

S rozvojom fyziky sa na uchovávanie informácií začal využívať celý rad fyzikálnych zákonitostí:

- elektro-mechanický záznam (dierne štítky, dierne pásky, vinylové platne),
- magnetické pole a magnetizácia látok (pevné disky, diskety),
- statické elektrické pole (dynamické pamäte),
- optika - vlastnosti monochromatického svetla (lasera) v zápise na kompaktné disky, DVD disky, Blu-ray disky, hologramy atď.

Údaje sú v technických zariadeniach uchovávané ako postupnosti **núl** a **jednotiek**. Takýto spôsob zápisu, ktorý používa na zápis informácií nuly a jednotky sa nazýva **binárny kód.**

3.1 Reprezentácia čísiel v počítači

Kódovanie informácií

Cieľom kódovania je, aby sme mohli prenášať informáciu. Často sa musíme prispôsobiť možnostiam technického zariadenia (preto vznikla napr. Morseova abeceda ako jazyk pre telegraf), alebo možnostiam ľudí zapojených do komunikácie (preto vzniklo Braillovo písmo pre nevidiacich).

VÝKLAD

Kódovanie informácii je ľubovoľná vopred dohodnutá a všeobecne známa množina pravidiel, ktorá dovoľuje informáciu vyjadriť tak, aby sa dala uchovať, alebo šíriť(https://melisko.webnode.sk/).

Šifrovanie sa používa všade tam, kde treba utajiť obsah komunikácie. Existuje veľmi veľa metód na tajné šifrovanie (a metód na dešifrovanie).

Binárny kód

Ľudia medzi sebou používajú na komunikáciu reč, ktorá pozostáva zo slov tvorených jednotlivými písmenami. Počítače komunikujú – prenášajú informácie - v číslach. Informácie, ktoré v nich človek uchováva, musia byť prekódované do im zrozumiteľného jazyka.

Pamäť počítača si môžeme predstaviť ako milióny miniatúrnych prepínačov, z ktorých každý je buď rozopnutý (pre znak 0) alebo zopnutý (pre znak 1). Dlhé postupnosti prepínačov predstavujú rôzne informácie.

Počítače preto používajú zvláštny spôsob kódovania informácii – ***binárny kód. Sú to postupnosti dvoch znakov – 0 a 1.***

Informácie zapísané v binárnom kóde nazývame ***digitálne informácie.*** ***Digitalizácia*** - prevod informácie z reálneho sveta (blízke človeku) do binárneho kódu podľa dohodnutých pravidiel.

Reprezentácia čísel v počítači

Všetky údaje v počítači sú kódované pomocou rôznej kombinácie hodnôt bitov. Každý z bitov môže nadobúdať iba dve rôzne hodnoty 0 a 1.
Tieto bity sú však do pamäťových buniek počítača ukladané po osmiciach, preto je výhodné na zakódovanie údajov použiť vždy taký počet bitov, ktorý je deliteľný ôsmimi (8,16,24,32 ....).

Čím väčší počet bitov použijeme, tým väčší rozsah čísel môžeme použiť.Napríklad pomocou 8 bitov dostaneme 28 = 256 rôznych kombinácií núl a jednotiek. Pomocou 8 bitov teda môžeme zakódovať napríklad čísla od ***0 do 255*** alebo čísla od ***-128 do 127*** v prípade, že potrebujeme i záporné čísla.

Na kódovanie čísiel v počítačoch je najvýhodnejšie použiť jedno ***„slovo“*** (Word), t.j. taký počet bitov, ktoré počítač dokáže spracovať počas jednej operácie (jedného taktu procesora). Najmodernejšie počítače dnes používajú 64 bitové slovo, teda dokážu spracovať 64 bitov pri jednej operácií.

Kódovanie prirodzených čísel a nuly

Každé číslo môžeme previesť do *dvojkovej sústavy,* ktorá používa iba cifry 0 a 1, čím pre každé číslo dostaneme jednoznačný zápis.

Prirodzené čísla sú v počítači uložené v tzv. ***priamom kóde***, čo je vlastne číslo prevedené do dvojkovej sústavy. Jedným z takýchto kódov je *kód BCD*.

Kódovanie celých čísel

Pri celých číslach je potrebné zohľadniť i znamienko. Našťastie sú znamienka len dve (+, -), preto ich môžeme zakódovať 1 bitom (0 = +, 1= -).

Pri kódovaní celých čísel sa znamienko zakóduje vždy prvým bitom zľava.

Binárna (dvojková) číselná sústava

Bežne v živote používame čísla vyjadrené v ***desiatkovej pozičnej sústave***, ktorá :

- používa cifry od nuly do deväť – teda 0, 1, 2, ..., 9,
- využívaj pozičný spôsob zápisu. Skutočnú hodnotu každej cifry v čísle určuje to, či stojí na pozícii jednotiek alebo desiatok alebo stoviek atď. Tieto kľúčové čísla sa nazývajú pozičné hodnoty.

Napríklad číslo *6 521* je súčtom 6000 + 500 + 20 + 1 teda ***6*10³+5*10²+2*10¹+1*10⁰***

Dvojková pozičná sústava :

- používa cifry od nula do jeden – teda iba 0 a 1,
- používa pozičný spôsob zápisu, ale pozičnými hodnotami sú mocniny čísla 2,

Napríklad číslo *101101* je súčtom ***1*2⁵+0*2⁴+1*2³+1*2²+0*2¹+1*2⁰***

Prevod čísel z desiatkovej do dvojkovej sústavy

Prevod desiatkového čísla na dvojkové vykonáme postupným celočíselným delením čísla dvomi a zapisovaním zvyšku po celočíselnom delení do výsledného dvojkového čísla.

Príklad

$327_{(10)}$ = $?_{(2)}$

327:2= 163	zvyšok 1
163:2= 81	zvyšok 0
81:2 =40	zvyšok 1
40:2 =20	zvyšok 0
20:2 =10	zvyšok 0
10:2 =5	zvyšok 0
5:2 =2	zvyšok 1
2:2 =1	zvyšok 0
1:2 =0	zvyšok 1

Zvyšky potom zapíšeme do dvojkového čísla tak, že prvý zvyšok bude na prvej pozícii zprava

Výsledok: *327 (10)* = ***101000111 (2)***

Prevod čísel z dvojkovej do desiatkovej sústavy

Postup: postupne, zprava doľava, sčitujeme čísla (0 a 1) v dvojkovej sústave vynásobené mocniteľom čísla 2 umocneného od 0 až po n.

Príklad:

$1\,0\,1\,0\,0\,1\,1_{(2)}$ = $?_{(10)}$

$1*2^6+0*2^5+1*2^4+0*2^3+0*2^2+1*2^1+1*2^0 = 83$

Výsledok: $1\,0\,1\,0\,0\,1\,1_{(2)}$ = $83_{(10)}$

Reprezentácia textovej informácie

Keď chceme pracovať s textovou informáciu v počítači musíme ju najprv nejakým spôsobom vedieť v počítači reprezentovať. Veľmi výhodnou možnosťou je rozložiť textovú informáciu na jednotlivé znaky (https://melisko.webnode.sk/).

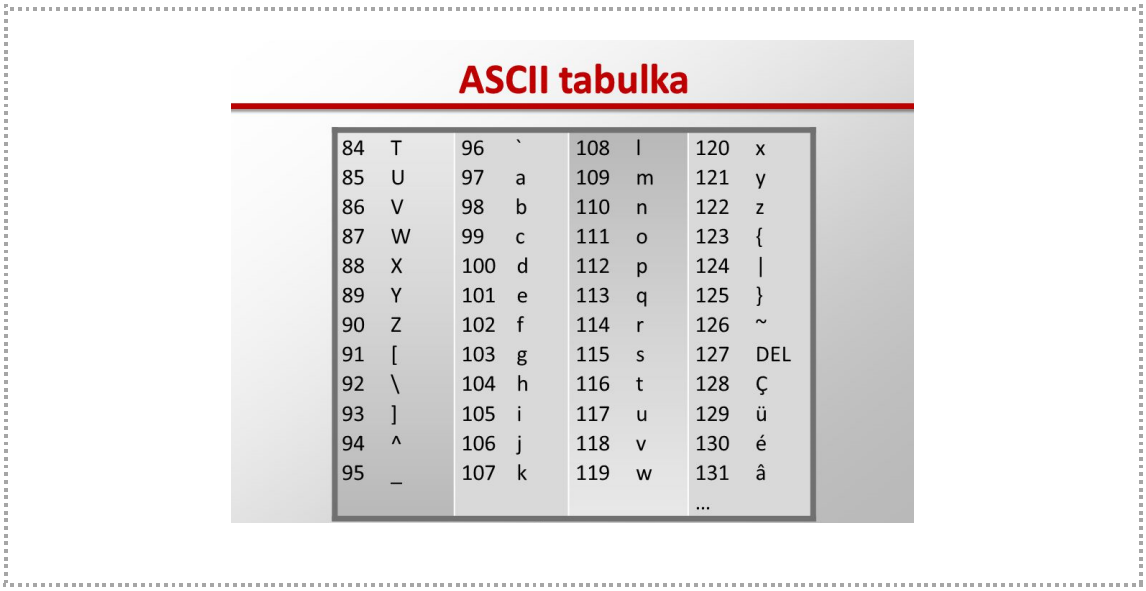
Znakom nie sú len malé a veľké písmená, ale aj číslice, špeciálne znaky – čiarka, bodka, plus, pomlčka, dvojbodka, matematické operácie plus, minus aj takzvané neviditeľné znaky. To sú napríklad medzera, tabulátor, koniec riadku, koniec súboru a iné.

Keď máme text rozložený na jednotlivé znaky, zistíme, že týchto znakov nie je nejako veľa. Sú len rôzne usporiadané a tým vytvárajú text. Teraz každému znaku priradíme nejaký kód a následne v súbore celý text zapíšeme ako postupnosť kódov znakov textu.

Takéto priradenie kódov znakom nazývame *kódová tabuľka*. V minulosti boli znaky kódované pomocou 7 bitov. To umožnilo kódovať 128 znakov. ***Takéto kódovanie označujeme ASCII*.**

Toto kódovanie celkom stačilo pre anglickú abecedu a množinu základných symbolov. Ale aj ostatné národy chcú písať svoje texty cez diakritiku a preto sa začalo používať na kód znaku 8 bitov = 1 byte. Teda dalo sa zapísať 256 rôznych znakov. Prvých 128 znakov bolo spoločných, zodpovedali pôvodnému ASCII kódovaniu. Zvyšných 128 znakov si každé kódovanie vybralo podľa svojich potrieb. Vznikla *rozšírená ASCII tabuľka*.

Spočiatku bol takýto systém výhodný. Umožňoval písať texty v národných abecedách s diakritikou. Ale neskôr s rozvojom počítačových sietí dochádzalo čoraz častejšie k problémom. Každé kódovanie malo znaky s kódmi 129 až 255 rôzne. To spôsobilo, že ak sme napísali text v programe s nastaveným kódovaním napríklad pre strednú Európu a poslali sme ho niekomu, kto používal kódovanie pre západnú Európu, tak znaky anglickej abecedy sa zobrazili korektne, ale znaky s diakritikou a rôzne špeciálne znaky sa mohli zobraziť úplne nezmyselne.



Obrázok 11 Acsi tabuľka

Pre Strednú Európu vzniklo viacero kódovaní: windows-1250, ISO-8859-2, MacCE, IBM-852 a pod. Aby sa predišlo podobným problémom bolo vytvorené kódovanie s použitím 16 bitov s

názvom *UNICODE*. V tomto kódovaní je možné zakódovať cez 65536 rôznych znakov, čo umožňuje zakódovať znaky všetkých abecied pomocou jednej medzinárodnej tabuľky. Toto kódovanie zabezpečuje, že ten istý znak má rovnaký kód v každej krajine i a každom type počítača.

Nevýhodou tohto kódovania je, že znaky, ktoré sme predtým vedeli zakódovať iba ôsmymi bitmi v kódovaní Unicode, sú kódované 16 bitmi, a teda zaberajú viac pamäte ako kód ASCII.

Istým vylepšením tohto kódovania je kódovanie ***UTF-8***. V tomto kódovaní je prvých 128 znakov tabuľky ASCII (tieto sú pre všetky krajiny rovnaké), zakódovaných pomocou 8 bitov a zvyšné znaky sú zakódované 16, 24, 32, 40 až 48 bitmi.

Reprezentácia grafickej informácie

Rastrová grafika

Je to spôsob uloženia grafických údajov ***pomocou farieb jednotlivých bodov obrazu***. Jeho podstatou je rozloženie obrazu na jednotlivé zobraziteľné body – *pixely*. Počet pixelov pomocou ktorých je obrázok uložený sa údáva v pixeloch na výšku a šírku. Napríklad 1024x768, hovoríme tomu ***rozlíšenie obrázka***, alebo tiež raster. Teda súbor obsahujúci údaje o rastrovej grafike obsahuje informácie o farbe jednotlivých pixelov. Samozrejme farba je zakódovaná binárnym kódom do postupnosti núl a jednotiek(https://melisko.webnode.sk/).

Vektorová grafika

Pri použití vektorovej grafiky nepracujeme z jednotlivými pixelmi obrázku, ale so základnými objektmi (úsečky, kružnice, oblúky, mnohouholníky, transformácie, farby), ktoré umožňujú vytvárať výsledný obrázok. V súbore vektorovej grafiky sú uložené príkazy, ktoré určujú ako sa výsledný obrázok poskladá zo základných tvarov a ich úpravami - zmenou tvaru, transformáciou, zmenou farby a inými.

Výhodou oproti bitmapovej grafike je možnosť zväčšovať a zmenšovať obrázok bez zmeny kvality.

Video

Videozáznam je v podstate sekvencia statických obrázkov, ktoré sa v rýchlom slede zobrazujú. Teda môžeme na ich uloženie použiť nejaký formát na uloženie statických obrázkov, ktorý doplníme o nejaké pokyny na zabezpečenie striedania obrázkov (ako rýchlo sa majú obrázky striedať a pod.).

Ak by sme použili nekomprimované statické obrázky štandardnej veľkosti, napríklad 1024x768 pixelov, pre plynulý pohyb by sa malo za sekundu vystriedať asi 30 obrázkov. Pri 24 bitovej farebnej hĺbke (3Byty) by to znamenalo, že na sekundu záznamu by sme potrebovali: 30x1024x768x3 Bytov = 70778880 B = 67,5 MiB. Ak film má hodinu, čo je 3600 sekúnd, tak výsledný súbor by mal 243000 MiB = 237,3 GiB. Čo je veľkosť takmer celého pevného disku.

Ako zmenšiť veľkosť videa? Je viacero metód:

- Použijeme menšie rozlíšenie obrázkov, napr.: 320x240.
- Nepoužijem 30 snímkov za sekundu, ale menej, napr. 12, obraz je potom trochu sekaný, ale nemusí to príliš prekážať.
- Použijem komprimovaný formát jednotlivých snímok - potrebujem dostatočne výkonný počítač, aby stíhal obrázky rýchlo načítavať.
- Celý sa uloží len prvý obrázok sekvencie, z ďalších obrázkov sa uložia len tie časti, ktoré sa oproti prvému zmenia.
- Použije sa lepšia kompresia celého videa (nie jednotlivých snímok, ale videa ako celku).

Reprezentácia zvukovej informácie

Zvuk je v podstate vlnenie, ktorého hlavnými charakteristikami sú *frekvencia* ***a*** *hlasitosť*. Ak chceme v počítači ukladať zvuk musíme ukladať ***frekvenciu - výšku tónu***. Podľa toho koľko bitov použijeme na jeden tón máme výsledný počet rôznych tónov. Pomocou 8 bitov môžeme rozlíšiť cez 256 tónov. Použitím ďalších bitov sa počet tónov zvyšuje. Pre rozlíšenie štandardných kvalít záznamu platí:

(počet bitov na vzorku, vzorkovacia frekvencia, počet kanálov)

- 8 bitov, 11kHz, mono – telefónna kvalita záznamu.
- 8 bitov, 22kHz, mono – rozhlasová kvalita.
- 16 bitov, 44kHz, stereo – CD kvalita.
- 24 bitov, 192kHz, 5+1 – DVD kvalita.

Pri ukladaní stereo zvuku potrebujeme uložiť samostatne každý kanál. To znamená, že potrebný počet bitov sa nám zdvojnásobí.

Digitalizácia informácie

Zjednodušene môžeme povedať, že svet okolo nás je plný analógových informácií. Keď ich chceme spracovať v počítači, musíme ich previesť na digitálne. Tento proces sa nazýva ***digitalizácia***.

Pôvodná analógová informácia musí byť rekonštruovateľná z výslednej digitálnej informácie. Z povahy digitalizácie vyplýva, že určité podrobnosti informácie sa pri digitalizácii stratia.

Základom digitalizácie (prevod spojitého signálu na diskrétny) sú tri procesy: *vzorkovanie, kvantovanie, kódovanie*.

Pri ***vzorkovaní*** sa odoberajú hodnoty (vzorky) v pravidelných intervaloch. Hustota odoberania vzoriek sa nazýva *vzorkovacia frekvencia*.

Pri *kvantovaní* ide o rozdelenie celého rozsahu hodnôt, ktoré môže nadobúdať analógová informácia na intervaly. Každý interval má potom priradenú svoju zástupnú hodnotu (kvantovanie podľa hodnoty).

Kvantovaním zaradíme hodnoty analógovej informácie namerané vzorkovaním do príslušných intervalov.

Po zaradení do intervalov nastáva tretia fáza spracovania - **kódovanie**. Pri kódovaní sa uložia navzorkované a nakvantované hodnoty vo formáte vhodnom pre digitálnu techniku. V podobe slov zložených z núl a jednotiek - zakódujeme ich do binárnej podoby.

3.2 Zálohovanie a archivácia dát

V profesionálnej praxi sú údaje spravidla to najcennejšie na celej počítačovej zostave. Dáta je nutné chrániť pred zničením, poškodením či zneužitím. Zabezpečeniu dát na počítači je potrebné venovať dostatočnú pozornosť, pretože tam je výsledok často niekoľkoročnej práce používateľa. Software sa dá z inštalačných diskov obnoviť v priebehu niekoľkých hodín, cena jednotlivých poškodených komponentov HW počítača nepresahuje spravidla desať tisíc korún, cena celej HW zostavy predstavuje niekoľko desiatok tisíc. Hodnota dát na profesionálne používanom zariadení je spravidla mnohonásobne vyššia. Dáta môžu byť zničené chybou obsluhy, chybou SW, haváriou HW, môžu byť spolu s počítačom zničené živelnou udalosťou, môžu byť aj zničené, poškodené, pozmenené v cudzí prospech, ukradnuté a zneužité nepovolanými osobami v dôsledku hackerského útoku – neoprávneného prieniku do počítača alebo celého počítačového systému. Osobitne sa budeme zaoberať rizikom straty dát v dôsledku pôsobenia vírusu. Zabezpečenie dát pred zneužitím nepovolanými osobami a ochrana počítačových systémov pred hackerskými prienikmi presahuje rámec tejto témy. Je však potrebné mať na pamäti, že oddelovať pôsobenie vírusov a iných nedokumentovaných programov od pôsobenia hackerov nie je možné, pretože tieto programy sú dielom hackerov a mnohé majú za úlohu pripraviť pôdu na napadnutej stanici pre ďalšie prieniky po sieti. Zabezpečenie dát pred stratou (zničením) z ľubovolného dôvodu vykonávame zálohovaním. Zálohovaniu a archivácii dát sa venuje samostatná kapitola, teraz si však pripomeňme, že účinné zálohovanie musí byť: F pravidelné F dôsledné F dáta musia byť uložené fyzicky mimo pracovného počítača, pokiaľ možno aj v inej miestnosti Pravidelné a dôsledné zálohovanie je prvým predpokladom minimalizácie škôd aj pri napadnutí vírusom. Pre prípad straty dát v dôsledku zničenia či poškodenia pracovných súborov s dátami vytvárame záložné kópie súborov. Neaktuálne dáta, ku ktorým nepotrebujeme pravidelný prístup, ale ktoré je potrebné uchovať vzhľadom na možnú potrebu použiť ich v budúcnosti, archivujeme. Pre efektívne uchovávanie záložných súborov a archívov používame komprimáciu súborov (http://pk-info.spsepn.edu.sk/).

VÝKLAD

Zálohovanie ako základný prostriedok ochrany údajov

Princípom zálohovania je pravidelné ukladanie záložných kópií pracovných súborov. V prípade zničenia pracovného súboru obnovíme dáta zo záložného súboru. Zmeny vykonané v pracovnom súbore od posledného zálohovania sú ovšem nenávratne stratené – túto časť práce je potrebné vykonať znova. V rôznych podmienkach a pre rôzne typy dát volíme rôzne metódy zálohovania(http://pk-info.spsepn.edu.sk/).

Metódy zálohovania podľa organizácie vytvárania záložných kópií:

- Jednoduché – z orginálu sa vytvára jediná záložná kópia používa sa na rýchle zálohovanie bežných pracovných súborov nepríliš veľkej dôležitosti.
- Viacnásobné – z orginálu sa vytvára viacero záložných kópií, uložených na rôznych médiách používa sa na zálohovanie dôležitých dát.
- Viacstupňové – z orginálu sa vytvorí záložná kópia, z pôvodnej zálohy sa vytvorí záloha druhého stupňa atď. – vzniká kaskáda záložných súborov, ktoré obsahujú projekty v rôznej fáze rozpracovanosti používa sa na zálohovanie dôležitých projektov, kde je potrebné mať odložené rôzne etapy realizácie.

Podľa spôsobu ukladania záložnej kópie

- Záloha vytvorená priamo na tom istom médiu najbežnejší spôsob rýchleho zálohovania v priebehu práce rýchle, pohotové, ale neochráni v prípade zničenia nosiča dát mnohé programy umožňujú automatické vytváranie záložných súborov typu .bak vždy pri uložení novej verzie súboru.
- Zálohovanie na záložnom serveri pravidelné zálohovanie väčších objemov dát rýchle, dobre chráni pred zničením, horšie chráni pred zneužitím (neoprávnené prieniky po sieti) automatické zálohovanie obstaráva špeciálny program (často býva súčasťou operačného systému).
- Zálohovanie na vonkajšie pamäťové médiá (diskety, výmenný HD, CD, CD RW, JAZ, pásky) dôkladné zálohovanie dôležitých údajov po ukončení väčších etáp práce najspoľahlivejšie, umožňuje veľmi bezpečné uloženie dát (napr. do trezoru), ale najmenej pohotové zálohovanie musí vykonať užívateľ, automatické zálohovanie na výmenné médiá podporujú iba špecializované prostriedky (pásové mechaniky).

Archivovanie

Dáta, ktoré nie je potrebné ďalej udržiavať na pracovnom disku, ale zároveň je potrebné ich odložiť pre neskoršie použitie alebo archívne účely (dokončené publikácie, projekty, uzatvorené účtovníctvo za určité obdobie a pod.), archivujeme spravidla na bezpečnom mieste na vonkajších pamäťových médiách (diskety, CD, pásky).

Zálohovať a archivovať súbory je možné buď jednoduchým kopírovaním, alebo s použitím špeciálnych zálohovacích utilít (napr. BACKUP, FASTBACK, automatické zálohovanie vo Windows a pod, .), alebo s využitím komprimácie. Ukladanie záložných kópií a archívov je náročné na kapacitu médií. Preto sa používa komprimácia (nazýva sa tiež kompresia, balenie, pakovanie, zipovanie...) Princípom komprimácie je prekódovanie súboru tak, aby sa zmenšil objem uchovávaných dát bez straty informácie. (Veľmi zjednodušene: Namiesto informácie „5,5,5,5,5,5,2,2,2“ zapíše „6x5, 3x2“) V skutočnosti ide o veľmi zložité matematické algoritmy, ktoré v súbore vyhľadávajú možnosti najefektívnejšieho zakódovania dát. Dáta v komprimovanom súbore zaberajú menej pamäťovej kapacity, je ľahšie ich uskladniť a prenášať, ale nie sú prístupné na spracovanie. Aby bolo možné komprimované dáta normálne používať, je nutné ich opäť dekomprimovať (rozbaliť). Dôležitým parametrom komprimácie je kompresný pomer: Udáva pomer medzi objemom dát v zkomprimovanom a nezkomprimovanom tvare, vyjadrený v percentách. Hodnota kompresného pomeru závisí jednak od účinnosti použitej kompresnej metódy, jednak od typu komprimovaných dát. Dosahovaný kompresný pomer je

najdôležitejšou charakteristikou každého komprimačného programu. Komprimácia sa veľmi často používa aj na účely prenosu veľkých súborov pomocou diskiet. V takom prípade je však potrebné zabezpečiť, aby aj na cieľovom počítači bol nainštalovaný kompaktibilný komprimačný program (http://pk-info.spsepn.edu.sk/).

Komprimačné programy

Sú to programy, spravidla distribuované ako shareware, špecializované na komprimáciu. Je to napr. PKZIP, LHARC, ARJ, RAR,,WinZIP, WinRAR, a mnoho ďalších. Komprimáciu umožňujú aj niektoré súborové manažery, kompresné utility sú súčasťou aj mnohých iných programov. Komprimovaným súborom sú priradené koncovky podľa typu komprimačného programu (napr *.arj, *.rar, *.zip atď.) Všetky komprimačné programy však v zásade obsahujú nasledovné funkcie:

- NEW – vytvorenie nového archívneho súboru.
- ADD - pripojenie súborov do archívu, (niekedy vytvorenie nového archívu) MOVE - presun súborov do archívu.
- DELETE - vymazanie súborov z archívu.
- UPDATE, FRESH - aktualizácia archívu.
- EXTRACT - dekompresia (obnovenie súborov z archívu).
- VIEW – prezeranie obsahu súborov v archíve.
- TEST, CHECK - kontrola celistvosti archívu.

Postup pri vytváraní archívneho súboru. Pri vytváraní archívu, bez ohľadu na použitý program, musíme špecifikovať:

- Ako sa má volať archívny súbor a kde má byť uložený (cesta).
- Ktoré súbory z ktorých adresárov majú byť do archívu komprimované.
- Druh výstupného súboru .
- Kompresnú metódu.

ÚLOHA 3 – RIEŠTE!

Zkomprimujte pomocou programu WinRar adresár, ktorý chcete archivovať.

Komprimačný program môže vytvoriť niekoľko druhov výstupných súborov:

- Bežný archívny súbor - kapacita výstupného súboru nie je obmedzená Používa sa pri ukladaní archívu na HD.
- Samorozbaliteľný archívny súbor – výsledkom je spustiteľný (*.exe) súbor, ktorý sám vykoná dekompresiu do aktuálneho adesára.

Komprimačné programy umožňujú spravidla nastaviť kompresnú metódu.

Na výber je zvyčajne niekoľko možností:

- BEST (MAXIMUM) – najlepší kompresný pomer, ale najpomalšia.
- FAST (SPEED) – kompresia je najrýchlejšia, ale najmenej účinná.
- OPTIMUM – kompromis medzi kvalitou a rýchlosťou kompresie.

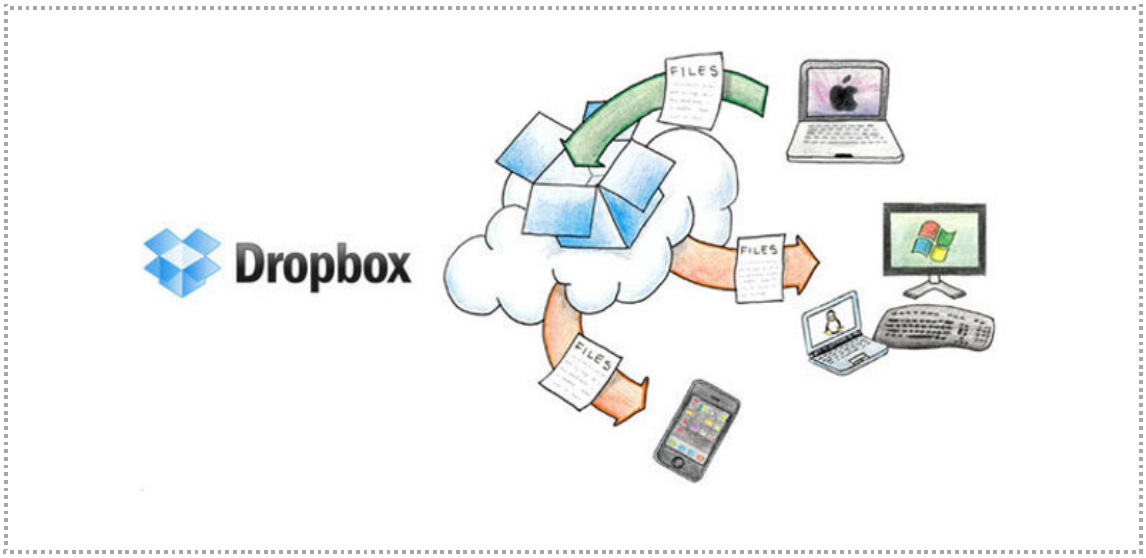
Po nastavení všetkých uvedených parametrov môže prebehnúť vlastná kompresia.Pri dekomprimácii musíme postupovať analogicky:

- Otvoriť archív pomocou kompresného programu .
- Vybrať súbory na dekomprimáciu .
- Určiť cielový adresár.
- Vykonať dekompresiu.

Prostredie programov sa líši podľa operačného systému, jednotlivé programy danej skupiny sú si však prostredím, príkazmi aj možnosťami veľmi podobné.

Technické prostriedky na zálohovanie dát v dnešnej dobe

1) Cloud: Dropbox



Obrázok 12 Dropbox (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)

Nebojte sa zveriť svoje dáta internetu. Nemyslíme tým ale, aby ste si všetky fotografie nahrali na Facebook a sprístupnili ich každému.

Cloud je čarovné slovo, ktoré vám umožní veľmi jednoducho zálohovať a pristupovať k súborom z prakticky akéhokoľvek zariadenia na celom svete, vyžaduje sa jedine prístup na internet. Dropbox sme vybrali kvôli jeho univerzálnosti, keďže má skvelé aplikácie na všetky rozšírené typy zariadení a spoľahlivo spolupracuje aj medzi rôznymi systémami. Je teda jedno, že váš

pracovný notebook má Windows, smartfón máte s Androidom a ako tablet ste práve dostali nový iPad.

Dropbox si po nainštalovaní v počítači vytvorí vlastný adresár, do ktorého si následne môžete ukladať všetky súbory. Po ich umiestnení alebo akejkoľvek zmene sa okamžite aktualizujú aj v cloude, pričom pri dokumentoch služba ukladá až niekoľko verzií daného súboru.

Ak sa teda počas práce pomýlite a chybu si všimnete až po niekoľkých dňoch, tak je stále možnosť návratu. Dropbox ponúka zadarmo kapacitu pre dva gigabajty dát, viac môžete získať napríklad kúpou nového smartfónu alebo odporučením služby priateľom. Pre väčšie objemy dát budete platiť desať eur mesačne, tu je maximálne kapacita obmedzená na jeden terabajt.

2) Profesionálny externý disk: WD My Book Pro

Obrázok 13 Externý disk (https://tech.sme.sk/c/20128709/pat-sposobov-ako-si-bezpecne-zalohovat-data.html#ixzz5GJYt9snn)

S kapacitou až 16 terabajtov, rýchlym pripojením Thunderbolt 2 a podporu pre automatické zálohovanie je WD My Book Pro riešením pre profesionálov, ktorí pracujú s veľkým objemom dát. V ich prípade totiž klasické pevné disky a cloud nepripadajú do úvahy, keďže by častým prenosom niekoľkých gigabajtov dát strácali veľa času.

WD My Book Pro môžete s počítačom alebo notebookom prepojiť cez rozhranie USB 3.0 alebo Thunderbolt 2, takže jediným limitom je v tomto prípade rýchlosť samotných diskov. Tie sú v zariadení dva, pričom ich môžete v prípade potreby vymeniť za nové.

Je len na vás, či si vyberiete vyššie zabezpečenie (a polovičnú kapacitu) v podobe zrkadlenia dát na obidvoch diskoch, alebo vysoký výkon pri zapojení RAID 0. V druhom prípade je rýchlosť

zápisu a čítania vyššia ako 400 megabajtov za sekundu, takže so zálohovanými dátami môžete pokojne pracovať aj v reálnom čase bez obmedzení. K WD My Book Pro si môžete pripojiť ďalšie dve USB 3.0 zariadenia, prípadne až dve Thunderbolt 4K obrazovky.

Podporovaná je aj funkcia Time Machine na automatické zálohovanie počítačov a notebookov od Apple.

3) Domáce NAS zariadenie: Synology DS216play

Obrázok 14 Synology DS216 play (https://tech.sme.sk/c/20128709/pat-sposobov-ako-si-bezpecne-zalohovat-data.html#ixzz5GJYt9snn)

Nebojte sa investovať do NAS zariadenia. Dnešné modely zvládne nastaviť a obsluhovať aj priemerne zdatný používateľ počítačov, pričom výhody sú pri dobre zvolenom modeli veľmi zaujímavé.

Klasické zálohovanie je totiž len jednou z množstva funkcií, ktoré využijete aj v úplne bežnej domácnosti. Výrobca totiž pripravil veľmi jednoduchý a zároveň silný nástroj na archiváciu a následné prezeranie fotografií. Každý člen domácnosti, alebo menšej firmy, môže mať vyčlenený svoj dátový priestor a k súborom sa dá dostať aj mimo domova, nevyžaduje sa dokonca ani pevná IP adresa.

Z pohľadu domácnosti je najzaujímavejšou funkciou multimediálne centrum, ktoré dokáže pracovať aj s filmami v 4K rozlíšení a vďaka výkonnému čipu ich v reálnom čase upraví aj na prezeranie na bežnej televízii alebo na tablete.

Aj tu sa pripravte na vyššiu cenu, keďže len samotné NAS zariadenie stojí približne 270 eur. Dokúpiť si k nemu musíte minimálne jeden disk, pre zvýšenie bezpečnosti musia byť disky dva. Najlacnejšie NAS zariadenie kúpite aj za menej ako 90 eur, neočakávajte pri ňom ale vysoký výkon, zrkadlenie dát či spracovanie multimédií v reálnom čase.

4) Bezpečný USB kľúč: Kingston DataTraveler 2000

Obrázok 15 Flash disk (https://tech.sme.sk/c/20128709/pat-sposobov-ako-si-bezpecne-zalohovat-data.html#ixzz5GJYt9snn)

Pôsobí ako klasický USB kľúč určený na prenos dát, výrobca ho však vybavil množstvom bezpečnostných funkcií, vložil do pevného obalu a pridal ochranu typu IP57.

Nemusíte sa teda báť prachu a ani krátkodobého ponorenia do vody. Najdôležitejšia je ochrana údajov, kvôli ktorej si výrobca za zariadenie pýta pomerne dosť peňazí – konkrétne 119 eur za 16 gigabajtovú a 149 eur za 32 gigabajtovú verziu. Ochrana dát je dvojstupňová. Základom je hardvérové šifrovanie priamo v zariadení, takže do počítača nemusíte inštalovať žiadny softvér.

Druhý stupeň ochrany predstavuje číselný PIN kód, ktorý pred použitím zadávate na tlačidlovej klávesnici. O úspešnom odomknutí vás informuje malá svetelná LED dióda. Ak sa ku kľúču dostane nepovolaná osoba tak má na uhádnutie kombinácie len desať pokusov.

Na zariadení môžeme kritizovať jedine pomalšie rýchlosti pri zápise dát (20 MB/s pre 16 gigabajtovú verziu). Keďže sa tento typ zariadení nezvykne využívať na časté kopírovanie veľkých objemov dát, nemusí byť kritizovaná vlastnosť veľmi dôležitá.

5) Klasický externý disk: Seagate BackUp Plus Slim

Obrázok 16 Klasický externý disk (https://tech.sme.sk/c/20128709/pat-sposobov-ako-si-bezpecne-zalohovat-data.html#ixzz5GJYt9snn)

Najlacnejšiu možnosť zálohovania dát predstavuje klasický externý disk. V 2,5-palcovej verzii ho môžete nosiť neustále so sebou aj na služobné cesty, keďže si vystačí s napájaním z USB portu na notebooku. Tým ale jeho výhody v porovnaní s drahšími produktami končia, keďže neposkytuje žiadne pokročilé funkcie pre bezpečné zálohovanie údajov.

Na druhú stranu, pre bežnú zálohu väčšieho množstva dát je optimálnym riešením. Výrobca ho predáva v rôznych kapacitách od 500 megabajtov až po 4 terabajty, navyše, cena nie je vôbec vysoká.

Zálohovať svoje súboru budete musieť buď ručne, alebo si pomôžete firemnou aplikáciou. Tá podporuje aj zálohovanie z mobilných zariadení a dokonca aj sociálnych sietí. Keďže je vo vnútri len jeden disk, nie sú dáta zrkadlené a pri poruche sa musíte spoľahnúť na špecializované aplikácie, prípadne počas prvých dvoch rokov od kúpy vám zadarmo pomôže aj priamo výrobca.

Podobných externých diskov je v ponuke viacero, pri výbere sa preto vždy rozhodujte na základe kvality konštrukcie, obslužného softvéru a doplnkových funkcií.

MOTIVÁCIA

- Zálohujte pravidelne, minimálne raz do týždňa. Pri práci s dôležitými súbormi je vhodná automatická záloha prebiehajúca aj niekoľko krát denne.Nešetrite.
- Tie najlacnejšie disky a online služby si určite preverte, nízka cena sa totiž môže prejaviť na ich horšej kvalite, podpore, alebo bezpečnosti.

- Aktualizovaný antivírový softvér je podmienkou. Ani ten najlepší disk totiž nemusí odolať aplikáciám, ktoré šifrujú súbory a následne od ich vlastníka požadujú výkupné za ich odblokovanie.
- Disky a USB kľúče so zálohami si odkladajte na bezpečné miesta. Pohodené v prašnom prostredí alebo na priamom slnku strácajú na životnosti.
- Pri zálohe v cloude si zvoľte silné heslo, ktoré nepoužívate v žiadnej inej službe.

Dropbox

Webové úložisko, ktoré je založené na cloudovom riešení. Nahrádza klasické prenášanie súborov na CD, externých diskoch, flash diskoch pod. elektronickou cestou. Všetky vaše fotografie, textové materiály a multimediálne súbory máte kedykoľvek po ruke vo vašom počítači alebo v zdieľanom priečinku na internete.

Dropbox môžete používať na akomkoľvek bežnom operačnom systéme - Microsoft Windows, Linux a MacOS, aj na chytrých mobilných telefónoch s platformami Android, iOS alebo BlackBerry.

História vzniku služby Dropbox sa datuje do roku 2007. Založil ho študent Massachusettského technologického inštitútu (MIT) menom Drew Houston predovšetkým pre svoje potreby, pretože sa mu často stávalo, že zabúdal svoj USB disk s materiálmi.

Ine existujúce online riešenia pre úschovu súborov boli podľa jeho úsudku nedostatočné z hľadiska rýchlosti, kapacity a početných chýb, tak sa rozhodol vytvoriť vlastný systém a sprístupniť ho aj ďalším ľuďom, ktorí sa stretávali s rovnakým problémom. O Dropbox bol veľký záujem, časom bol doplnený o množstvo užitočných funkcií a dnes ho používajú milióny užívateľov.

Na webových stránkach www.dropbox.comsi najprv vytvorte svoj užívateľský účet. Overte svoju e-mailovou adresu kliknutím na odkaz v správe, ktorú by ste mali onedlho dostať. Stiahnite si a nainštalujte aplikáciu Dropbox do svojho počítača a prihláste sa k svojmu účtu.

Odkaz na stiahnutie aplikácie: https://www.dropbox.com/downloading

V nastavení programu (sekcia Advanced settings) si vyberte zložku (Dropbox location) pre ukladanie dát, ktorá sa bude automaticky synchronizovať s vaším diskom na serveri. A ďalej si zvoľte tiež zložky, ktoré sa majú synchronizovať s počítačom. Ak nenastavíte inú cestu, predvolené umiestnenie priečinka je C: / Users // Dropbox.

Nastavenie je kompletné. Teraz keď uskutočníte v Dropbox zložke akúkoľvek zmenu, uskutoční sa tiež v priečinku v cloude. Ak máte akékoľvek problémy s inštaláciou alebo nastavením služby, skúste navštíviť oficiálne nápovedu Dropboxu na stránke: https://www.dropbox.com/help

Dropbox je možné používať dvoma spôsobmi - buď bezplatne, alebo ako platenú verziu. U Dropboxu zadarmo môžete využívať pre uloženie súborov 2 GB priestoru, ktorú môžete pomocou pozvánok ďalším užívateľom rozšíriť až na 16 GB.

Druhou možnosťou je platená verzia, v tomto prípade máte na výber z dvoch variantov Standard a Advanced. Prvý vychádza na 10 € mesačne, ak si ju predplatíte na celý rok. V cene je 2TB

kapacita, integrácia Office 365, múdra synchronizácia, 56 bitové AES a SSL / TLS šifrovanie dát a mnoho ďalšieho. Naopak Dropbox Advanced ponúka od 15 € mesačne neobmedzené úložisko a ďalšie rozšírené funkcie.

výhody

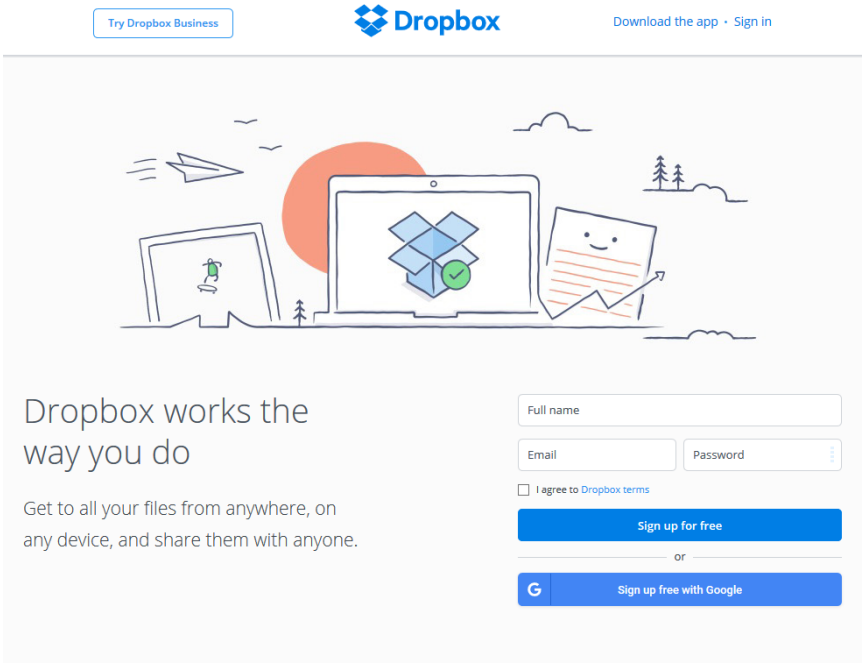
- Spoľahlivé cloudové úložisko pre zálohovanie a synchronizáciu súborov pre súkromné i firemné účely.
- Možnosť zdieľania dokumentov medzi ďalších používateľov, ktorí pracujú napríklad na rovnakom projekte.
- K svojim súborom môžete okrem aplikácie na počítači pristupovať tiež cez desktopové rozhranie alebo smartphone.
- Vďaka popularite službu podporuje stále viac aplikácií.

nevýhody

- Bezplatná verzia ponúka len 2GB úložisko. Poplatok sa pohybuje od 10 € za mesiac.
- Rozhranie je síce intuitívne, avšak nepodporuje slovenský jazyk.

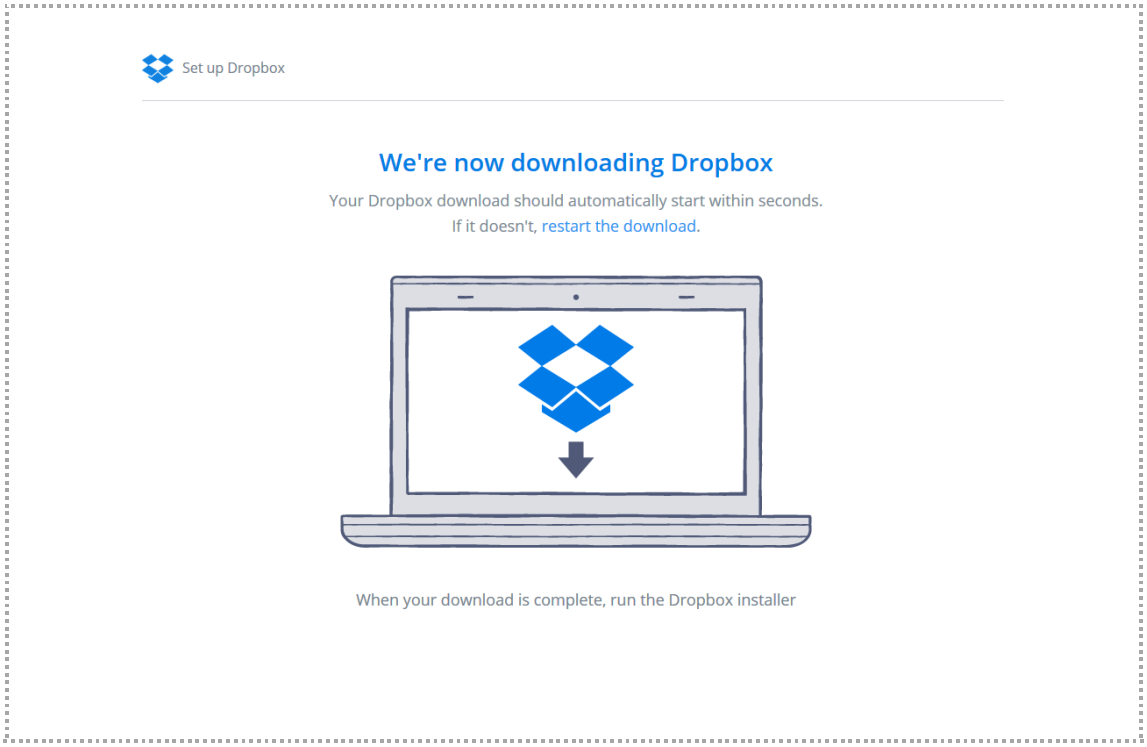
Práca so službou Dropbox

Prvým krokom pre užívanie služieb Dropboxu je vytvorenie účtu. Založenie účtu je zdarma a služby možno zdarma využívať s určitými obmedzeniami. Nový užívateľ využívajúci bezplatných služieb má k dispozícii 2GB kapacity Dropboxu.Účet na Dropboxe je možné založiť na stránkach www.dropbox.com. Je nutné vyplniť potrebné údaje a potvrdiť vytvorenie účtu (https://ksvi.mff.cuni.cz/).

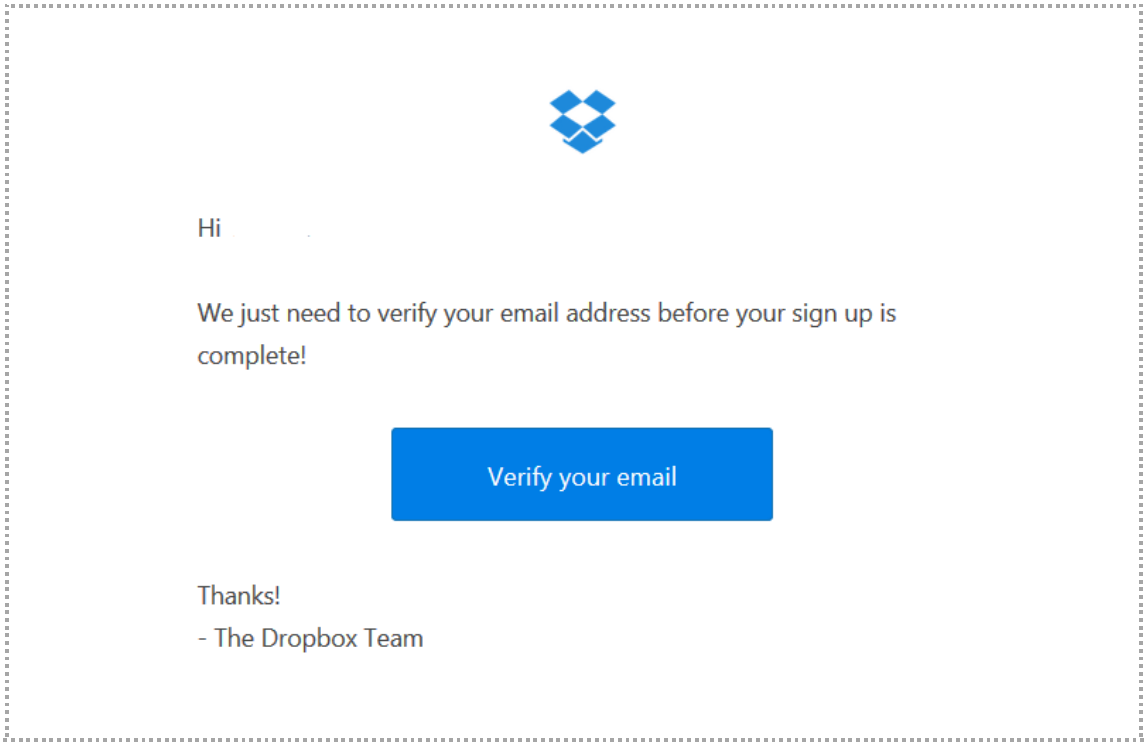


Obrázok 17 Založenie účtu (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)

Následne nato sa začne sťahovať inštalačný program a zároveň s ním je odoslaný aj potvrdzujúci mail obsahujúci nutný odkaz k dokončeniu vašej registrácie.

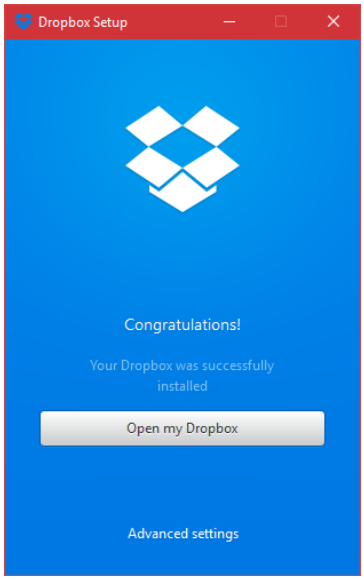


Obrázok 18 Sťahovanie programu (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)



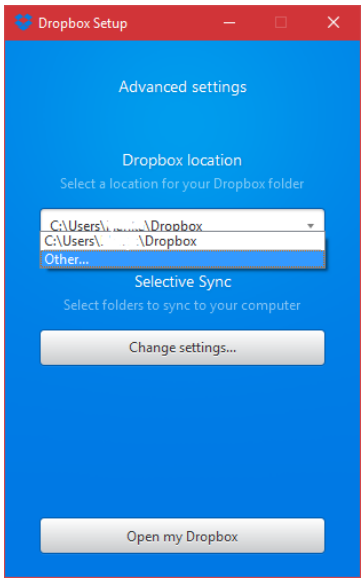
Obrázok 19 Potvrdenie odkazu poslaného na váš email

Spustením stiahnutého súboru sa nainstaluje Dropbox a po nainštalovaní sa otvorí uvítacie okno.



Obrázok 20 Okno s možnosťou nastavenia (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)

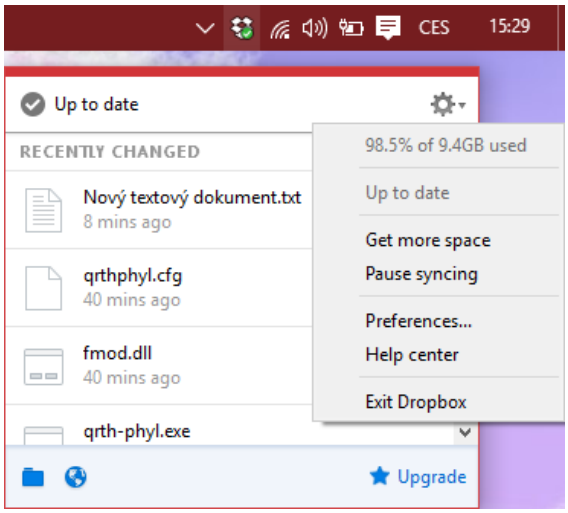
Kliknutím na možnosť Advanced settings sa zobrazí nastavenie priečinkov Dropboxu. Prvou položkou (Dropbox location) je voľba miestnej zložky, v ktorej budú uložené dáta synchronizované s Dropboxom. Druhou položkou (Selective Sync) je výber zložiek v Dropboxe, ktoré majú byť synchronizované s konkrétnym počítačom. Oba výbery je možné kedykoľvek zmeniť v nastaveniach.



Obrázok 21 Okno s možnosťou nastavenia (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)

Teraz je v počítači dostupná zložka synchronizovaná s Dropboxom. Predvolené umiestnenie synchronizovanej zložky je C: / Users / <user> / Dropbox, pokiaľ nebolo zmenené po inštalácii. Všetok obsah zložky je synchronizovaný s Dropboxom. Ak je niečo pridané alebo vymazané, aktualizuje sa stav na Dropboxe a všetkých prepojených zariadeniach. Navyše Dropbox pri inštalácii integruje možnosti do kontextovej ponuky. Pre súbory mimo synchronizovaných

priečinkov pridáva možnosť presunutia do zložky Dropbox. Pre súbory, ktoré už sú v Dropboxe pridáva možnosti zdieľania. Dropbox beží na pozadí, pre rýchly prístup možno využiť ikonu v oznamovacej oblasti. V ponuke je zobrazený zoznam posledných prenosov synchronizácie, ikony pre otvorenie priečinka a webového rozhrania a ďalšie možnosti a nastavenia.



Obrázok 22 Rýchly prístup do Dropboxu (http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more)

CVIČENIE – VYTVORTE!

Vytvorte si vlastné úložisko s využitím pomocou služby google.

4 KOMUNIKÁCIA MEDZI POČÍTAČOM A PERIFÉRNYMI ZARIADENIAMI – 3D TLAČIARNE

Takmer všetky programy potrebujú pre svoju činnosť vstupné údaje, ktoré spracujú a zobrazia výsledky, čiže poskytnú výstupné údaje. Počítač vlastne prostredníctvom nich komunikuje s okolím.

Vstupno-výstupné zariadenia (ďalej V/V alebo I/O zariadenia) alebo tiež periférie sú všetky jednotky, ktoré zabezpečujú zber údajov od používateľov alebo z iných zariadení a naopak sprostredkujú údaje používateľom alebo ich odosielajú iným zariadeniam. Zariadenia rozdeľujeme podľa rôznych faktorov (https://gkmke.sk/):

PODĽA funkčnosti :

- Pre pripojenie počítač – človek (klávesnica, myš, trackball, monitor, tlačiareň...).
- Počítač – stroj (prevodníky kódov, D/A a A/D prevodníky ...).
- Externé pamäte (páskové jednotky, diskové polia).

PODĽA smeru prenášanej informácie:

- Vstupné (klávesnica, skener, myš ...).
- Výstupné (monitor, tlačiareň ...).
- Vstupno-výstupné (modem, disky).

4.1 3D Tlačiarne

Základným princípom fungovania 3D tlačiarne – postupné zostavovanie modelov vrstvu po vrstve, ktoré dostali názov prísada. Najčastejšie sa používa plast, ktorý sa roztaví a stuhne alebo prášok, ktorý tvrdne pod laserom alebo UV svetlom. Tiež sa používa kov, keramické hmoty, sadra a dokonca aj jedlo - krém, čokoláda. Použité technológie sa vyvíjajú v dvoch smeroch: laserové a atramentové tlačenie na 3D tlačiarni.

VÝKLAD

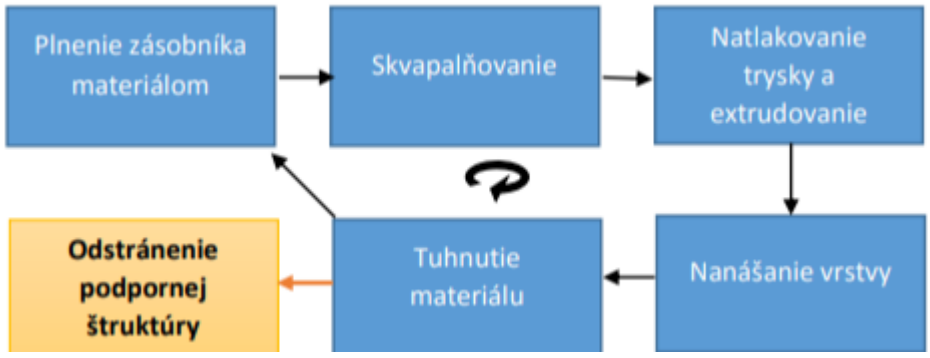
3D tlač je technológia aditívneho nanášania vrstiev materiálu na seba s cieľom vytvoriť reálny trojdimenzionálny objekt(http://www.adlerka.sk).

Rozlišujeme nasledovné metódy:

- Aditívna metóda tzv. sčítavacia, znamená, že materiál sa pridáva vrstvami na podložku. Pri aditívnej metóde nevzniká odpad.
- Subtraktívna metóda z bloku sa materiál uberá, odčítava. Napríklad frézovaním, obrábaním a pod. Vtedy vzniká nepoužiteľný odpad.

3D tlačiarne podľa technológie rozdeľujeme na:

- Nanášanie nataveného materiálu tryskou – Materiál sa v tlačovej hlave zahrieva do polotekutého stavu. Následne sa vytláča cez trysku na tlačiacu plochu. Chladnutím materiál tuhne. Po stuhnutí sa môže naniesť ďalšia vrstva.

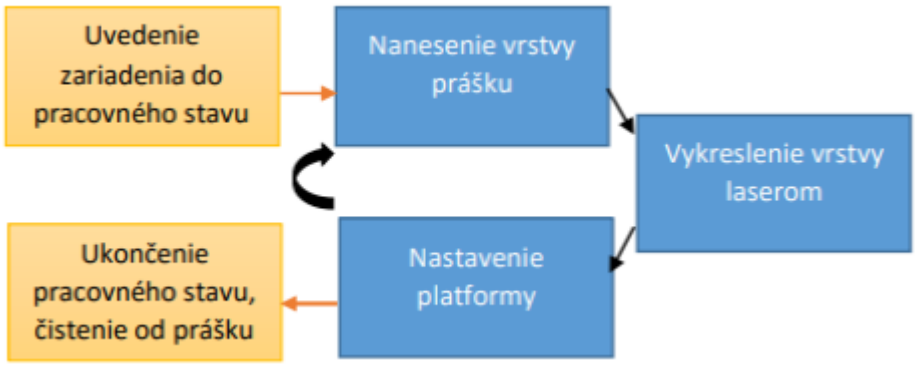


Obrázok 23 Cyklus nanášania vrstiev (http://www.adlerka.sk/UserFiles/File/3D_tlac.pdf)

Technológia FDM – Fused Deposition Modeling Technológia pracuje princípom nanášania nataveného materiálu tryskou. Materiál je polymér ABS vo forme vlákna. Polymér ABS je mechanicky pevný a trvácny. Okrem ABS je možné využívať aj ďalšie materiály ako napr. polykarbonát, PC/ABS PPSF. Využitie: prezentačné účely, funkčné prototypy, vzory na odlievanie.

- Spracovanie práškového materiálu PBF
  - SLS (selektívne laserové spekanie) Princíp:
    1. nanesenie vrstvy prášku
    2. laser zapečie prášok do pevnej štruktúry v určených bodoch
    3. nanesenie novej vrstvy prášku
    4. zopakovanie bodu 2
    5. ukončenie pracovného stavu, čistenie od prášku

Materiály: konštrukčné plasty, kompozity s vysokou tuhosťou, gumové materiály atď. Využitie: časti áut, kabíny lietadiel, potrubia, sterilné lekárske potreby, tesnenia, formy na odlievanie a pod.



Obrázok 24 SLS technológia (http://www.adlerka.sk/UserFiles/File/3D_tlac.pdf)

- LOM (Laminated Object Manufacturing) – laminovanie, vzájomné ukladanie vrstiev materiálu na seba a ich tepelné spájanie

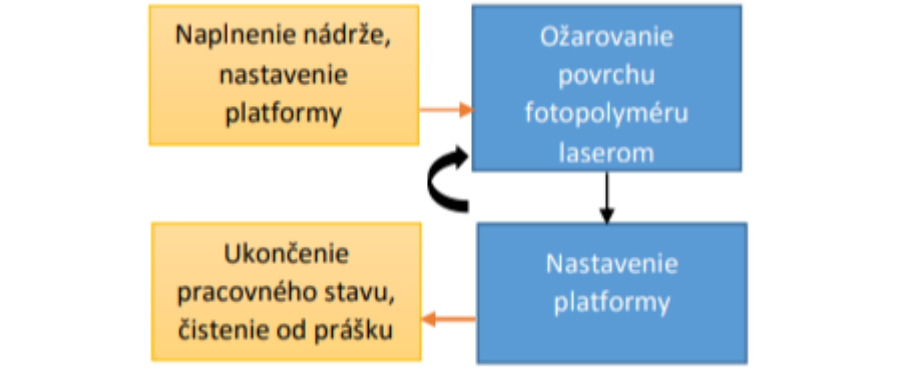
Princíp:

1) Doplnenie zásobníka, nastavenie prvej vrstvy
2) Laminovanie zahrievaným valčekom
3) Vyrezávanie laserovým systémom
4) Posun platformy nižšie a nastavenie novej vrstvy
5) Posun platformy o úroveň vyššie
6) Odstránenie odpadu

Nevýhody: menšia odolnosť spájaných vrstiev, vznik odpadu, nižšia kvalita

- Technológia fotopolymerizácie - Využíva sa materiál citlivý na fotóny. Po dopade fotónov sa kvapalný materiál mení na pevné skupenstvo.

Stereolitografia (SLA): Nádoba sa naplní materiálom citlivým na fotóny. Nastaví sa platforma na najvyšší bod. Ožiari sa hladina fotopolyméru laserom, čím vzniká proces tuhnutia. Následne sa posunie platforma o vrstvu nižšie a proces sa opakuje. Na záver sa objekt vytvrdí v UV komore. Pri tejto technológii sú potrebné podporné štruktúry



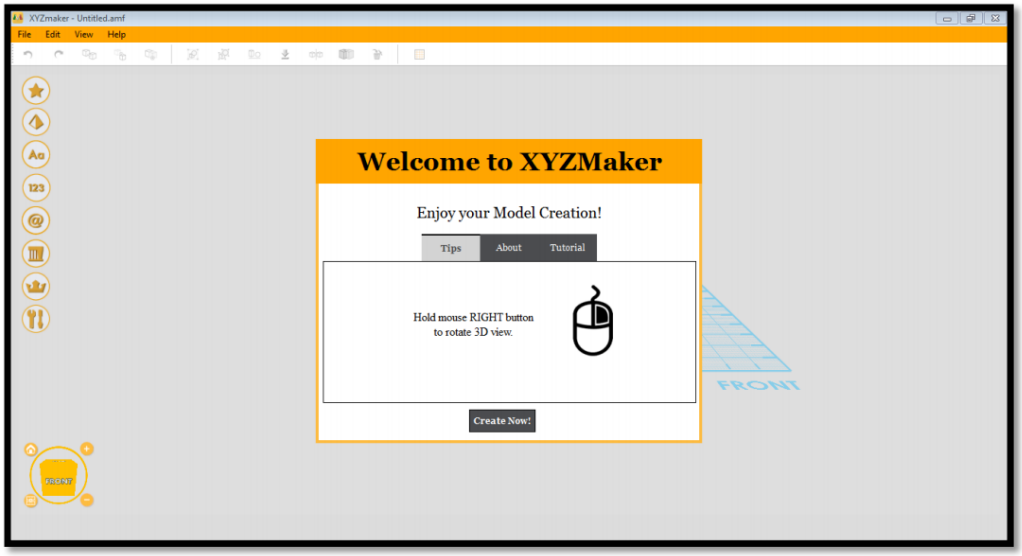
Obrázok 25 SLA technológia (http://www.adlerka.sk/UserFiles/File/3D_tlac.pdf)

4.2 XYZ MAKER

XYZmaker je desktopová 3D dizajnová aplikácia, ktorá vám umožní vytvoriť skutočný objekt podľa vašej predstavy a následne ho vytlačiť na 3D tlačiarni.. Môžete ľahko vytvoriť a prispôsobiť si svoje vlastné vzory. Program je vhodný pre odborníkov ale i začiatočníkov. Môže sa tiež zmeniť rozmer vášho myslenia. Dozviete sa ako vytvoriť časti jednoduchého 3D modelu. Vybrali sme jednoduchý geometrický tvar , ktorý by ste mohli zostaviť v programe XYZ a následne ho vytlačiť. Skôr než začnete pracovať s programom je potrbné sa s ním oboznámiť.

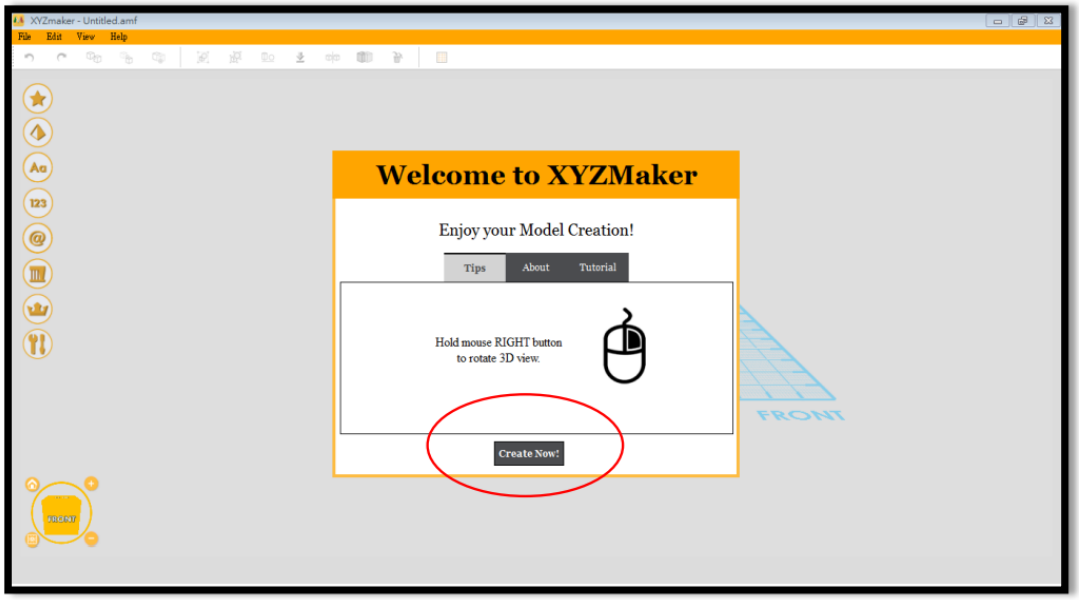
VÝKLAD

Po spustení aplikácie XYZmaker sa zobrazí úvodná obrazovka, ktorá vás privíta s ponukou na tipy, ovládanie a tútoriál. V pozadí môžete vidieť okno, v ktorom sa dá vytvárať konkrétny model.



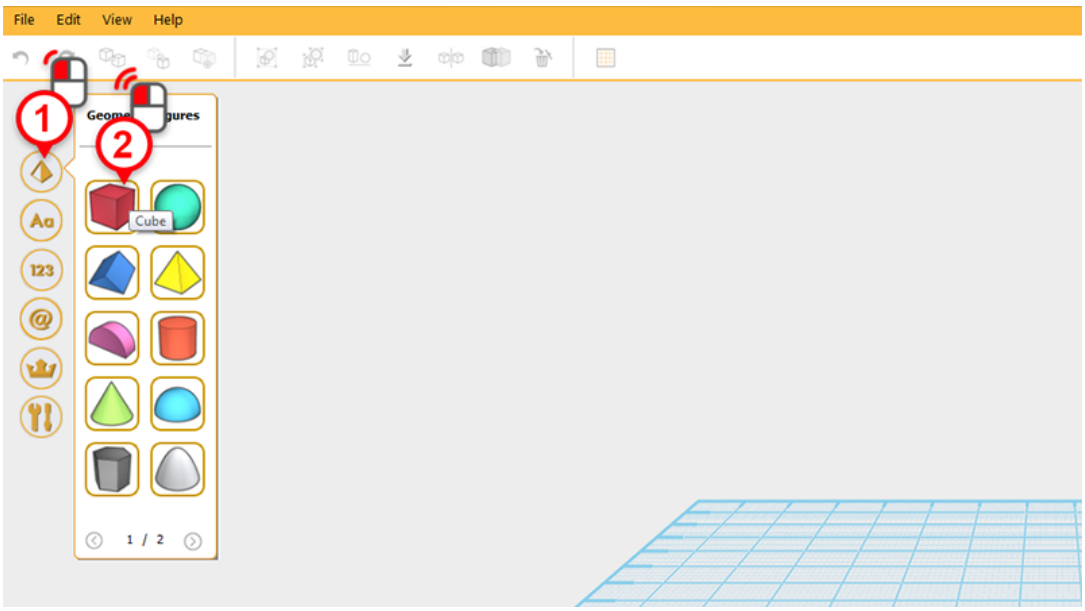
Obrázok 26 Spustený program (XYZaker Software user manual)

Aby sme si dokázali vytvoriť 3D objekt, musíme sa za pomoci jednoduchého príkladu oboznámiť s touto aplikáciou. Kliknite na položku Vytvoriť teraz. Následne nato sa vám vytvorí plocha, na ktorej môžete začať vytvárať vlastný model.

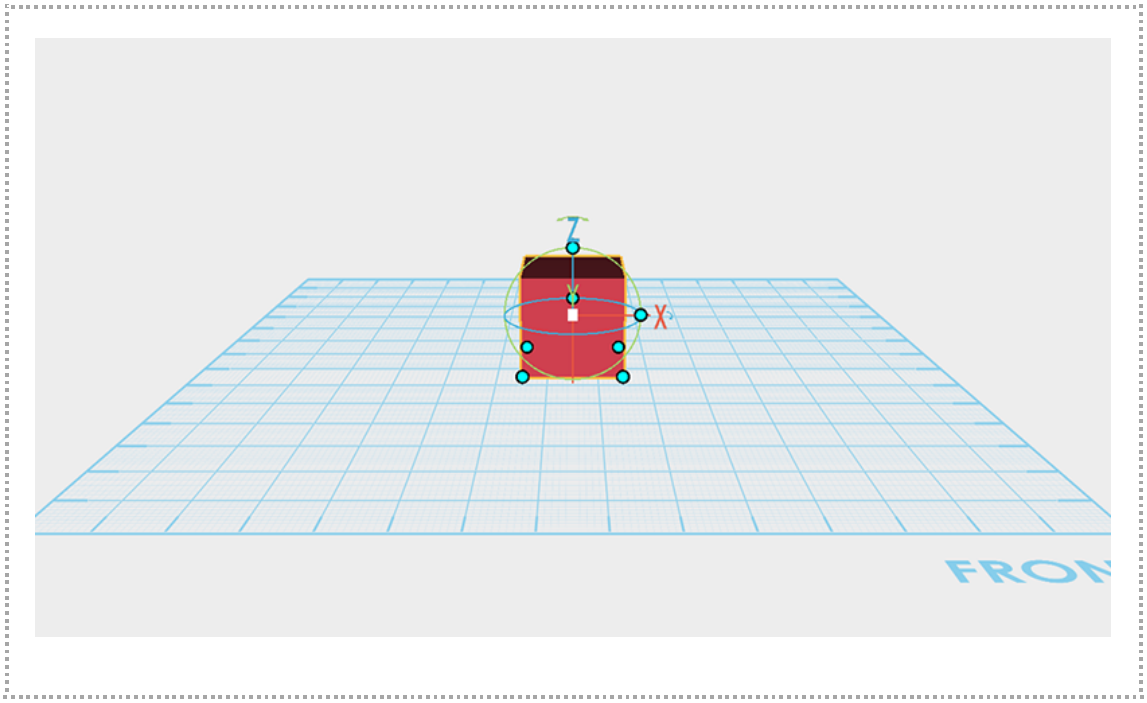


Obrázok 27 Začiatok tvorby vlastného modelu (XYZaker Software user manual)

Tvorba vlastného modelu

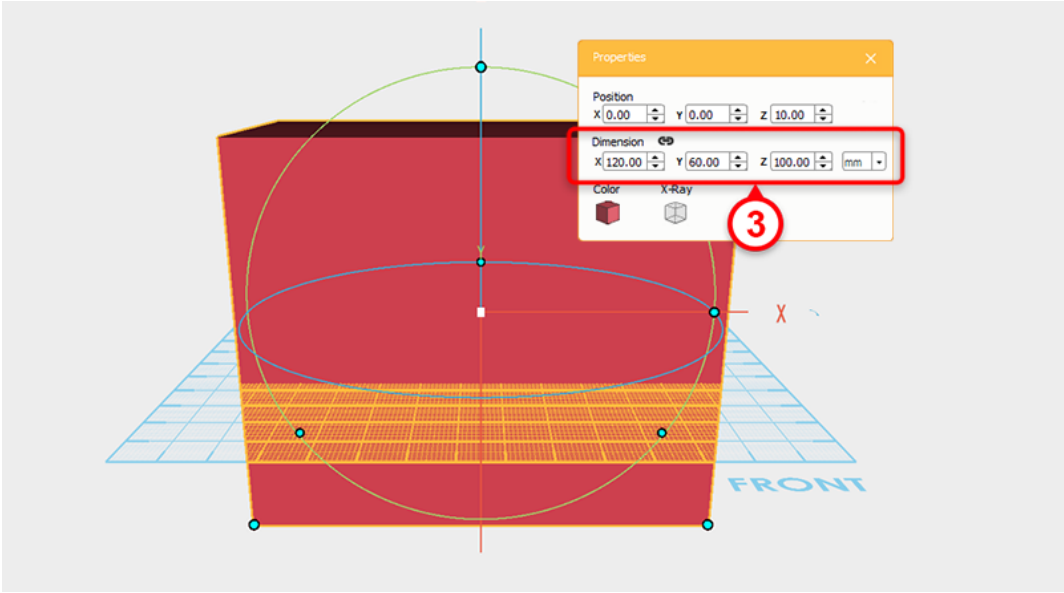


Obrázok 28 Začiatok tvorby vlastného modelu (XYZaker Software user manual)



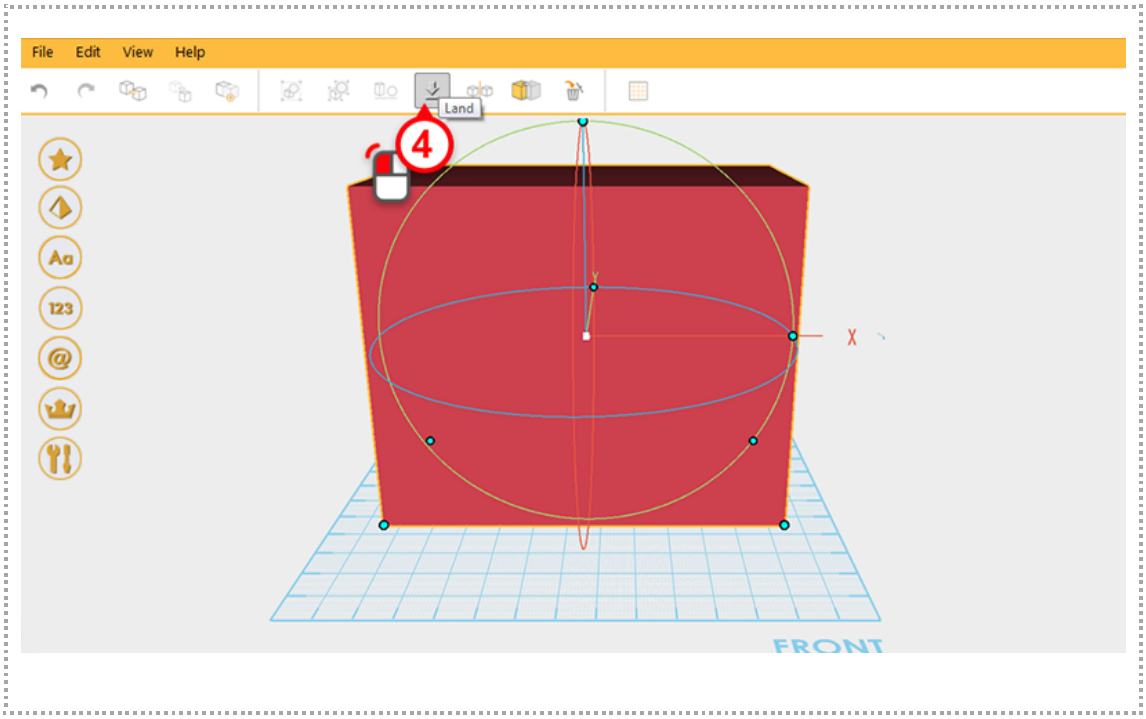
Obrázok 29 Vloženie kocky na mriežku (XYZaker Software user manual)

Na ľavej strane nájdete modelovú lištu. V tomto menu nájdite a kliknite na ikonu pyramídy označenú číslom (1), potom dvakrát kliknite na ikonu kocky (2). Vybraná kocka sa dostane do stredu mriežky. Ak jednoducho kliknete na ikonu raz, môžete pretiahnuť 3D objekt a umiestniť ho do mriežky manuálne.



Obrázok 30 Nastavenie veľkosti modelovanej kocky (XYZaker Software user manual)

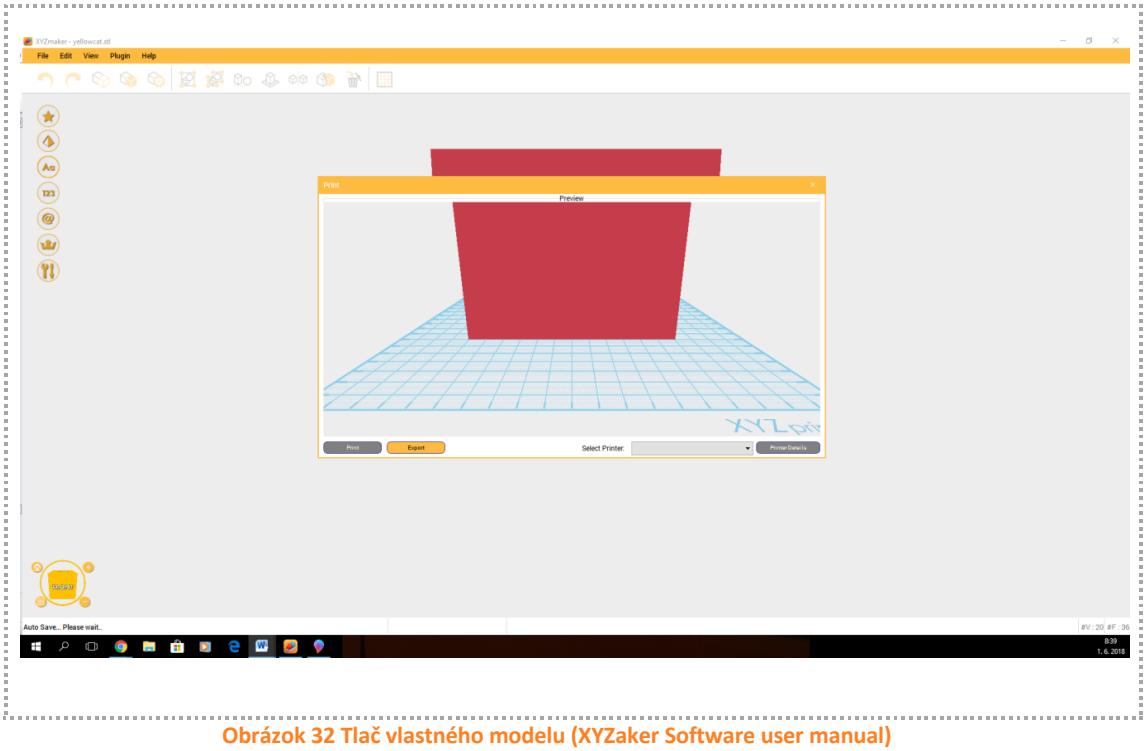
V okne Vlastnosti zmente rozmery kocky na X: 120, Y: 60, Z: 100 mm.



Obrázok 31 Nastavenie kocky na modelovú mriežku (XYZaker Software user manual)

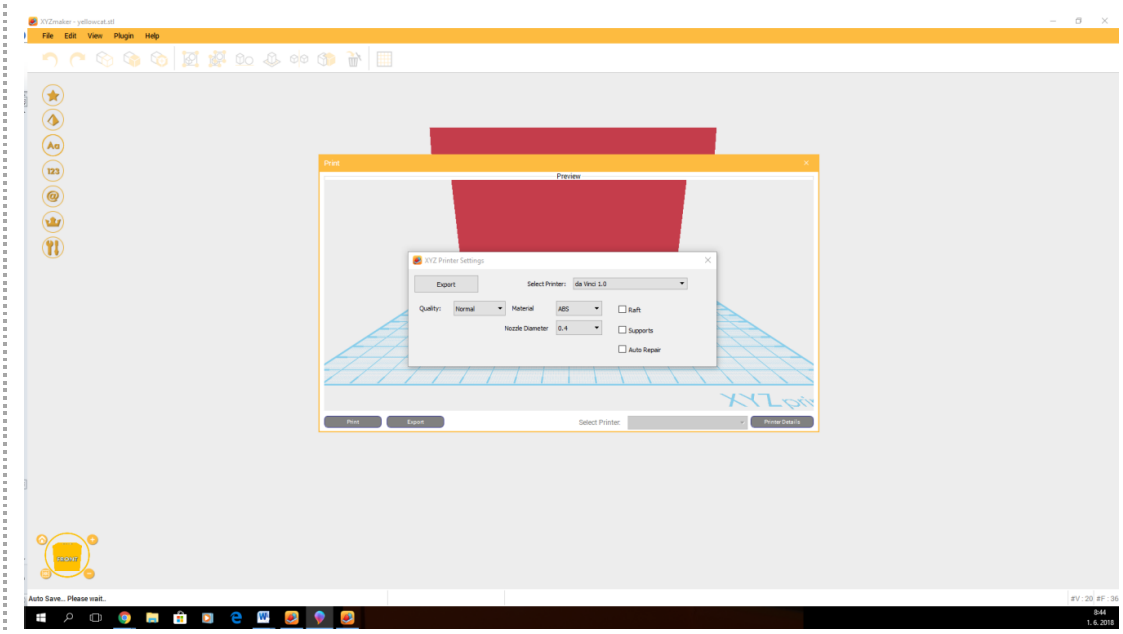
Vzhľadom k tomu, že výška kocky sa zvýšila, spodná polovica je teraz zakrytá modelovou mriežkou. Stlačením tlačidla Land prilepte základňu kocky do modelovej mriežky.Po vytvorení samotného objektu môžete pristúpiť k samotnému tlačeniu.

V ponuke File si vyberte možnosť print a kliknite na ňu. Následne na to sa vám zobrazí nasledovné okno.



Obrázok 32 Tlač vlastného modelu (XYZaker Software user manual)

V dialógovom okne vyberte položku export. Následne nato sa vám zobrazí ponuka, kde si môžete nastaviť aký materiál použijete na tlač, typ tlačiarne a aj kvalitu tlače.



Obrázok 33 Nastavenie vlastností tlače (XYZaker Software user manual)

Potom kliknite na položku export. V ďalšom kroku vás program vyzve aby ste svoj model uložili. Následne nato kliknite na print a tlač žačne prebiehať. V závislosti od typu tlačiarne môžete prevádzať dodatočné technické nastavenia priamo na tlačiarni. Napríklad, môžete nastaviť teplotu, pri ktorej sa material začne taviť.

5 MIKROKONTROLÉR ARDUINO

Mikrokontrolér je malý počítač v jedinom čipe, riadi vložený systém. Má svoju pamäť a procesor Nemá operačný systém ani periféria

Jeden zo zakladateľov Arduino, Massimo Banzi, uvádza, že platforma tohto zariadenia je open source. Založená je na jednoduchej vstupno-výstupnej doske a vývojovom prostredí, ktoré implementuje jazyk Processing. Ďalej uvádza, že sa zariadenie Arduino môže používať a rozvíjať samostatne, ale je možné sa pripojiť aj k softvéru, ktorý beží na počítači. Arduino môžeme objednať už zostavené alebo si môžeme zakúpiť komponenty a zostaviť si dosku sami. Aby dokázali, že je tento pristroj naozaj open source, vývojári zverejnili celú schému základnej dosky, ako aj nástroj na programovanie mikroprocesoru IDE, ktorý je tiež open source a zadarmo.

Arduino I/O Board bol tradične založený na Atmel AVR ATMEGA8. Doska obsahuje tiež sériový port, napájacie obvody, rozširujúce konektory a rôzne podporné zložky (Wheat, 2011). Od začiatku projektu Arduino v roku 2005, sa predalo viac ako 150.000 oficiálnych dosiek v celom svete. Počet klonov, ktorý sa predal, pravdepodobne dosahuje viac ako pol milióna.

V nasledujúcej tabuľke porovnávame vlastnosti niekoľkých zariadení Arduino. Porovnávali sme veľkosť Flash pamäte zariadenia, počet analógových, digitálnych a PWM vstupov a výstupov, rýchlosť a typ procesora, ako aj aktuálnu cenu zariadenia, ktorá sa nachádza na oficiálnej stránke Arduino http://store.arduino.cc/.

Tabuľka 3 Porovnanie zariadení Arduino

Arduino	Meno zariadenia	Pamäť	Digitálne I/O	Analógové I/O	PWM	Rýchlosť procesora / Mikrokontrolér
	Leonardo	32Kb	20	12	7	16MHz / ATmega32u4
	Uno - R3	32 KB	20	6	6	16 MHz / ATmega328
	Mega 2560	256 KB	54	16	15	16 MHz / ATmega2560
	Nano	16 KB	14	8	6	16 MHz / ATmega168
	Pro Mini	16 KB	14	8	6	8 MHz / ATmega168

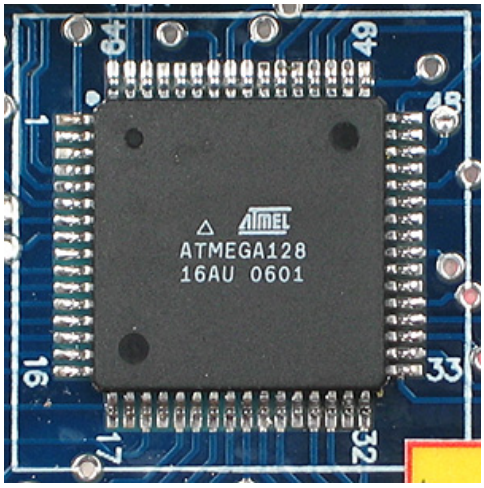
	Fio	32 KB	14	8	6	8 MHz ATmega328P
	Micro	32 KB	20	12	7	16 MHz ATmega32u4

5.1 Arduino – 8 bitový mikrokontrolér

Základ Arduina tvorí 8 bitový mikrokontrolér s AVR architektúrou od firmy Atmel.

VÝKLAD

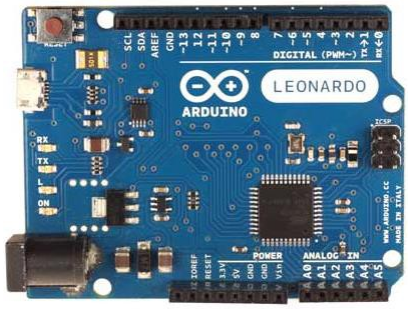
Mikrokontroléri disponujú rôznymi perifériami, napríklad čítač/časovač , ktorý môže slúžiť na rôzne účely ako presné časovanie udalostí, generovanie PWM, čítanie frekvencii, atď. ďalšou periférioumôže byť AD prevodník (prevod analógovej hodnoty na digitálnu t.j. prevod napätia na číslo). Aby sme mohli takéto periférie použiť musíme poznať patričnéregistre (každá periféria môže mať vlastné). Správnym nastavením registrov tak docielime správnu funkcionalitu periférie. Teraz sa ukáže jedna veľká výhoda Arduina – na to aby sme mohli niečo vytvoriť, naprogramovať nemusíme poznať žiadny register, nemusíte sa učiť architektúru žiadneho mikrokontroléra (ak nechcete). Samozrejme vedieť viac o architektúre nie je na škodu, ale nie je to nutné na to aby ste niečo vytvorili. Ďalšou výhodou je spôsob akým sa Arduino programuje, nie je totižto nutné použiť programátor pre mikrokontroléri. Mikrokontrolér na plošnom spoji Arduina obsahuje bootloader, ktorý nám zabezpečuje naprogramovanie Arduina po sériovej linke a následné spustenie programu v mikrokontroléri. Programovanie je jednoduché stačí pripojiť Arduino pomocou USB kábla k PC vo vývojovom prostredí vybrať správny port, verziu Arduina a kliknúť na tlačítko „Upload“ alebo klávesy Ctrl+U (http://mirobozik.sk/).



Obrázok 34 Osem bitová verzia mikrokontroléra Arduino

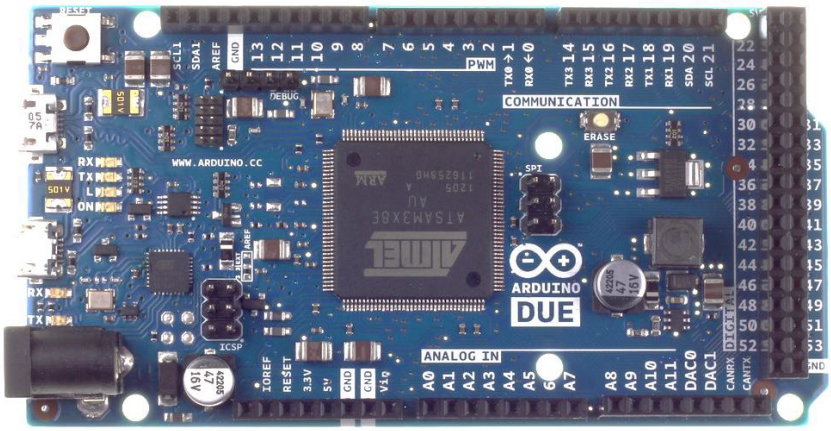
5.2 Verzie Arduina

Arduino existuje vo viacerých verziách. Existujú verzie v tzv. štandardnej veľkosti ako napríklad Arduino UNO, Leonardo, alebo zo starších Duemilanove.



Obrázok 35 Verzia mikrokontroléra Arduino Leonardo

Potom sú verzie „veľké“ a to Arduino Mega 2560, Arduino Mega ADK a Arduino Due.



Obrázok 36 Verzia mikrokontroléra Arduino DUE

Do tretice sú aj malé verzie ako Arduino Pro Mini, Arduino Micro, Arduino Nano a Lilypad Arduino.

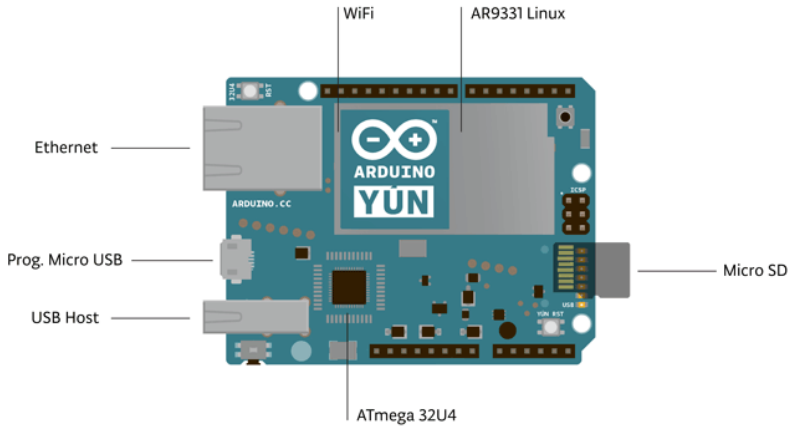
Obrázok 37 Verzia mikrokontroléra Arduino Pro Mini

Každá verzia sa hodí na niečo iné. Arduino vyberáme podľa projektu, ak potrebujem malé rozmery, vyberám z tých malých verzii, záleží ešte na tom či budem potrebovať s USB konektorom alebo bude stačiť bez. Ak robíme samostatný projekt, ktorý nebude komunikovať s

PC, USB v tomto prípade potrebovať nebudeme, respektíve USB bude nutné iba pri programovaní Arduina. Verzie Arduina bez USB portu sa programujú pomocou USB to Serial adaptéra.

5.3 Arduino Yun

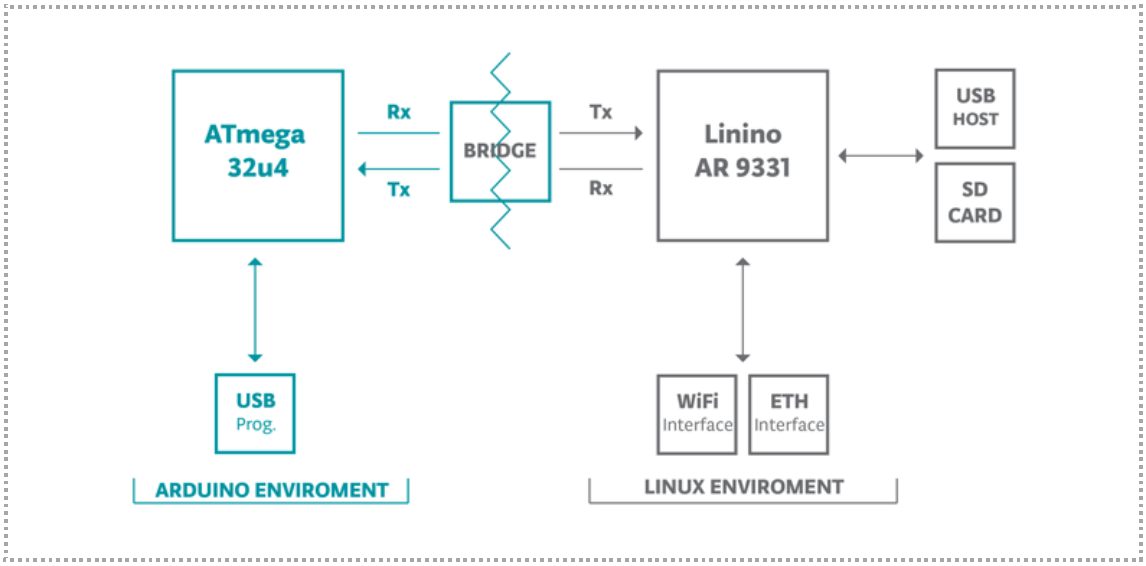
V tejto časti učebného materiálu si popíšeme dosku Arduino Yun. Jadrom regulačného modulu je mikrokontrolér Arduino YÚN, ktorý je založený na čipe ATmega32u4 a procesore Atheros AR9331. Procesor Atheros podporuje distribúciu Linuxu založenom na OpenWrt pod názvom OpenWrt-Yun. Zariadenie disponuje podporou zabudovaného Ethernetu, WiFi, portom USB-A, micro-SD slotom, dvadsiatich digitálnych vstupno/výstupných pinov, 16MHz kryštálovým oscilátorom, micro USB konektorom a tromi resetovacími tlačidlami. Toto zariadenie sa ukázalo ako skvelá voľba pretože v sebe spája viaceré výhody. Čip ATmega32u4 poskytuje čítanie a zapisovanie dát na výstupné porty v reálnom čase, ich spracovanie prebieha teda veľmi rýchlo a s minimálnou odozvou.



Obrázok 38 Arduino Yun board

VÝKLAD

Čip Atheros umožňuje používať spomínané periférie (WiFi, Ethernet, USB, micro-SD) a poskytuje vhodné prostredie na riešenie zložitejších úloh akými sú TCP komunikácia so vzdialeným serverom, spravovanie databázy a v podstate všetky Linuxom podporované funkcie. Na 10 vidíme blokovú schému ako tieto dva čipy medzi sebou komunikujú. Komunikácia je umožnená pomocou knižnice, ktorá sa volá Bridge library. pomocou tejto knižnice je možné priamo zo zdrojového kódu Arduina zavolať spustenie shell skriptu, komunikovať so sieťovými rozhraniami a prijímať informácie z procesoru AR9331.



Obrázok 39 Bloková schéma komunikácie ATmega32u4 a Atheros AR9331

Napájanie a reset mikrokontroléra

Arduino dosky môžeme napájať 4 spôsobmi (Nano iba tromi):

1) ***USB káblom*** (5V), ktorý môže slúžiť zároveň na komunikáciu s PC
2) ***napájacím konektorom*** 2.1mm x 5.5mm - sieťový adapteur alebo batérie 7V-20V (neaplikovateľné pre Nano)
3) ***pinom Vin*** - akýmkoľvek elektrickým napätím 7V-12V
4) ***pinom 5V*** - stabilizovaným elektrickým napätím +5V

Elektrické napätie z príslušných pinov (GND, 5V) môžeme zároveň používať pre vlastné zapojenia a obvody (aj +3.3V z pinu 3.3V, okrem prípadu, keď používame Nano a nenapájame ho pomocou USB kábla).

Na resetovanie mikrokontroléra môžeme stlačiť tlačidlo RESET na doske Arduina. Pull up rezistor na doske zabezpečuje stav pinu RST na hodnotu HIGH (1, +5V) a po stlačení tlačidla RESET sa stav na pine RST zmení na LOW (0, 0V, GND). Mikrokontrolér môžeme resetnúť aj privedením GND (LOW, 0, 0V) na pin RST.

Digitálne I/O piny

Na digitálnom pine môže byť vo všeobecnosti iba jedna z dvoch hodnôt 0 alebo 1, ide o binárnu hodnotu (neplatí pre PWM výstup). Hodnota 0 znamená, že na digitálnom pine je elektrické napätie 0V, teda GND (alebo menej ako 2V pre vstup). Túto skutočnosť môžeme vyjadriť stavmi LOW alebo 0. Hodnota 1 znamená, že na digitálnom pine je elektrické napätie +5V (alebo viac ako 3V pre vstup). Túto skutočnosť môžeme vyjadriť stavmi HIGH alebo 1.

Pri digitálnych výstupných pinoch v móde PWM sa na pin posiela celé číslo z rozsahu 0 až 255. Veľmi zjednodušene si to môžeme predstaviť ako namapovaný rozsah el. napätia 0V až 5V do číselného intervalu 0 až 255 a zariadenie pripojené na takýto výstup sa chová, akoby sme zmenou číselnej hodnoty 0-255 na pine zmenili el. napätie na zariadení. Praktické využitie je

napr. pri postupnom rozsvecovaní a zhasínaní LED, pri ovládaní rýchlosti motorov pomocou ovládača H-bridge a pod.

Pri použití digitálneho pinu ako vstupu je možné využiť aj interný pull up rezistor priamo v mikrokontroléri. Digitálny pin č.13 (LED_BUILTIN) je už priamo spojený s LED na doske. Treba s tým uvažovať najmä v prípade, ak chceme pin č.13 využiť ako vstupný (využitie napríklad ako jednoduchý blikač zostavený pomocou diódy a slučky). Analógové vstupné piny môžu byť použité aj ako všeobecné digitálne I/O piny (okrem pinov A6 a A7 na doske Arduino Nano, tie sú výhradne analógové), viď vyššie.

Analógové vstupy

Pri použití analógových vstupných pinov tieto pracujú ako A/D prevodník a nadobúdajú hodnoty od 0 do 1023 v závislosti od elektrického napätia na vstupnom analógovom pine. Tento rozsah 0V-5V možno zmeniť buď softvérovo alebo privedením referenčného el. napätia (max. +5V) na pin AREF (táto referenčná hodnota sa stane max. hodnotou pre namapovanie napäťového rozsahu do číselného intervalu 0-1023).

Ďalšie funkcie pinov

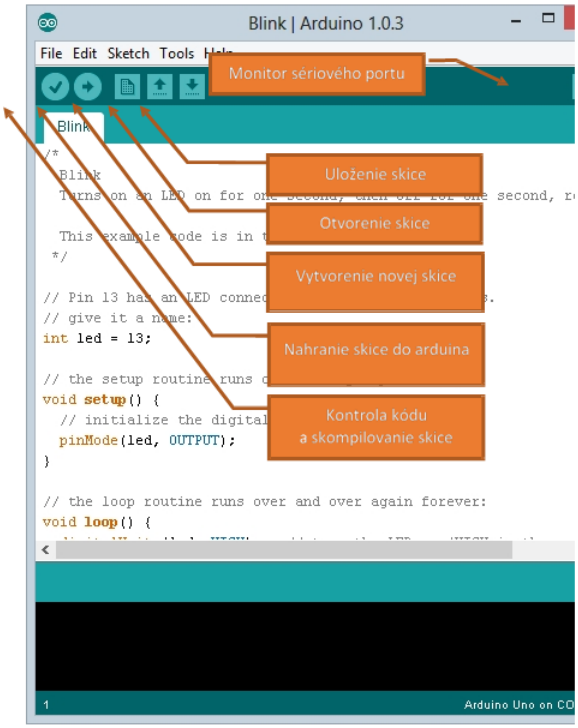
Niektoré piny mikrokontroléra majú okrem svojej I/O (vstupno-výstupnej) funkcie ešte jednu alebo viac ďalších funkcií, napr. RX, TX (serial) pre komunikáciu so SW Arduino (upload, serial monitor), SDA, SCL pre komunikáciu po zbernici i2c (ovládanie displeja LCD 1602), INT0, INT1, ... pre ošetrenie externých prerušení. Je dobré na toto pamätať už pri návrhu na ktorý pin čo bude pripojené. Napríklad nevhodným obsadením pinov RX a TX sa môže znefunkčniť upload a komunikácia s PC.

5.4 Vývojové prostredie pre Arduino

Arduino IDE je cross-platformová aplikácia napísaná v Jave. Je navrhnutá tak, aby umožnila programovanie začínajúcim vývojárom neoboznámených s vývojom softvéru. Obsahuje editor kódu s funkciami, ako je zvýraznenie syntaxe, Brace Matching, a automatické odsadenie, a je tiež schopný kompilovať a nahrávať programy do Arduina pomcou jediného kliknutia. Program alebo kód písaný pre Arduino sa nazýva skica.

VÝKLAD

Ak spustíme Arduino IDE (súbor arduino.exe v rozbalenom priečinku arduino-1.0.2) naskytne sa nám pohľad podobný ako na obr. 41.

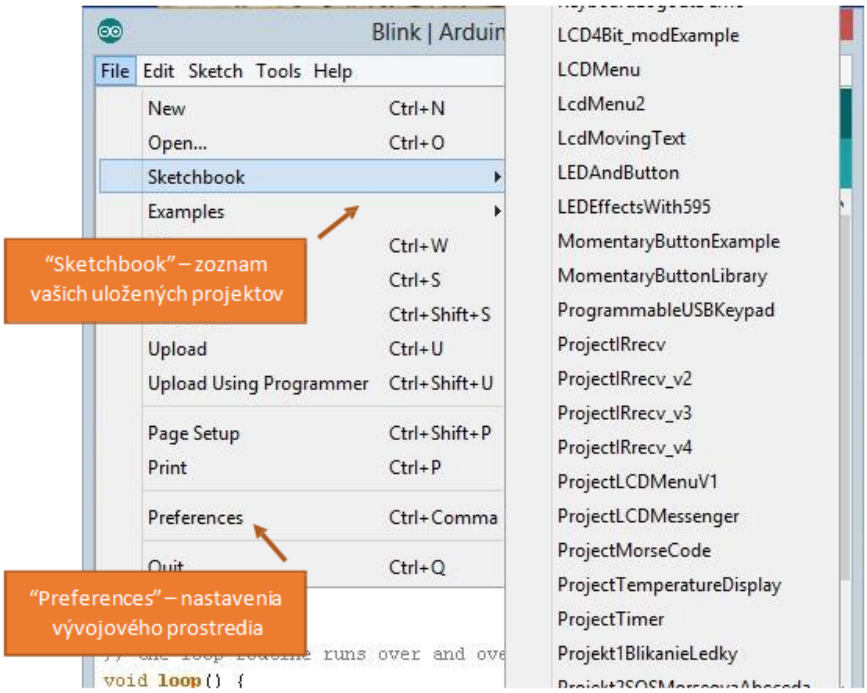


Obrázok 40 Arduino IDE, verzia 1.0.2

Na obrázku sú popísané jednotlivé tlačítka (zľava doprava) kontrola a kompilovanie (Verify), nahranie (Upload), Nová skica (New), Otvoriť (Open) a Uložiť (Save).

Sketchbook – kniha projektov

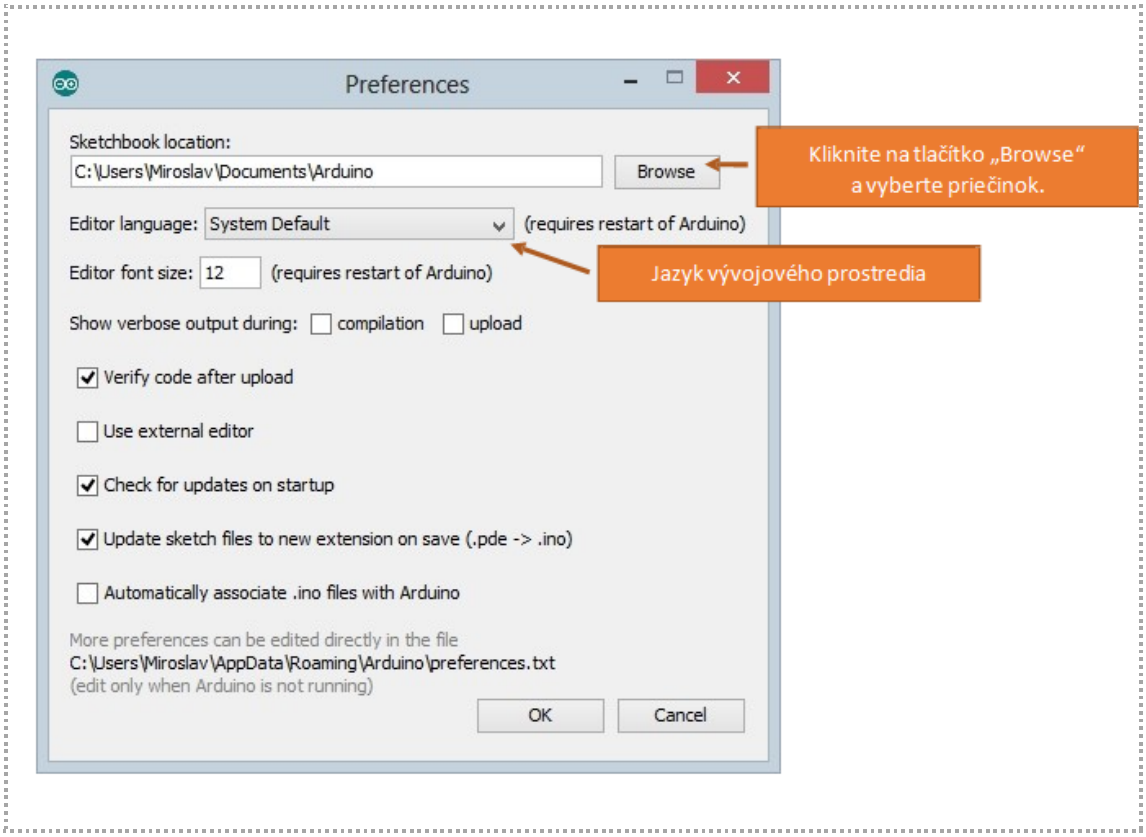
Na obrázku nižšie môžete vidieť položku z menu File a to „Sketchbook“ – tu sa vám budú zobrazovať všetky projekty, ktoré ste vytvorili a uložili.



Obrázok 41 Sketchbook

Štandardná cesta, kde sa ukladajú projekty je nasledovná:

Pre Windows Vista/7/8 C:\Users\<User>\Documents\Arduino\ alebo pre Windows XP C:\Documents and settings\<User>\My Documents\Arduino\ Ak si chceme nastaviť inú cestu pre ukladanie projektov, môžeme tak urobiť v nastaveniach vývojového prostredia File->Preferences. Vyberieme tlačítkom „Browse" priečinok, kde chceme aby sa nám ukladali projekty. Po nastavení novej cesty klikneme na tlačítko OK aby sa nastavenia uložili.



Obrázok 42 Sketchbook – nastavenia

Arduino Sketch

Každý program v Arduino IDE sa volá sketch, po slovensky teda skica alebo náčrt. Teraz si popíšeme štruktúru jednej takej skice:

Každá skica by mala mať minimálne dve metódy setup a loop. Metóda setup sa vykoná iba raz pri štarte programu alebo po stlačení tlačidla reset na Arduino doske.

Do tejto metódy sa vpisujú počiatočné nastavenia programu, napr. nastavenie vstupných a výstupných pinov, nastavenie sériového portu a iné (záleží na konkrétnom programe). Metóda loop sa vykonáva neustále dokola pokiaľ je samozrejme Arduino pripojené k zdroju napätia. Do tejto metódy sa píše všetko ostatné, čo má program vykonávať.

```
void setup() {
  // nastavenie pinov, seriového portu, atď
}

void loop() {
  // sem sa vpíše hlavný kód programu, ktorý sa bude vykonávať dookola
}
```

5.5 Blikač s použitím jednej LED a funkcie Delay ()

Ak náhodou nevlastníte žiadne zariadenie Arduino, ale aj napriek tomu si chcete vyskúšať prácu s týmto mikrokontrolérom, všetky uvedené príklady je možné otestovať pomocou on-line simulátora, ktorý nájdete na adrese: https://123d.circuits.io.

Na zostrojenie jednoduchého blikača z jednej LED diódy je potrebný nasledovný hardvér:

- Arduino.
- LED.
- Rezistor.

VÝKLAD

Voľba Arduina je v tomto prípade jasná – vystačíme si s Arduino Uno, alebo Nano. V prípade, ak by sme chceli merať i VA (Volt-Ampérovú) charakteristiku, môžeme použiť univerzálne Arduino DUE.

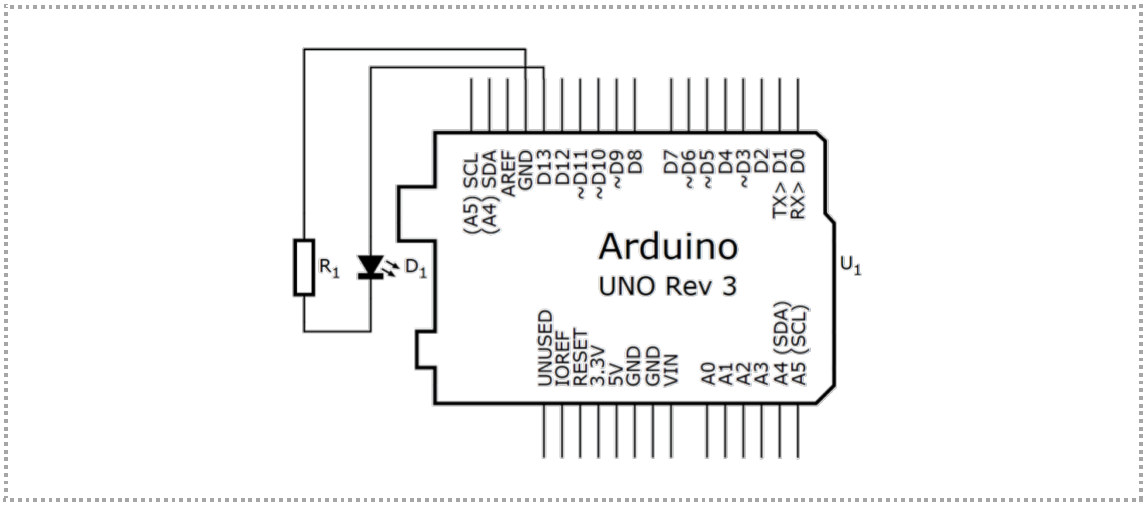
Všeobecná schéma zapojenia je zobrazená na nasledovnom obrázku **Chyba! Nenašiel sa žiaden zdroj odkazov.**, pričom na zapojenie blikač pozostávajúceho iba z jednej LED nám postačuje digitálny výstup z pin číslo 13 (LED_BUILTIN). Veľkosť elektrického odporu pri rezistore volíme na základe výpočtu:

$\mathbf{U_{zdroja}}$ - napájacie napätie zdroja ku ktorému chceme LED pripojiť (5V dodávaných z Arduina),

$\mathbf{U_{LED}}$ - napájacie napätie pre LED (zistíme z katalógu od výrobcu),

$\mathbf{I_{LED}}$ - prúd LED (zistíme z katalógu od výrobcu).

Ak by sme pre blikač použili červenú LED (najčastejšie používaná ako výstražná), tak štandardné napätie podľa výrobcu je ULED=1,9V a prúd ILED=20mA (0,02A) . Potom po dosadení do vzťahu dostaneme hodnotu elektrického odporu pre rezistor R=155Ω (ohmov).



Obrázok 43 Schéma zapojenia Arduino a Led

CVIČENIE – POUŽITE!

Na prepísanie kódu použite program Arduino IDE pre jeho naprogramovanie

```
INT LEDPIN = 13;

VOID SETUP() {

PINMODE(LEDPIN, OUTPUT);

}

VOID LOOP() {

DIGITALWRITE(LEDPIN, HIGH);

DELAY(1000);

DIGITALWRITE(LEDPIN, LOW);

DELAY(1000);

}
```

ZHRNUTIE

Prvý riadok programu je určený na inicializáciu pinu pre LED. Týmto krokom vlastne dokážeme prideliť akýkoľvek digitálny pin pre LED. Digitálny pin prideľujeme z dôvodu, že na rozdiel od analógového pracujeme s hodnotou logickej nuly a jednotky, čo predstavuje dve úrovne napätia LOW (LED nesvieti) a HIGH (LED svieti). Druhý riadok programu predstavuje metóda setup(), ktorá sa vykoná iba raz a to pri spustení programu alebo pri aktivovaní tlačidla RESET. V podstate ide o pridelenie módu pre pin, ktorý je vyhradený LED dióde. Aby sa pravidelne striedali hodnoty úrovne napätia LOW a HIGH, použili sme v programe slučku, teda jednoduché opakovanie. Toto opakovanie je vždy realizované po uplynutí určitého časového okamihu. V našom prípade tento časový okamih predstavuje hodnotu 1000, čo je 1 sekunda. Čakanie v rámci tohto časového

okamihu sme zabezpečili príkazom delay (). Pre vyslanie úrovne napätia LOW alebo HIGH je určený príkaz digitalWrite, čo je vlastne zápis na digitálny port, na ktorom je umiestnená LED.

5.6 Striedavý blikač s použitím dvoch LED a funkcie Delay ()

Ako sme uviedli vyššie, pre pripojenie LED môžeme použiť akýkoľvek digitálny pin.

VÝKLAD

V prípade, že chceme zostrojiť striedavý blikač a využiť okrem pinu číslo 13 aj pin číslo 7, program v Arduino IDE môžeme upraviť nasledovne:

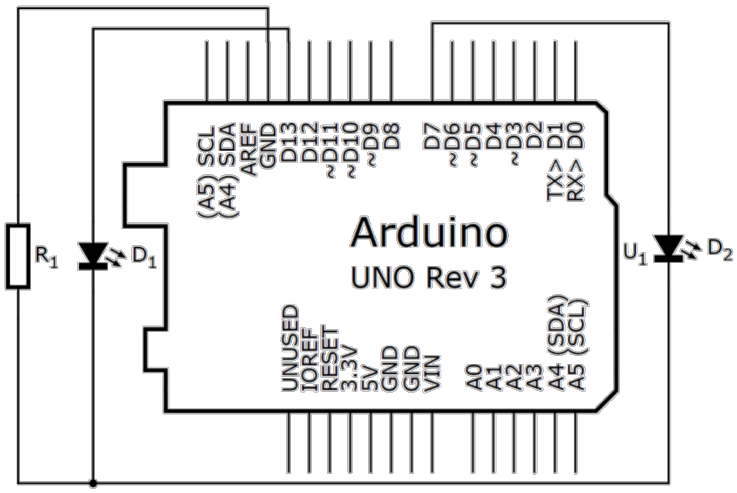
CVIČENIE – POUŽITE!

Použite v programe Arduino IDE kód, ktorý je uvedený nižšie.

```
INT LEDPIN1 = 13;
INT LEDPIN2= 7;
VOID SETUP() {
PINMODE(LEDPIN1, OUTPUT);
PINMODE(LEDPIN2, OUTPUT);
}
VOID LOOP() {
DIGITALWRITE(LEDPIN1, HIGH);
DELAY(1000);
DIGITALWRITE(LEDPIN1, LOW);
DELAY(1000);
DIGITALWRITE(LEDPIN2, HIGH);
DELAY(1000);
DIGITALWRITE(LEDPIN2, LOW);
DELAY(1000);
}
```

ZHRNUTIE

Schematické zapojenie je potom nasledovné:



Obrázok 44 Schéma zapojenia Arduino a dvoch Led

5.7 Zapojenie svetelného hada (postupné rozsvietenie a zhasnutie Led v rade)

VÝKLAD

Pre prípad takéhoto zapojenia nám postačuje použiť 5 LED diód a rozšíriť predchádzajúcu verziu programu. Avšak netreba zabúdať na to, že funkcia delay(); by v tomto prípade trvala veľmi dlho (1 sekundu zasvietenie, 1 sekundu zhasnutie každej LED). Z tohto dôvodu sme čas v programe upravili na hodnotu 100, čo má za následok postupné rozsvietenie a zhasnutie každej LED v poradí.

CVIČENIE – POUŽITE!

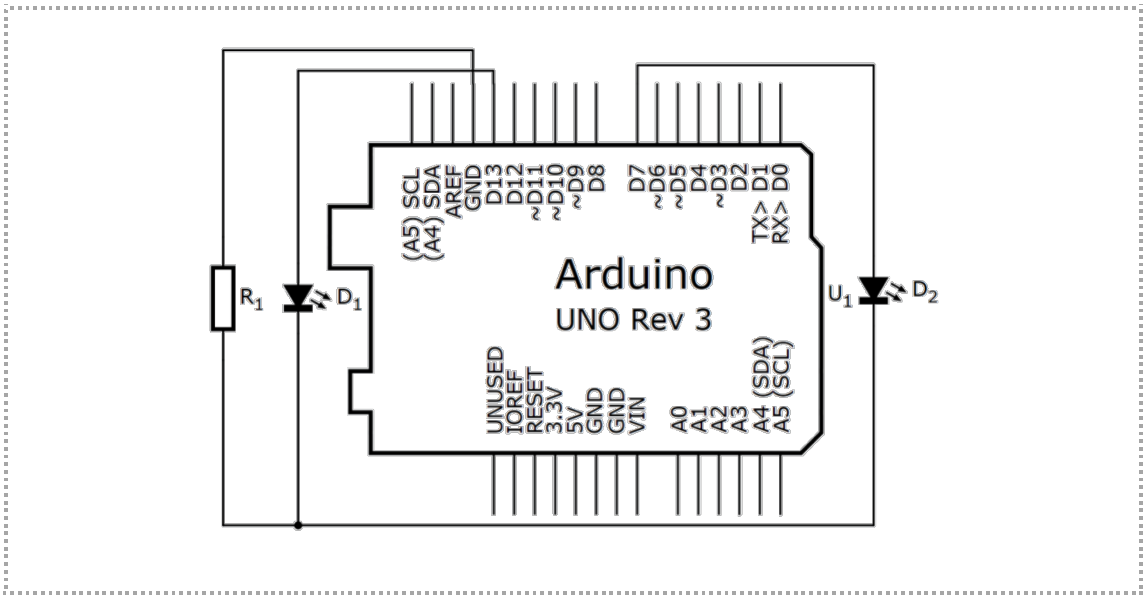
Použite v programe Arduino IDE kód, ktorý je uvedený nižšie

ZDROJOVÝ KÓD:

```
INT LED1 = 13;

INT LED2 = 7;

INT LED3 = 6;

INT LED4 = 5;

INT LED5 = 4;
```

```
VOID SETUP() {
  PINMODE(LED1, OUTPUT);
  PINMODE(LED2, OUTPUT);
  PINMODE(LED3, OUTPUT);
  PINMODE(LED4, OUTPUT);
  PINMODE(LED5, OUTPUT);
}
VOID LOOP() {
  DIGITALWRITE(LED1, HIGH);
  DIGITALWRITE(LED1, LOW);
  DELAY(100);
  DIGITALWRITE(LED2, HIGH);
  DIGITALWRITE(LED2, LOW);
  DELAY(100);
  DIGITALWRITE(LED3, HIGH);
  DIGITALWRITE(LED3, LOW);
  DELAY(100);
  DIGITALWRITE(LED4, HIGH);
  DIGITALWRITE(LED4, LOW);
  DELAY(100);
  DIGITALWRITE(LED5, HIGH);
  DIGITALWRITE(LED5, LOW);
  DELAY(100);
}
```

Výsledné schematické zapojenie svetelného hada je:



Obrázok 45 Schéma zapojenia Arduino – svetelný had

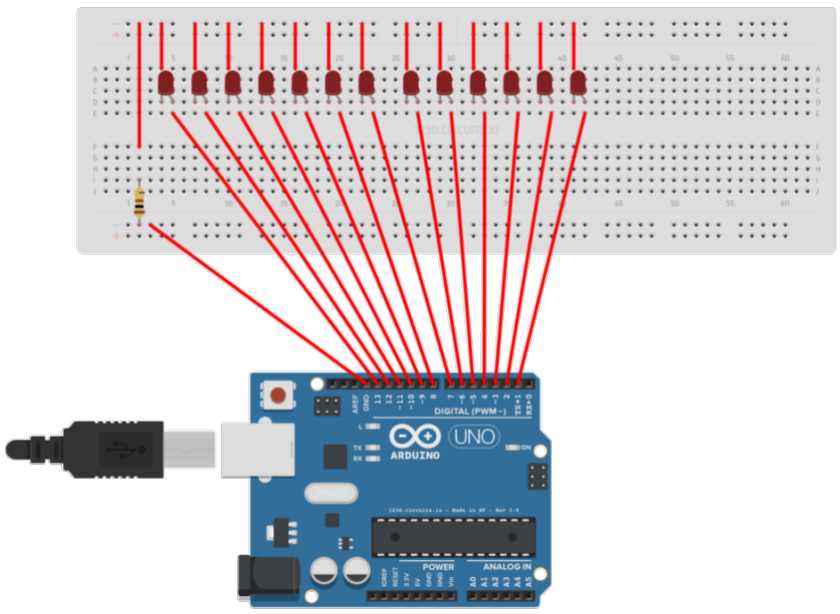
CVIČENIE – ANALYZUJTE!

Po prečítaní kódu je nám jasné, že takto zapísaný program je síce funkčný, ale z hľadiska zápisu nejde o príliš optimalizovanú časť kódu. Keďže niektoré časti kódu sa opakujú, je možné ich nahradiť napríklad cyklom:

```
BYTE OUTPUTPINS[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};

VOID SETUP()

{

FOR(INT I=0; I<13;I++)

{

PINMODE(OUTPUTPINS[I], OUTPUT);

DIGITALWRITE(OUTPUTPINS[I], LOW);

}

}

VOID LOOP()

{

FOR(INT I=12; I>0; I--)

{

DIGITALWRITE(OUTPUTPINS[I], HIGH);
```

```
DELAY(50);
DIGITALWRITE(OUTPUTPINS[I], LOW);
}
FOR(INT I=0; I<12; I++)
{
DIGITALWRITE(OUTPUTPINS[I], HIGH);
DELAY(50);
DIGITALWRITE(OUTPUTPINS[I], LOW);
}
}
```

Na nasledujúcom obrázku uvádzame reálne zapojenie (kvôli prehľadnosti) LED svetelného hada, pričom sú použité piny 1-13 a LED diódy svietia z ľavej strany do pravej a potom opačne.



Obrázok 46 Schéma zapojenia Arduino – svetelný had (13 zapojených Led s využitím cyklu for)

ZHRNUTIE

Prvým príkazom v programe inicializujeme piny 1-13. Metódou setup() nastavujeme piny ako digitálne výstupy. V tejto metóde sme taktiež nastavili hodnotu LOW (logická nula) pre všetky piny 1-13. Metóda setup() sa vykoná pri prvom štarte programu, teda všetky LED budú pri spustení zhasnuté. Prvým cyklom postupne aktivujeme LED z ľavej strany do pravej a druhým cyklom zase z pravej strany do ľavej. V obidvoch cykloch postupne na každý pin vysielame hodnotu HIGH (log. 1) alebo LOW (log. 0). Medzi obidvoma stavmi je oneskorenie 50ms, čo spôsobuje stav bliknutia LED.

5.8 Ovládanie lasu Led tlačidlami – využitie podmienok

VÝKLAD

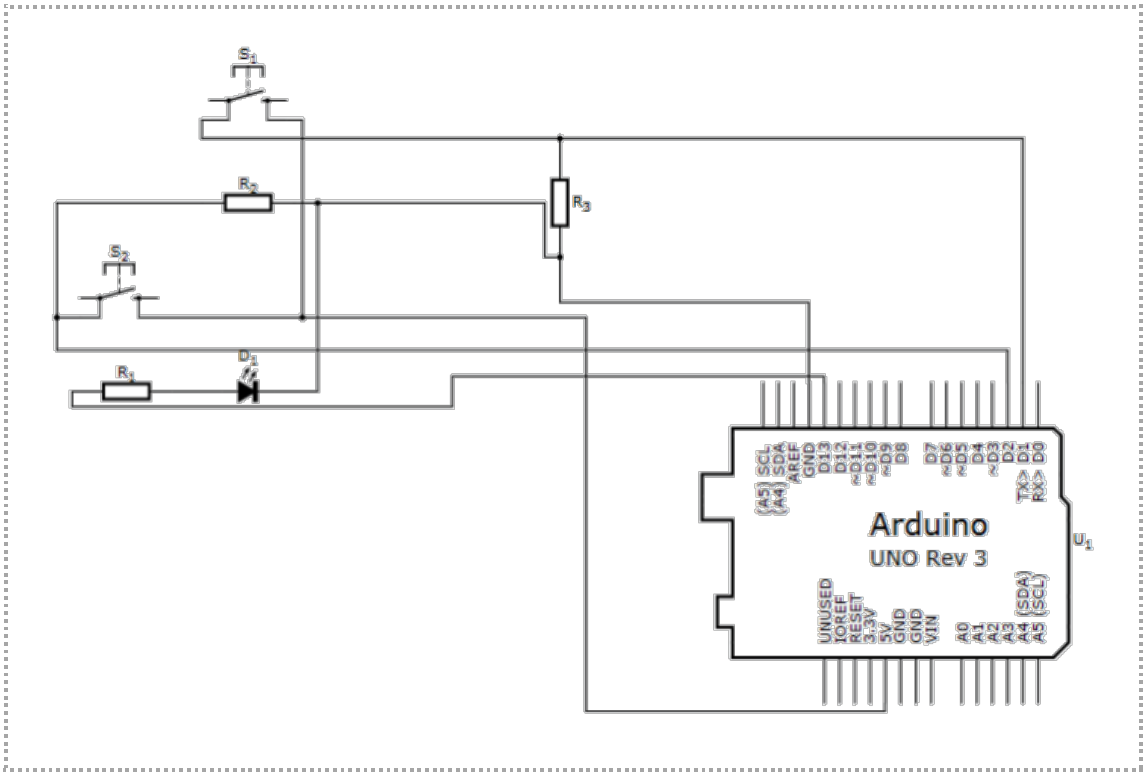
Dnes sa štandardnou súčasťou ovládania hlasitosti na rádiách (autorádiách), hi-fi vežiach alebo televízoroch stali tlačidlá, ktoré nahradili v minulosti využívané potenciometre. Potenciometer je súčiastka s meniteľným elektrickým odporom. Zmena veľkosti odporu mohla byť buď lineárna alebo exponenciálna. V prípade použitia tlačidiel je nutné určiť, či je stlačené tlačidlo, a ak áno, tak vykonať požadovaný príkaz (alebo sekvenciu). Toto nazývame stanovením podmienky:

if(podmienka){

príkaz

}

Zapojenie LED doplníme o dve tlačidlá, ktoré budú využívať taktiež digitálne vstupy (1 a 2).



Obrázok 47 Ovládanie zapínania a vypínania jasu Led dvoma tlačidlami

ZHRNUTIE

Program je tak jednoduchý, že sme použili formu komentárov, ktoré Vám uľahčia čítať kód. Základom celého programu je aktivovanie jednotlivých pinov, či už pre LED (13) alebo tlačidla (1 a 2), nastavenie premenných, ktoré uchovávajú stav tlačidla a nastavenie premennej pre jas (brightness) a premennej, ktorá určuje o akú veľkú časť hodnoty sa jas zmení (stepValue).

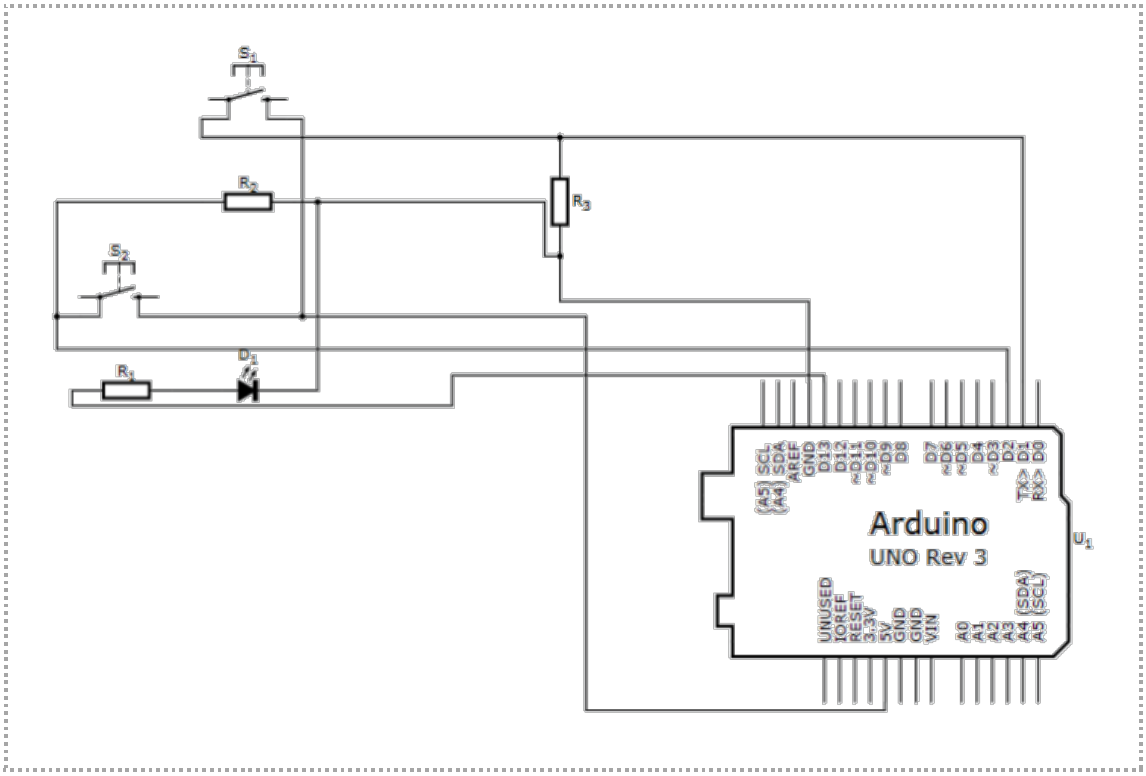
```
INT BUTTONPIN1 = 1; //ČÍSLO PINU PRE TLAČÍTKO_1 - ZVYŠOVANIE JASU

INT BUTTONPIN2 = 2; //ČÍSLO PINU PRE TLAČÍTKO_2 - ZNIŽOVANIE JASU

INT LEDPIN = 13; //ČÍSLO PINU PRE LED

INT BUTTONSTATE1 = 0; //PREMENNÁ UCHOVÁVAJÚCA STAV TLAČIDLA_1

INT BUTTONSTATE2 = 0; //PREMENNÁ UCHOVÁVAJÚCA STAV TLAČIDLA_2

INT BRIGHTNESS = 0; //JAS LED

INT STEPVALUE = 5; //VEĽKOSŤ KROKU NA NASTAVENIE JASU LED

VOID SETUP() {

 PINMODE(LEDPIN, OUTPUT); //NASTAVENIE PINU PRE LED AKO VÝSTUPNÝ

 PINMODE(BUTTONPIN1, INPUT); //NASTAVENIE PINU PRE TLAČÍTKO_1 AKO
VSTUPNÝ

 PINMODE(BUTTONPIN2, INPUT); //NASTAVENIE PINU PRE TLAČÍTKO_2 AKO
VSTUPNÝ

}

VOID LOOP(){

 BUTTONSTATE1 = DIGITALREAD(BUTTONPIN1); //NAČÍTAME STAV TLAČIDLA_1

 BUTTONSTATE2 = DIGITALREAD(BUTTONPIN2); //NAČÍTAME STAV TLAČIDLA_2

 ANALOGWRITE(LEDPIN, BRIGHTNESS);  //NASTAVENIE JASU LED

 IF (BUTTONSTATE1 == HIGH) {    //AK JE TLAČIDLO_1 STLAČENÉ, TAK ZVÝŠI
JAS

   BRIGHTNESS = BRIGHTNESS + STEPVALUE;

  }

IF (BUTTONSTATE2 == HIGH) {  //AK JE TLAČIDLO_2 STLAČENÉ, TAK ZNÍŽI
JAS

   BRIGHTNESS = BRIGHTNESS - STEPVALUE;

 }

}
```

5.9 Semafor bez použitia (automatický)a s použitím podmienok (po stlačení tlačidla)

VÝKLAD

Pri experimentovaní s LED diódami a kódom sa môžeme stretnúť na internete s rôznymi návodmi a jedným z nich je vytvorenie semafora, ktorý by pracoval automaticky – v určitom časovom okamihu by rozsvecoval postupne zelenú, oranžovú a červenú LED. Okrem mikrokontroléra Arduino sú potrebné nasledujúce súčiastky: LED v 3 farbách (zelená, červená

a oranžová, resp. žltá) a rezistory pripojené k LED s elektrickým odporom 220Ω . Zapojenie semaforu je veľmi jednoduché, vychádzame zo schémy blikača s jednou LED.



Obrázok 48 Automatický semafor

ZHRNUTIE

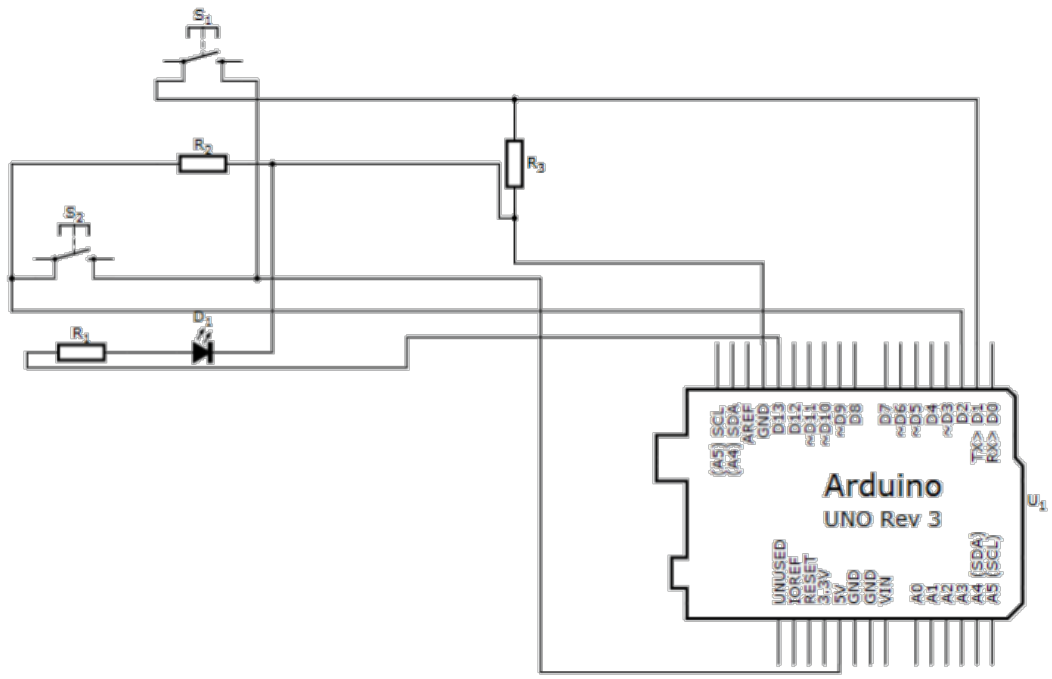
Program podľa zapojenia obsahuje funkciu delay(), ktorou sme zabezpečili čakanie semafora. LED sa rozsvecujú automaticky po uplynutí časového intervalu.

```
INT GREEN = 13;

INT YELLOW = 9;

INT RED = 6;

VOID SETUP(){

PINMODE(GREEN,OUTPUT);

PINMODE(YELLOW,OUTPUT);

PINMODE(RED,OUTPUT);

}

VOID LOOP(){

CHANGELIGHTS();              //FUNKCIA PRE ZMENU LED

DELAY(5000);

}
```

```
VOID CHANGELIGHTS(){
DIGITALWRITE(GREEN,LOW);     //ZELENÁ NESVIETI
DIGITALWRITE(YELLOW,HIGH);   //ŽLTÁ SVIETI
DELAY(3000);                 //STAV TRVÁ 3 SEKUNDY
DIGITALWRITE(YELLOW,LOW);    //ŽLTÁ NESVIETI
DIGITALWRITE(RED,HIGH);            //ČERVENÁ SVIETI
DELAY(5000);                 //STAV TRVÁ 5 SEKÚND
DIGITALWRITE(YELLOW,HIGH);   //ŽLTÁ SVIETI A SVIETI AJ ČERVENÁ
DELAY(2000);                 //STAV TRVÁ 2 SEKUNDY
DIGITALWRITE(YELLOW,LOW);    //ŽLTÁ NESVIETI
DIGITALWRITE(RED,LOW);       //ČERVENÁ NESVIETI
DIGITALWRITE(GREEN,HIGH);    //ZELENÁ SVIETI
}
```

CVIČENIE – POUŽITE!

Takýto semafor je možné zdokonaliť napríklad tlačidlom pre chodcov, ktoré by zabezpečovalo zmenu svetla. Použite tlačidlo a vyskúšajte nasledovný kód a schému



Obrázok 49 Semafor s tlačidlom pre chodcov

ZHRNUTIE

Program je doplnený o inicializáciu pinu, na ktoré je pripojené tlačidlo a o podmienku zabezpečujúcu určenie, či bolo tlačidlo aktivované alebo nie. V celom príklade v podstate ide o kombináciu doposiaľ používaných príkazov.

```
INT BUTTON = 2;                 //PRIPOJENIE TLAČIDLA NA PIN2

INT BUTTONVALUE = 0;            //NASTAVENIE TLAČIDLA NA HODNOTU 0
(NEZATLAČENÉ)

INT GREEN = 13;

INT YELLOW = 9;

INT RED = 6;

VOID SETUP(){

PINMODE(GREEN,OUTPUT);

PINMODE(YELLOW,OUTPUT);

PINMODE(RED,OUTPUT);

PINMODE(BUTTONVALUE,INPUT);

DIGITALWRITE(GREEN,HIGH);

}

VOID LOOP(){

BUTTONVALUE = DIGITALREAD(BUTTON); //NAČÍTANIE HODNOTY Z TLAČIDLA

IF (BUTTONVALUE == HIGH){    // TESTOVANIE HODNOTY

CHANGELIGHTS();                    //FUNKCIA PRE ZMENU LED

DELAY(5000);

}

}

VOID CHANGELIGHTS(){

DIGITALWRITE(GREEN,LOW);     //ZELENÁ NESVIETI

DIGITALWRITE(YELLOW,HIGH);   //ŽLTÁ SVIETI

DELAY(3000);                       //STAV TRVÁ 3 SEKUNDY

DIGITALWRITE(YELLOW,LOW);          //ŽLTÁ NESVIETI

DIGITALWRITE(RED,HIGH);            //ČERVENÁ SVIETI

DELAY(5000);                       //STAV TRVÁ 5 SEKÚND

DIGITALWRITE(YELLOW,HIGH);         //ŽLTÁ SVIETI A SVIETI AJ ČERVENÁ
```

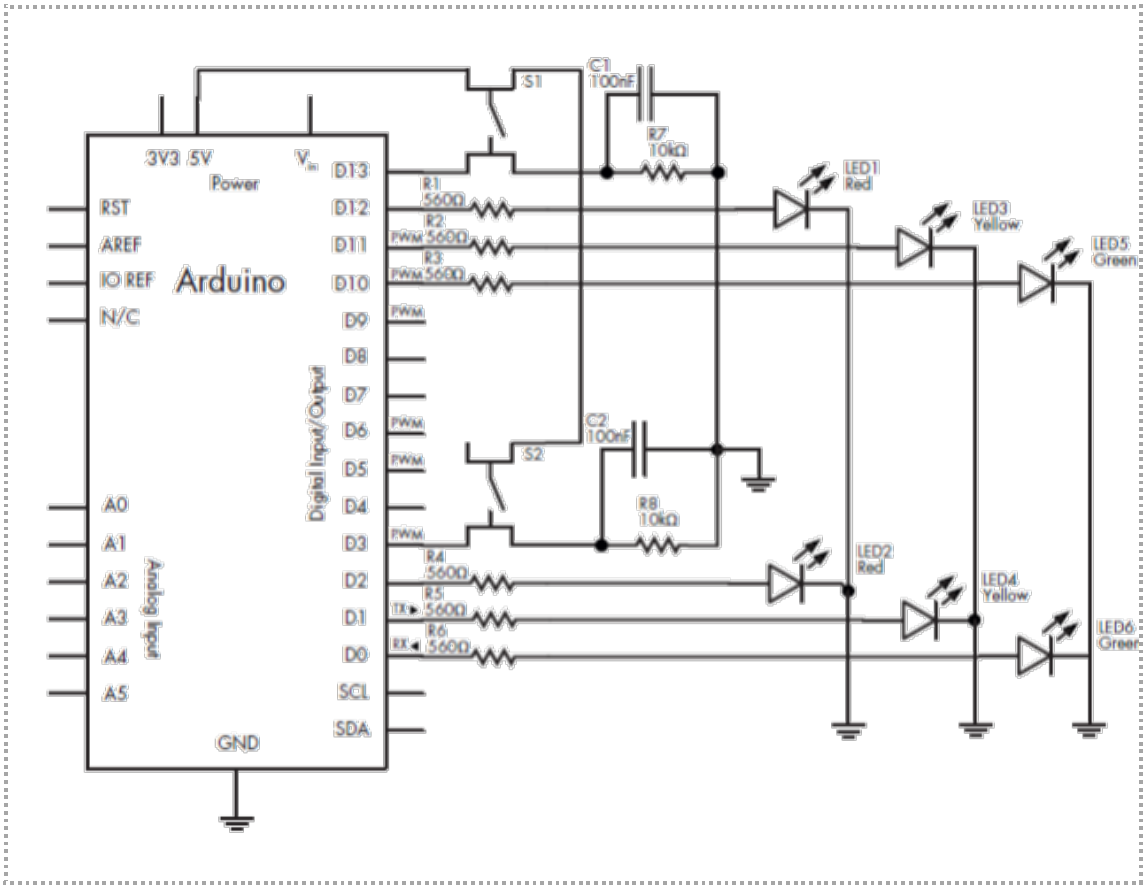
```
DELAY(2000);                    //STAV TRVÁ 2 SEKUNDY
DIGITALWRITE(YELLOW,LOW);       //ŽLTÁ NESVIETI
DIGITALWRITE(RED,LOW);      //ČERVENÁ NESVIETI
DIGITALWRITE(GREEN,HIGH);   //ZELENÁ SVIETI
}
```

CVIČENIE – POUŽITE!

Okrem predchádzajúcich komponentov použite aj pasívny prvok kondenzátor, ktorý slúži ako pamäťový prvok. Cieľom cvičenia je vytvoriť schému a program, ktorý bude riadiť premávku v dvoch smeroch.

VÝKLAD

A teraz niečo trochu zložitejšie... Predstavte si, že sa opravuje cesta na nejakom veľmi dlhom a kľukatom moste v neprehľadnej dlhej zákrute. Aby bolo možné bezpečne riadiť premávku, používa sa semafor ako na jednej tak i na druhej strane opravovanej cesty. Semafor je možné realizovať viacerými spôsobmi. Povedzme časovačom, potom tlačidlom alebo fotobunkou. To znamená, že ak napríklad v noci príde ku kritickému úseku automobil, fotobunka v podobe fotodiódy ho zosníma a upraví cyklus prepínania semaforu, aby zbytočne nedochádzalo k čakaniu. Prípadne je možné využívať spomínané tlačidlo, kde chodci po stlačení tlačidla sami upravujú cyklus, ako dlho bude svietiť červená. V našom prípade sme namodelovali situáciu, kedy sme použili takéto tlačidla na jednej i druhej strane pre riadenie činnosti semafora. Schéma zapojenia je rozšírená o kondenzátor – ten slúži na uchovanie stavu a teda nepriamo určuje dĺžku trvania časového okamihu pre príkaz delay().



Obrázok 50 Semafor s tlačidlami pre zmenu činnosti

ZHRNUTIE

V predchádzajúcich príkladoch sme si ukázali, že definovanie pinov môže prebiehať rôznymi spôsobmi. Jeden z menej tradičných spôsobov, ako definovať piny si ukážeme aj teraz. Pozor (!), ide o veľmi dlhý program:

```
// DEFINOVANIE PINOV PRE PRIPOJENIE TLAČIDIEL A LED:

#DEFINE WESTBUTTON 3

#DEFINE EASTBUTTON 13

#DEFINE WESTRED 2

#DEFINE WESTYELLOW 1

#DEFINE WESTGREEN 0

#DEFINE EASTRED 12

#DEFINE EASTYELLOW 11

#DEFINE EASTGREEN 10

#DEFINE YELLOWBLINKTIME 500 // 0.5 SEKUNDOVÉ BLIKANIE ŽLTEJ LED

BOOLEAN TRAFFICWEST = TRUE; // WEST = TRUE, EAST = FALSE
```

```
INT FLOWTIME = 10000; // ČAS PONECHANÝ NA PLYNULOSŤ PREMÁVKY

INT CHANGEDELAY = 2000; // DOBA MEDZI ZMENOU FARBY

VOID SETUP()

{

// SETUP DIGITAL I/O PINS

PINMODE(WESTBUTTON, INPUT);

PINMODE(EASTBUTTON, INPUT);

PINMODE(WESTRED, OUTPUT);

PINMODE(WESTYELLOW, OUTPUT);

PINMODE(WESTGREEN, OUTPUT);

PINMODE(EASTRED, OUTPUT);

PINMODE(EASTYELLOW, OUTPUT);

PINMODE(EASTGREEN, OUTPUT);

// NASTAVUJE POČIATOČNÝ STAV LED, ZÁPADNÁ STRANA SVIETI NA ZELENO

DIGITALWRITE(WESTRED, LOW);

DIGITALWRITE(WESTYELLOW, LOW);

DIGITALWRITE(WESTGREEN, HIGH);

DIGITALWRITE(EASTRED, HIGH);

DIGITALWRITE(EASTYELLOW, LOW);

DIGITALWRITE(EASTGREEN, LOW);

}

VOID LOOP()

{

IF ( DIGITALREAD(WESTBUTTON) == HIGH ) // POŽIADAVKA ZÁPADU NA VÝCHOD,
ABY BOLA PLYNULÁ PREMÁVKA

{

IF ( TRAFFICWEST != TRUE )

// POKRAČOVAŤ LEN V PRÍPADE, AK JE PREMÁVKA REALIZOVANÁ V OPAČNOM
SMERE (NA VÝCHOD)

{

TRAFFICWEST = TRUE; // ZMENA DOPRAVY ZO ZÁPADU NA VÝCHOD

DELAY(FLOWTIME); // POSKYTNUTIE ČASU NA PLYNULÝ DOPRAVNÝ TOK
```

```
DIGITALWRITE(EASTGREEN, LOW); // ZMENA NA VÝCHODNEJ STRANE ZO ZELENEJ
NA ŽLTÚ A POTOM NA ČERVENÚ

DIGITALWRITE(EASTYELLOW, HIGH);

DELAY(CHANGEDELAY);

DIGITALWRITE(EASTYELLOW, LOW);

DIGITALWRITE(EASTRED, HIGH);

DELAY(CHANGEDELAY);

FOR ( INT A = 0; A < 5; A++ ) // BLIKANIE ŽLTÉHO SVETLA

{

DIGITALWRITE(WESTYELLOW, LOW);

DELAY(YELLOWBLINKTIME);

DIGITALWRITE(WESTYELLOW, HIGH);

DELAY(YELLOWBLINKTIME);

}

DIGITALWRITE(WESTYELLOW, LOW);

DIGITALWRITE(WESTRED, LOW); // ZMENA NA ZÁPADNEJ STRANE Z ČERVENEJ NA
ZELENÚ

DIGITALWRITE(WESTGREEN, HIGH);

}

}

IF ( DIGITALREAD(EASTBUTTON) == HIGH ) // POŽIADAVKA ZMENY VÝCHODNEJ
STRANY NA ZÁPADNÚ

{

IF ( TRAFFICWEST == TRUE )

// POKRAČOVAŤ LEN V PRÍPADE, AK JE PREMÁVKA REALIZOVANÁ V OPAČNOM
SMERE (NA ZÁPAD)

{

TRAFFICWEST = FALSE; // ZMENA DOPRAVY Z VÝCHODU NA ZÁPAD

DELAY(FLOWTIME); // POSKYTNUTIE ČASU NA PLYNULÝ DOPRAVNÝ TOK

DIGITALWRITE(WESTGREEN, LOW);

// ZMENA NA ZÁPADNEJ STRANE ZO ZELENEJ NA ŽLTÚ A POTOM NA ČERVENÚ

DIGITALWRITE(WESTYELLOW, HIGH);

DELAY(CHANGEDELAY);
```

```
DIGITALWRITE(WESTYELLOW, LOW);

DIGITALWRITE(WESTRED, HIGH);

DELAY(CHANGEDELAY);

FOR ( INT A = 0 ; A < 5 ; A++ ) // BLIKANIE ŽLTÉHO SVETLA

{

DIGITALWRITE(EASTYELLOW, LOW);

DELAY(YELLOWBLINKTIME);

DIGITALWRITE(EASTYELLOW, HIGH);

DELAY(YELLOWBLINKTIME);

}

DIGITALWRITE(EASTYELLOW, LOW);

DIGITALWRITE(EASTRED, LOW); // ZMENA NA VÝCHODNEJ STRANE Z ČERVENEJ
NA ZELENÚ

DIGITALWRITE(EASTGREEN, HIGH);

}

}

}
```

Ak sa Vám program nechce prepisovať, môžete si jeho elektronickú verziu stiahnuť aj zo stránky: http://pop.h-cdn.co/assets/cm/15/06/54cfd847b77c5_-_arduino_project5.pdf.

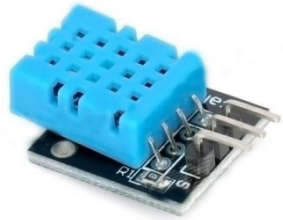
Program pracuje tak, že modeluje prípad využitia semaforu v neprehľadnej zákrute. Rozdelili sme premávku od západu na východ, pričom vozidlá ktoré prichádzajú zo západu majú nastavené svetlo semafora na zelenú a automobily na východnej strane zatiaľ musia čakať (svieti červená). Keď sa vozidlo blíži k mostu (namiesto fotobunky používame tlačidlo), semafor ho detekuje, systém sám prepne svetlo na opačnej strane zo zelenej na oranžovú (žltú) a potom na červenú. Čaká určitý časový okamih, aby bolo umožnené vozidlám, ktoré prešli ešte na žltú, bezpečne sa dostať na druhú stranu mosta. Na druhej strane mosta však už začne postupne blikať žltá LED pre upozornenie vodičov, aby sa pripravili na vyštartovanie. Po určitom časovom okamihu sa stav zo žltej LED prepne na zelenú. Proces sa potom postupne opakuje, podľa toho, z ktorej strany postupne prichádzajú (a v akom časovom okamihu) vozidlá. V programe sú ošetrené všetky možné stavy. Aj keď program vyzerá na prvý pohľad príliš zložito, v podstate ide o rozšírenie myšlienky predchádzajúceho prípadu, kedy sme pri semafore využili jedno tlačidlo.

5.10 Meranie teploty a vlhkosti

Ak máme k dispozícii LCD panel, stačí nám dokúpiť vhodný senzor na meranie teploty a vlhkosti. Potom je možné vytvoriť nielen jednoduchý domáci teplomer, ale najmä termostat, ktorý vy spínal napríklad pomocou relé iný obvod – ventilátor v klimatizácii a podobne.

VÝKLAD

Najskôr si popíšeme jednotlivé dostupné senzory na meranie teploty a vlhkosti. Najjednoduchší typ digitálneho modulu, s ktorým sa môžeme stretnúť v rôznych eshopoch je DHT-11. Je to senzor využívajúci napätie 3-5V, to znamená že ho môžeme bez akýchkoľvek problémov pripojiť k mikrokontroléru Arduino. Merací rozsah vlhkosti je 20-90% RH. Presnosť však nie je jeho silnou stránkou, keďže v prípade vlhkosti presnosť dosahuje podľa údajov výrobcu ± 5% RH a pri teplote je presnosť okolo ±2%. Uvádzame okolo, pretože v praxi sme sa stretli s oveľa väčšou odchýlkou. Teplotu dokáže odmerať v rozsahu 0-60°C. Jeho výhodou je digitálny výstup, a teda výstupnú teplotu zobrazuje aj desatinnými číslami (https://arduino8.webnode.cz/).

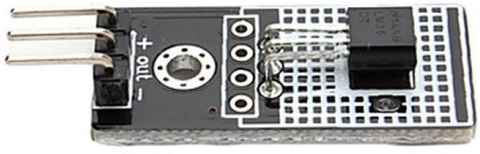


Obrázok 51 Senzor DHT - 11

Druhým typom digitálneho senzoru je modul DS18B20. Ide o modul, ktorým sa dá merať iba teplota, ale na rozdiel od predchádzajúceho typu ide o poloprofesionálny senzor, keďže dokáže merať teplotu v špecifickom rozsahu -55°C až + 124°C, čo už znie veľmi zaujímavo. Presnosť je pritom jeho veľmi silnou stránkou, veď v rozsahu -10°C až +85°C sa dopustí chyby približne ±0,5°C. Dokáže zobrazovať teplotu so zaokrúhlením na desatiny.

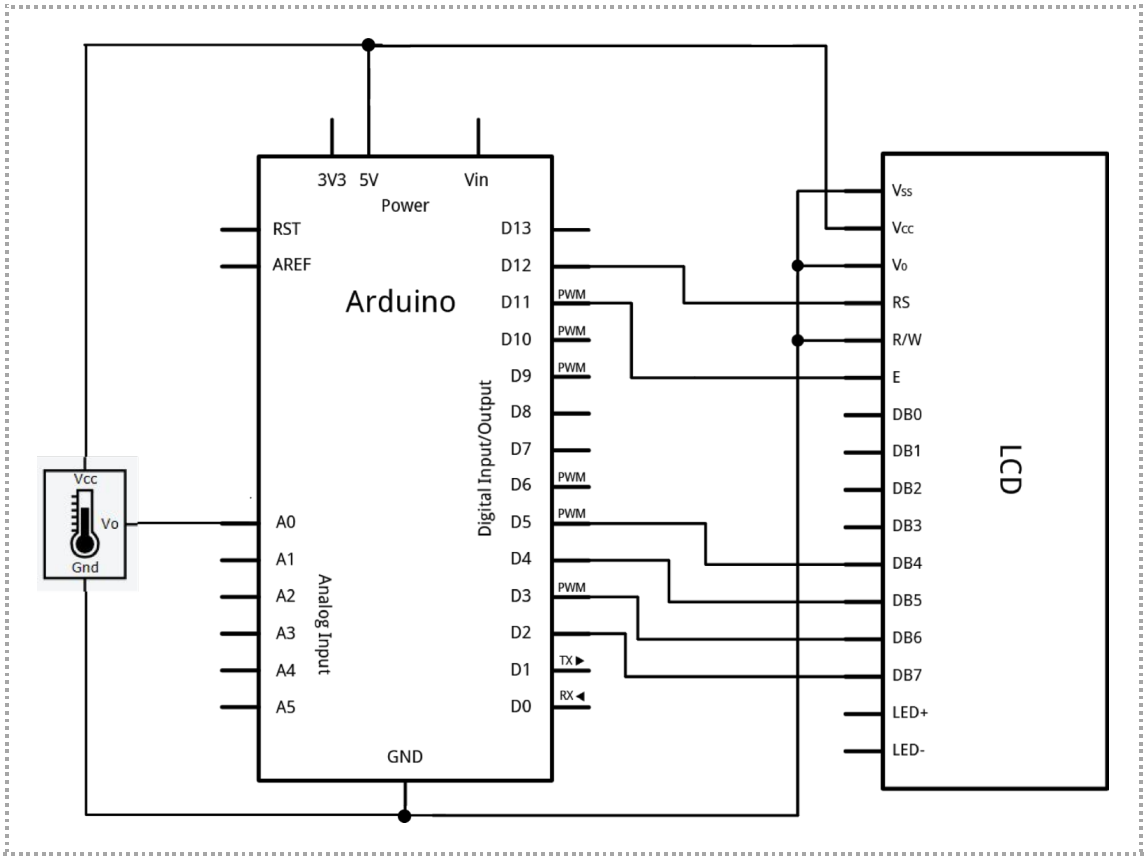
Obrázok 52 Senzor DS18B20

Posledným typom je analógový senzor LM35D, ktorý nevyniká presnosťou (±10%), ale pre účely skúšania práce s mikrokontrolérom plne postačuje.



Obrázok 53 Senzor LM35D

K mikrokontroléru Arduino sme pripojili LCD panel a snímač teploty, ktorý má 3 vývody. Dva sú určené na napájanie (5V a uzemnenie) a tretí vývod je označovaný ako Vout. Snímač sníma teplotu okolia, pričom hodnota je poskytovaná vo forme napätia pomocou analógového vstupu. Preto v programe musíme zrealizovať prevod analógovej hodnoty na teplotu.



Obrázok 54 Zapojenie mikrokontroléra Arduino pre meranie teploty senzorom LM35D

CVIČENIE – POUŽITE!

Podľa schémy na obrázku 55 skúste v prostredí Arduino IDE naprogramovať mikrokontrolér tak aby zobrazoval na LCD panely teplotu.

ZHRNUTIE

Vypísanie hodnoty nameranej teploty môže byť aj v jednom riadku, avšak ak v prvom riadku uvedieme príliš dlhý popis (ako aj v našom prípade), je vhodné nameraný číselný údaj umiestniť do druhého riadku LCD panela.

```
#INCLUDE <LIQUIDCRYSTAL.H>   // KNIŽNICA PRE LCD

LIQUIDCRYSTAL LCD(12, 11, 5, 4, 3, 2);  // NAČÍTANIE PINOV LCD, KTORÉ
SA BUDÚ POUŽÍVAŤ

FLOAT TEMPC; // DEKLAROVANIE PREMENNÝCH
```

```
INT TEMPPIN = 0; // NAČÍTANIE PINU PRE SENZOR

VOID SETUP(){

  LCD.BEGIN(16, 2);        // NASTAVENIE LCD NA POČET STĹPCOV A RIADKOV

  LCD.PRINT("TEPLOTA"); // VYPÍSANIE "TEPLOTA" NA DISPLEJI

  LCD.SETCURSOR(0, 1);

}

VOID LOOP(){

  TEMPC = ANALOGREAD(TEMPPIN); // NAČÍTANIE HODNOTY ZO SENZORA

  TEMPC = (5.0 * TEMPC * 100.0)/1024.0;  //PREVOD ANALÓGOVEJ HODNOTY
NA TEPLOTU

  LCD.SETCURSOR(6, 1); // ZOBRAZENIE HODNOTY NA LCD PANELY V DRUHOM
RIADKU

  LCD.PRINT(TEMPC,1);

  LCD.PRINT("°C");

  DELAY(1000);    // POČKÁ 1 S A POTOM OPÄŤ NAČÍTA HODNOTU

}
```

Podobným spôsobom môžeme odmerať aj vlhkosť, ibaže potrebujeme na to senzor DHT-11. Čo sa týka zapojenia, to sa nemení. Zmena je iba v programe, kde je potrebné importovať knižnicu pre senzor1, ktorá zabezpečuje aj potrebný prevod analógovej hodnoty na teplotu a vlhkosť :

```
#INCLUDE  <LIQUIDCRYSTAL.H>    // KNIŽNICA PRE LCD

#INCLUDE <DHT11.H>    // KNIŽNICA PRE DHT11

LIQUIDCRYSTAL LCD(12, 11, 5, 4, 3, 2); NAČÍTANIE PINOV LCD, KTORÉ SA
BUDÚ POUŽÍVAŤ

DHT11 MOJSENZOR; // VYTVORÍ OBJEKT DHT11 S NÁZVOM MOJSENZOR

VOID SETUP() {

  LCD.BEGIN(20, 4); // NASTAVÍ TYP DISPLEJA NA 20 ZNAKOV A 4 RIADKY
(MÔŽETE UPRAVIŤ PODLA SEBA)

}

VOID LOOP() {
```

```
  MOJSENZOR.READ(0); // NAČÍTA ÚDAJE ZO SENZORA DTH11 PRIPOJENÉHO NA
PIN 0

  INT HODNOTA = MOJSENZOR.TEMPERATURE;  // NAČÍTA HODNOTU Z A0

  INT VLHKOST = MOJSENZOR.HUMIDITY;

  LCD.SETCURSOR(0,2); // NASTAVÍ KURZOR NA TRETÍ RIADOK A PRVÝ ZNAK

  LCD.PRINT("T = ");  // VYPÍŠE TEXT T =

  LCD.PRINT(HODNOTA); // VYPÍŠE HODNOTU TEPLOTY

  LCD.PRINT(" OC");   // VYPÍŠE OC

  LCD.SETCURSOR(0,3); // NASTAVÍ KURZOR NA ŠTVRTÝ RIADOK A PRVÝ ZNAK

  LCD.PRINT("V = ");  // VYPÍŠE TEXT V =

  LCD.PRINT(VLHKOST); // VYPÍŠE HODNOTU VLHKOSTI

  LCD.PRINT(" %   "); // VYPÍŠE %

  DELAY(1000);  // POČKÁ 1S

}
```

BIBLIOGRAFIA

Literatúra

AGRAWAL, N., et al. Practical Handbook Of Thin-Client Implementation. New Delhi: New Age International, 2005. 232 p. ISBN 978-8122416855

Agriculture, 2015. In. Compusoft : An international journal of advanced computer technology. - ISSN 2320-0790, Vol. 4, no. 2 (2015), p. 1492-1494.

Analýza možností 3D tlače [bakalárska práca] / Mikuláš Fúra ; Škol. Nadežda Čuboňová, . - Katedra automatizácie a výrobných systémov Fakulta strojnícka Žilinskej univerzity v Žiline. - Žilina; 2015. - 69 s.

Cyril Klimeš, Ildikó Pšenáková, Veronika Stoffová, Typi Universitatis Tyrnaviensis, 2017

HOOPES, John. 2009. Virtualization for Security. Burlington : Syngress Publishing, Inc., 2009. ISBN 13: 978-1-59749-305-5.

KOPRDA, Š.; MAGDIN, M. (2015). New Trends and Developments in Automation in Agriculture, 2015. In. Compusoft : An international journal of advanced computer technology. - ISSN 2320-0790, Vol. 4, no. 2 (2015), p. 1492-1494.

Lačezar Ličev, David Morkes, Computer Press

Náučný slovník elektrotechnický (4.zv.). Kybernetické systémy. ALFA, Bratislava 1987

Olivka Peter Studijní materiál pro předmět Architektury počítačů a paralelních systémů, Ostrava 2020

RULE, David a DITTNER, Rogier. 2007. The Best Damn Server Virtualization Book Period. Burlington : Syngress Publishing, 2007. ISBN 13: 978-1-59749-217-1.

Technológie dizajnu a výroby 3D modelov. [bakalárska práca] / Michal Marko ; Škol. Róbert Hudec, Patrik Kamencay. - Katedra telekomunikácií multimédií Fakulty elektrotechnickej Žilinskej univerzity v Žiline. - Žilina; 2013. - 64 s.,

Technologické zariadenie pre dokončovacie operácie produktov z technológie 3D tlače. [bakalárska práca] / Marián Stopka ; Škol. Rudolf Madaj, N/A Ing. Peter Bezák. - Katedra konštruovania a časti strojov Fakulty strojníckej Žilinskej univerzity v Žiline. - Žilina; 2013. - 61 s.,

Wheat, D: Arduino Internals, 2011

Wilkinson, B.: Computer Architecture. Design & performance. Prentice Halí Int. Ltd., London 1991

Online zdroje

Ako naprogramovať Arduino bez predchádzajúcich znalostí. Miro Božík [online]. 2013 [cit. 2015-11-05]. http://mirobozik.sk/Media/Default/ebooks/mirobozik-ako-napr-arduino.pdf

http://aplo.fiit.stuba.sk/aps/frames/generuj.php?id=12

http://ksvi.mff.cuni.cz/~holan/did/VojtechTuma/DropBox.html#more

https://melisko.webnode.sk/news/uchovavanie-informacii-kodovanie-/

https://tech.sme.sk/c/20128709/pat-sposobov-ako-si-bezpecne-zalohovat-data.html#ixzz5GJYt9snn

https://www.etrend.sk/technologie/superpocitace-su-sede-eminencie-kybernetickeho-veku.html

http://www.adlerka.sk/UserFiles/File/3D_tlac.pdf

https://www.grc.com/securable.htm

[PDF]XYZaker Software user manual

Lekce 6 - čidlo DHT11 - teplota, vlhkost. Arduino8.cz: Arduino, elektronika a vše okolo [online]. 2013 [cit. 2015-11-05]. Dostupné z: http://www.arduino8.cz/lekce-6-cidlo-dht11-teplota-vlhkost/

LiquidCrystal Library. Arduino.cc. [online]. 2014 [cit. 2015-11-05]. Dostupné z: http://arduinoliquidcrystal.readthedocs.org/en/latest/liquidcrystal.html

http://wh.cs.vsb.cz/mil051/images/d/d7/PAP_Vyu%C5%BEit%C3%AD_grafick%C3%BDch_karet_pro_obecn%C3%A9_v%C3%BDpo%C4%8Dty_(GPGPU)_(Petr_Hrub%C3%BD).pdf

http://poli.cs.vsb.cz/edu/arp/down/procrisc.pdf

https://www.fi.muni.cz/~kas/pv090/referaty/2013-jaro/virt.html

http://pk-info.spsepn.edu.sk/studium/ucebtext/tis/sw/bezpec/bezpec.pdf

https://gkmke.sk/informatika/4.rocnik/IOzariadenia.pdf

Schmotzer Milan: Architektúra počítačov http://maturitazinf.mrazovci.eu/

6 ÚVOD DO OPERAČNÝCH SYSTÉMOV

Operačný systém poskytuje prostredie, v ktorom sa vykonávajú programy. Operačný systém predstavuje súbor riadiacich programov, ktoré zabezpečujú najmä riadenie procesov, riadenie súborov, riadenie pamätí a riadenie periférnych zariadení.

CIEĽ

Čo budeme vedieť a čo budeme vedieť urobiť

- vysvetliť základné úlohy operačných systémov,
- vedieť pracovať v operačnom systéme Linux,
- vedieť vytvoriť jednoduché skripty v skriptovacom jazyku Bash.

V tejto kapitole sa naučíme aké základné úlohy a funkcie majú operačné systémy. Pomocou príkladov sa naučíme pracovať v operačnom systéme Linux a programovať jednoduché skripty v pomocou interpretera príkazov Bash.

Kľúčové slová

pamäť, operačná pamäť, disk, súbor, adresár, riadenie pamätí, riadenie súborov, riadenie procesov, riadenie periférnych zariadení

MOTIVÁCIA

V multiúlohových systémoch môžeme mať spustených viac programov (procesov). Všetky programy, ktoré máme spustené musia byť umiestnené v operačnej pamäti. Súbory ukladáme na disk, alebo iné pamäťové médiá. *Zabezpečuje operačný systém prácu periférnych zariadení?*

CVIČENIE – DISKUTUJTE!

Aké operačné systémy poznáte? Ktorý operačný systém je viac otvorený?

Môžeme sa pozrieť do systému, aké údaje si systém ukladá?

Poznáme príkazy, ktorými sa vieme pozrieť na údaje v systéme? Ako vieme programovať v systéme?

6.1 Hlavné úlohy a funkcie operačného systému

Operačný systém poskytuje prostredie, v ktorom sa vykonávajú programy. Operačný systém predstavuje súbor riadiacich programov, ktoré zabezpečujú najmä:

- riadenie procesov,
- riadenie súborov,
- riadenie pamätí,
- riadenie periférnych zariadení.

VÝKLAD

Jednou z úloh operačného systému je riadiť procesy v systéme. Zabezpečuje, aby správne bežali naše programy. Každý program je potrebné najskôr preložiť (skompilovať). Zjednodušene povedané, prekladom sa príkazov programovacieho jazyka stanú inštrukcie. Pri spustení programu sa inštrukcie zavedú do tzv. operačnej pamäti počítača.

Program je súbor uložený na disku, ktorý môžeme spustiť. Keď program spustíme, vykonáva určitú činnosť. Každý program, ktorý spustíme v systéme, vytvorí jeden alebo viac procesov. Proces je program v činnosti. Proces nadobúda stavy ako nový, pripravený, bežiaci, blokovaný a ukončený. V multiúlohových systémoch systém prepína procesy. Procesy čakajú na spracovanie v rade pripravených procesov. [1]

Riadenie súborov je tiež veľmi dôležité. Súbory ukladáme na disk, alebo iné externé pamäťové médiá. Súbory sa na disk ukladajú do blokov, ktoré nemusia nasledovať za sebou. Pomocou Prieskumníka, alebo iných nástrojov vieme zobraziť informácie o súboroch. Operačné systémy si uchovávajú informácie o súboroch v tzv. súborových systémoch. Súborové systémy obsahujú mená všetkých súborov, ich veľkosti, dátum a čas ich vytvorenia, resp. zmeny súboru, a prístupové práva. Okrem týchto informácií ukazujú na bloky údajov, v ktorých sú uložené všetky bloky súboru. [3].

Riadenie pamäti v operačnom systéme zodpovedá za to, že efektívne pracuje najmä s pamäťou RAM. Programy (procesy), ktoré chceme spracovávať pomocou procesora musia byť umiestnené v pamäti RAM. Systém si dynamicky ukladá aktuálne informácie o voľných úsekoch pamäte. Z radu pripravených procesov vyberie proces a hľadá vhodný úsek (segment pamäti) pre jeho umiestnenie. Pri riadení pamäte sa často používajú stránkované segmenty pamäte. Stránkovanie „rozdelí“ segment pamäte na rovnako veľké úseky tzv. stránky.

Počítač komunikuje s okolitým prostredím pomocou periférnych zariadení. Periférne zariadenia majú hardvérové časti a tiež softvérové ovládače. Operačný systém zodpovedá za riadenie periférnych operácií tak, že prijíma príkazy od používateľa a dáva povely ovládaču periférnych zariadení a následne periférnemu zariadeniu. [2]

Ak chceme porozumieť trochu detailnejšie ako operačný systém pracuje treba sa pozrieť ako si systém ukladá údaje, ktoré potrebujú riadiace algoritmy operačného systému. Keďže

operačný systém Linux má systémové údaje do určitej miery „otvorené“, pozrieme sa ako tento systém riadi procesy, resp. súbory. Preto sa v ďalších podkapitolách zameriame na základné príkazy Linuxu a uvedieme niekoľko príkladov v skriptovacom jazyku a interpreteri Bash.

6.2 Príkazy a práca v operačnom systéme Linux

V tejto podkapitole sa naučíme používať príkazy operačného systému Linux. Pri písaní syntaxe príkazov budeme dodržiavať určité pravidlá. [4]

Syntax príkazov:

ls	[-aClpR]	[subor]

Prvý reťazec znakov je príkaz. Príkaz vykoná určitú činnosť. V našom prípade príkaz `ls` (je podmaľovaný žltou farbou) zobrazí položky aktuálneho adresára. Za menom príkazu vždy nasleduje medzera. Hranaté zátvorky znamenajú, že parametre, ktoré sú v nich uvedené sú nepovinné.

Parametre príkazu, napríklad `–aClpR` (sú podmaľované zelenou farbou), začínajú znakom pomlčka (-). Parametre príkazov sa môžu používať samostatne, alebo viaceré naraz, pričom nezáleží na poradí zápisu.

`Subor` (je podmaľovaný modrou farbou) predstavuje objekt, nad ktorým chceme vykonať určitú operáciu (príkaz). Ak je daným objektom súbor, alebo adresár, treba k nemu špecifikovať cestu. Cesta sa neuvádza, ak sa objekt nachádza v aktuálnom adresári.

V príkazovom riadku systém vypisuje tzv. prompt. V našom prípade je to: `jskrinarova@labs:~$.` Prompt zvyčajne končí znakom $. Systém čaká, že za týmto znakom napíšeme príkaz. V príklade 1 sme systému zadali príkaz `ls`. Príkaz sme zadali bez parametrov.

CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz `ls.`

```
jskrinarova@labs:~$ ls
linux  neuro  priklady  ulozisko  zoznam1  zoznam2  zoznam3
jskrinarova@labs:~$
```

Obrázok 55

Príklad zadávania príkazov v Linuxe

Systém príkaz vykoná príkaz `ls` a vypíše zoznam položiek (súborov a adresárov) aktuálneho adresára. V našom prípade systém vypísal položky: `linux, neuro, priklady, ulozisko, zoznam1, zoznam2, zoznam3.` Pričom nie je špecifikované, či ide o súbory, alebo adresáre. Po vykonaní príkazu sa systém znova ozve promptom a čaká na ďalší príkaz.

Pohyb po štruktúre adresárov

Príkaz cd slúži na pohyb po adresároch a urobí zmenu aktuálneho pracovného adresára.

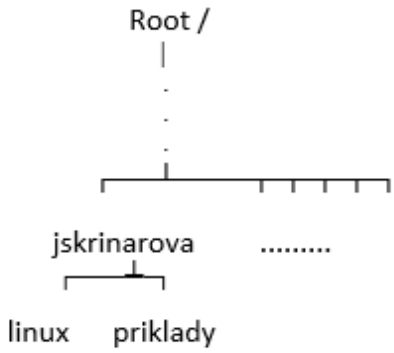
Syntax:

cd [adresar]

kde adresar je meno adresára, vrátane cesty k nemu, do ktorého sa chceme dostať.

Príklad:

Predpokladajme, že používateľ chce prejsť zo svojho adresára označeného napr. menom `jskrinarova` do adresára s menom `priklady`. Nech sú adresáre (priečinky) umiestnené takto:



CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz na prejdenie do adresára s názvom Linux.

```
jskrinarova@labs:~$ cd linux
jskrinarova@labs:~ /priklady$
```

Obrázok 56
Príklad na pohyb po štruktúre adresárov v systéme Linux

Ako vidíme na obrázku, systém vykonal príkaz a v prompte vypisuje meno aktuálneho adresára. Výpis promptu je označený žltou farbou. Ak by váš systém nezobrazoval meno pracovného adresára, použite príkaz pwd (angl. print working directory).

V príkaze cd môžeme použiť aj symbolické mená adresárov:

. označuje meno aktuálneho adresára,

.. označuje meno najbližšieho nadradeného (rodičovského) adresára,

/ označuje meno koreňového adresára ROOT.

CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz na prechod do koreňového (root) adresára.

```
jskrinarova@labs:~$ cd /
jskrinarova@labs:/$
```

Obrázok 57
Príklad na prejdenie do adresára root

CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz na výpis cesty k aktuálnemu adresáru

```
jskrinarova@labs:~$ pwd
jskrinarova@labs:/$
```

Obrázok 58
Zobrazenie cesty z adresára root do aktuálneho adresára

Príkaz cd je môžeme použiť aj bez argumentu. V takom prípade systém použije meno domovského adresára. Inak povedané, systém sa nastaví do nášho (tzv. domovského) adresára. To je ten adresár, do ktorého Vás systém nastavil po prihlásení.

CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz na presun do nášho vlastného domovského adresára

```
jskrinarova@labs:/$ cd
jskrinarova@labs:~$
```

Obrázok 59
Spôsob presunu do domovského adresára

ZAPAMÄTAJTE SI!

Všimnime si, že systém cestu do nášho domovského adresára vypisuje znakom ~.

Výpis obsahu adresárov

S príkazom `ls` sme sa už stretli na začiatku tejto kapitoly. Tu sa zameriame najmä na použitie niektorých parametrov. Príkaz `ls` umožňuje vypísať obsah (položky) aktuálneho adresára. Syntax príkazu [4]:

ls [-aClpR] [subor]

kde:

-aClpR - sú parametre

subor - je meno súboru, o ktorom chceme získať informácie.

CVIČENIE – POUŽITE!

Príklad: Zadajme systému príkaz ls tak, aby sa zobrazili aj vlastnosti položiek.

```
jskrinarova@labs:~$ ls -l
total 16
drwxr-xr-x 2 jskrinarova UCITEL 4096 Oct 20 14:14 priklady
drwxr-xr-x 2 jskrinarova UCITEL    0 Sep 30 10:38 ulozisko
-rw-r--r-- 1 jskrinarova UCITEL 1230 Apr 29  2020 zoznam1
-rw-r--r-- 1 jskrinarova UCITEL  678 Apr 29  2020 zoznam2
-rw-r--r-- 1 jskrinarova UCITEL  735 Apr 29  2020 zoznam3
jskrinarova@labs:~$
```

Obrázok 60
Príkaz ls -l

Každej položke (súboru alebo adresáru) je vo výpise priradený jeden riadok. Na konci riadku, vpravo je uvedené meno súboru (alebo adresára). To znamená, že riadok hovorí o tej položke, ktorej meno je na konci riadku.

Poďme teraz po poriadku a objasnime vlastnosti súboru od začiatku riadku:

Celkom vľavo sú uvedené prístupové práva. S nimi sa zoznámime neskôr. Prvý znak prístupových práv má [3] nasledovný význam:

- d označuje adresár,
- znak „-“označuje obyčajný súbor,

- b označuje špeciálny blokový súbor,
- c označuje špeciálny znakový súbor.

Z toho vyplýva, že ak je na začiatku riadku d, meno, ktoré je na konci riadku označuje adresár a môžeme s ním robiť operácie ako s adresárom (vojsť do adresára, vytvárať podadresáre, prípadne ho zmazať). Ak je na začiatku riadku znak -, meno označuje súbor. Súbor môžeme spustiť, ak ide o vykonávateľný súbor, alebo vypísať jeho obsah, ak ide o textový (ASCII) súbor.

Za prvým znakom nasleduje 9 znakov prístupových práv. Sú to 3 trojice prístupových práv. Trojica práv obsahuje práva:

- r - právo čítať súbor
- w - právo zapisovať do súboru
- x - právo spustiť súbor

Prvá trojica sa týka vlastníka, druhá trojica skupiny vlastníkov a tretia trojica všetkých ostatných. V našom prípade (pozri príklad 6) je vlastníkom `jskrinarova` a skupina vlastníkov je UCITEL.

Po prístupových právach, nasleduje číslo, ktoré súvisí s odkazmi na umiestnenie súboru a bližšie to objasníme v kapitole Riadenie súborov.

Ďalej nasleduje kapacita, alebo inak povedané veľkosť súboru v bytoch. Po kapacite nasleduje dátum vytvorenia, alebo poslednej zmeny súboru.

CVIČENIE – POUŽITE!

Príklad : Zadajme systému príkaz ls tak, aby sa zobrazili aj vlastnosti položiek a systémové súbory.

```
jskrinarova@labs:~$ ls -la
total 36
drwx--x--x   6 jskrinarova UCITEL 4096 Nov 20 09:45 .
drwxr-xr-x 100 root        root   4096 Nov 19 14:31 ..
-rw-------   1 jskrinarova UCITEL 2387 Nov 20 23:27 .bash_history
drwx------   3 jskrinarova UCITEL 4096 Apr 20  2020 .gnupg
drwxr-xr-x   3 jskrinarova UCITEL 4096 Apr 28  2020 .local
drwxr-xr-x   2 jskrinarova UCITEL 4096 Oct 20 14:14 priklady
drwxr-xr-x   2 jskrinarova UCITEL    0 Sep 30 10:38 ulozisko
-rw-r--r--   1 jskrinarova UCITEL 1230 Apr 29  2020 zoznam1
-rw-r--r--   1 jskrinarova UCITEL  678 Apr 29  2020 zoznam2
-rw-r--r--   1 jskrinarova UCITEL  735 Apr 29  2020 zoznam3
jskrinarova@labs:~$
```

Obrázok 61
Príkaz ls –la

Tvorba adresárov

Každý používateľ si môže vo svojom (domovskom) adresári a v jeho podadresároch vytvárať nové podadresáre. Adresár je možné vytvoriť pomocou príkazu mkdir. [5]

Syntax:

mkdir [-p] adresar

kde:

p - je parameter a adresar - je názov adresára, ktorý chceme vytvoriť.

CVIČENIE – POUŽITE!

Príklad: Predpokladajme nasledovný obsah aktuálneho adresára, ktorý zistíme príkazom ls -l:

```
jskrinarova@labs:~$ ls -l
total 16
drwxr-xr-x 2 jskrinarova UCITEL 4096 Oct 20 14:14 priklady
drwxr-xr-x 2 jskrinarova UCITEL    0 Sep 30 10:38 ulozisko
-rw-r--r-- 1 jskrinarova UCITEL 1230 Apr 29  2020 zoznam1
-rw-r--r-- 1 jskrinarova UCITEL  678 Apr 29  2020 zoznam2
-rw-r--r-- 1 jskrinarova UCITEL  735 Apr 29  2020 zoznam3
jskrinarova@labs:~$
```

Vytvorme adresár s názvom Zoznamy. Treba dať pozor, aby sme meno adresára Zoznamy napísali s veľkým písmenom.

```
jskrinarova@labs:~$ mkdir Zoznamy
jskrinarova@labs:~$
```

Obrázok 62
Príkaz mkdir

Vytvorili sme adresár s názvom Zoznamy. Systém nevypisuje žiadne chybové hlásenie. Z toho vyplýva, že adresár je vytvorený. Môžeme sa o tom presvedčiť tak, že znova urobíme výpis.

```
jskrinarova@labs:~$ ls -l
total 28
drwxr-xr-x 2 jskrinarova UCITEL 4096 Oct 20 14:14 priklady
drwxr-xr-x 2 jskrinarova UCITEL    0 Sep 30 10:38 ulozisko
-rw-r--r-- 1 jskrinarova UCITEL 1230 Apr 29  2020 zoznam1
-rw-r--r-- 1 jskrinarova UCITEL  678 Apr 29  2020 zoznam2
-rw-r--r-- 1 jskrinarova UCITEL  735 Apr 29  2020 zoznam3
drwxr-xr-x 2 jskrinarova UCITEL 4096 Nov 21 16:23 Zoznamy
jskrinarova@labs:~$
```

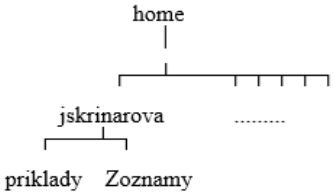
Kopírovanie súborov

Príkaz `cp` vytvorí kópiu existujúceho súboru. Syntax:

```
cp zdroj cieľ
```

kde `zdroj` je meno zdrojového, skutočne existujúceho súboru, ktorý chceme niekam prekopírovať a `cieľ` je názov cieľového adresára, kam chceme zdrojový súbor prekopírovať. Ak je na konci cesty uvedené odlišné meno od mena zdrojového súboru, súbor bude v cieľovom adresári uložený pod novým menom. V prípade, že meno súboru nemeníme, nemusíme ho zadávať pri špecifikácii cieľa.

Predpokladajme nasledovnú štruktúru adresárov:



CVIČENIE – POUŽITE!

Príklad 9: Najskôr urobme výpis obsahu aktuálneho priečinka:

```
jskrinarova@labs:~$ ls -l
total 20
drwxr-xr-x 2 jskrinarova UCITEL 4096 Oct 20 14:14 priklady
drwxr-xr-x 2 jskrinarova UCITEL    0 Sep 30 10:38 ulozisko
-rw-r--r-- 1 jskrinarova UCITEL   75 Nov 21 17:24 zoznam1
-rw-r--r-- 1 jskrinarova UCITEL  678 Apr 29  2020 zoznam2
-rw-r--r-- 1 jskrinarova UCITEL  735 Apr 29  2020 zoznam3
drwxr-xr-x 2 jskrinarova UCITEL 4096 Nov 21 16:23 Zoznamy
jskrinarova@labs:~$
```

Prekopírujme súbor zoznam1 do adresára Zoznamy.

```
jskrinarova@labs:~$ cp zoznam1 Zoznamy
jskrinarova@labs:~$
```

Obrázok 63
Príkaz cp

Aby sme skontrolovali, že sme správne skopírovali súbor zoznam1, pozrieme sa do adresára Zoznamy.

```
jskrinarova@labs:~$ ls -l Zoznamy
total 4
-rw-r--r-- 1 jskrinarova UCITEL   75 Nov 21 17:24 zoznam1
jskrinarova@labs:~$
```

Vidíme, že súbor zoznam1 sa skutočne nachádza v adresári Zoznamy.

Výpis obsahu textových súborov

Príkaz cat slúži na výpis textových súborov na štandardný výstup. [5]

Syntax:

cat [-uv][subor]

kde:

-uv sú parametre príkazu cat

subor je meno súboru, ktorý chceme vypísať

CVIČENIE – POUŽITE!

Príklad: Vypíšme obsah súboru zoznam1. V prípade, že takýto súbor nemáme, môžeme ho vytvoriť pomocou jednoduchého textového editora nano. Stačí napísať príkaz nano zoznam1.

Z predchádzajúcich výpisov vidíme, že sa súbor zoznam1 nachádza v aktuálnom adresári. Z rovnakého obrázku vidíme, že súbor zoznam1 je , pre nás (ako vlastníka) určený na čítanie (symbol r v prvej trojici prístupových práv). Z toho vieme, že tento súbor môžeme vypísať pomocou príkazu cat.

```
jskrinarova@labs:~$ cat zoznam1
Adam
Eva
Cecília
Dušan
Emil
Františka
Gustáv
Helena
Ivan
Janka
Klaudia
jskrinarova@labs:~$
```

Obrázok 64
Príkaz cat

Zmena prístupových práv

Príkaz chmod slúži na zmenu prístupových práv k súboru. Prístupové práva sa zadávajú pre rôzne triedy používateľov pomocou symbolov [4]:

- u - vlastník (angl. user) – vo výpise pomocou príkazu ls ide o prvú trojicu prístupových práv,
- g - skupina vlastníkov (angl. group of owners), vo výpise pomocou príkazu ls ide o druhú trojicu prístupových práv,
- o - ostatní používatelia (angl. others), vo výpise pomocou príkazu ls ide o druhú trojicu prístupových práv.

Prístupové práva sa zadávajú pomocou symbolov:

- r - právo čítať súbor,
- w - právo zapisovať do súboru,
- x - právo spustiť súbor.

CVIČENIE – POUŽITE!

Nech sú práva k súboru pred ich zmenou takéto:

```
-rw-r--r-- 1 jskrinarova UCITEL   75 Nov 21 17:24 zoznam1
```

Zmeňme prístupové práva k súboru zoznam1 tak, aby skupina vlastníkov mala právo zapisovať do tohto súboru.

```
jskrinarova@labs:~$ chmod g+w zoznam1
jskrinarova@labs:~$
```

Obrázok 65
Príkaz chmod

Po zmene prístupových práv, budú tieto práva nastavené takto:

```
-rw-rw-r-- 1 jskrinarova UCITEL   75 Nov 21 17:24 zoznam1
```

Prístupové práva ľahko nastavíme aj pre každú triedu vlastníkov inak pomocou váh. K jednotlivým prístupovým právam sú váhy nastavené tak, ako je to uvedené v tabuľke .

Tabuľka 4 Váhy prístupových práv

Prístupové právo	Váha
r	4
w	2
x	1

Prístupové práva sa potom zadávajú v príkaze pomocou troch číslic, každá pre jednu triedu vlastníkov (vlastník, skupina vlastníkov, ostatní používatelia). Jednotlivé číslice sú tvorené súčtom váh. [5]

CVIČENIE – POUŽITE!

Príklad: Majme súbor priklad1. Zmeňme prístupové práva k súboru priklad1 ak, aby sme (vlastník) mohli tento súbor spúšťať, skupina vlastníkov mohla súbor čítať aj do neho zapisovať a ostatní mohli tento súbor len čítať.

Nech sú práva k súboru pred ich zmenou takéto:

```
-rw------- 1 jskrinarova UCITEL  46 Oct  1 09:42 priklad1
```

Zmenu práv urobíme príkazom na obrázku 12. Práva nastavíme takto:

- pre vlastníka súboru: 4+2+1=7 t.j. r + w + x,
- pre skupinu vlastníkov: 4+1=5 t.j. r + x,
- pre ostatných užívateľov: 4 t.j. r.

```
jskrinarova@labs:~$ chmod 754 priklad1
jskrinarova@labs:~$
```

Obrázok 66
Príkaz chmod s použitím váh

Práva k súboru priklad1 po zmene práv:

```
-rwxr-xr-- 1 jskrinarova UCITEL  46 Oct  1 09:42 priklad1
```

6.3 Niekoľko príkladov v skriptovacom jazyku Bash

Skriptovaci jazyk Bash nám umožňuje spustiť dávku príkazov, ktoré zvyčajne zadávame, postupne po jednom, do príkazového riadku. Takto vytvorené súbory nazývame skripty. Skripty píšeme v textovom editore. Veľmi jednoduchý skript, ktorý len vypisuje určitý text, je zobrazený na obrázku 13. Príkaz echo vypíše text. [6]

CVIČENIE – POUŽITE!

Príklad: Vytvorme jednoduchý skript, ktorý vypisuje text. Použijeme textový editor nano a súbor uložíme pod menom priklad1. Pozri obrázok 13.

```
jskrinarova@labs:~/priklady$ cat priklad1
echo Tento program vypise vetu
echo AHOJ SVET
```

Obrázok 67
Obsah jednoduchého skriptu

Keď napíšeme skript v textovom editore, automaticky bude mať nastavené prístupové práve takto rw-r--r--. To znamená, že vlastník súboru môže tento súbor čítať a do neho aj zapisovať. Ak chceme novovytvorený skript spúšťať, musíme mu nastaviť práva na vykonávanie. Pozri obrázok 14.

```
jskrinarova@labs:~$ chmod g+w zoznam1
jskrinarova@labs:~$
```

Obrázok 68
Nastavenie práv spustiteľného súboru

Po zmene prístupových práv bude výpis práv nasledovný:

```
-rwxr--r-- 1 jskrinarova UCITEL 46 Oct  1 09:42 priklad1
```

Po takto nastavených právach môžeme súbor spustiť, ako to ukazuje obrázok 15.

```
jskrinarova@labs:~/priklady$ ./priklad1
Tento program vypise vetu
AHOJ SVET
jskrinarova@labs:~/priklady$
```

Obrázok 69
Spôsob spustenia skriptu

Všimnime si, že skript sa spúšťa pomocou znakov ./priklad1. Pomocou znakov./ sa sprístupní cesta k interpreteru Bash a za ním nasleduje meno skriptu. (v našom prípade je to priklad1).

Skripty však možno spúšťať aj s parametrami. Parametre, vo vnútri skriptu, majú mená $1, $2, ..., $n. Tieto parametre sa naplnia pri spustení skriptu. Preto pri spúšťaní skriptu treba parametre napísať za menom skriptu. Pomocou špeciálneho parametra $0 môžeme v skripte

používať meno tohto skriptu a pomocou parametra $# zistíme počet parametrov, ktorý bol zadaný pri spustení tohto skriptu.

CVIČENIE – POUŽITE!

Príklad 14: Vytvorme jednoduchý skript, ktorý vypisuje text a do textu doplní parametre, ktoré sme zadáme pri spustení skriptu.

```
echo Tento subor zisti svoje meno, pocet zadanych parametrov a prve dva echo parametre vypise
echo Nazov suboru $0
echo Pocet parametrov $#
echo parameter1=$1
echo parameter2=$2
echo Poznamka: Subor spustime napr. takto ./priklad2 ja ty on,
echo kde ja ty on su parametre skriptu
```

Obrázok 70
Obsah skriptu s parametrami

Skript na prvý pohľad vyzerá zložito, no keď ho spustíme uvidíme, že práca s parametrami je jednoduchá. Nezabudnime, že skriptu (t.j. súboru napr. s názvom príklad2) treba nastaviť prístupové práva tak, aby sme ho mohli spustiť. Potom môžeme skript priklad2 spustiť spôsobom, ktorý je uvedený na obrázku 17

```
jskrinarova@labs:~/priklady$ ./priklad2 ja ty on
```

Obrázok 71
Spôsob spustenia skriptu s parametrami

Výpis skriptu priklad2 vidíme na obrázku 18.

```
Tento skript zisti svoje meno, pocet zadanych parametrov a prve dva
parametre vypise
Nazov suboru: ./priklad2
Pocet parametrov: 3
parameter=ja
parameter=ty
Poznamka: Subor spustime napr. takto ./priklad2 ja ty on,
kde ja ty on su parametre skriptu
```

Obrázok 72
Výpis po spustení skriptu priklad2

Z výpisu vidíme, že skript má meno priklad2, pri spustení skriptu sme zadali 3 parametre, prvý parameter je „ja“ a druhý parameter je „ty“.

ÚLOHA – RIEŠTE!

Zistite, čo sa stane, ak spustíme skript priklad2 a zabudneme napísať parametre za menom skriptu.

Pomocou nasledujúceho príkladu sa naučíme používať vetvenie v skriptoch. Majme skript priklad3.

CVIČENIE – POUŽITE!

Príklad 15: Vytvorme skript, ktorý obsahuje vetvenie. Pozri obrázok 19.

```
echo Tento skript vyhodnoti chybove hlasenia pri kopirovani
if cp $1 $2
then echo OK
else echo Nepodarilo sa
fi
echo Spustenie: priklad3 subor1 subor2
```

Obrázok 73
Obsah skriptu s vetvením

V skripte používame príkaz if. [6]

Syntax:

```
if testovací príkaz
      then príkaz1
      else príkaz2
fi
```

Pomocou príkazu if testujeme, či testovací príkaz prebehne bez chybového hlásenia. Ak prebehne bez chyby vykoná sa prikaz1, inak sa vykoná prikaz2. V našom prípade (obrázok 17), ak príkaz cp úspešne skopíruje súbor $1 do súboru $2 (parametre zadané za menom skriptu) vypíše sa „OK“, v opačnom prípade sa vypíše „Nepodarilo sa“.

Majme súbor zoznam4 v aktuálnom adresári. Ak taký súbor nemáme, vytvoríme ho pomocou editora nano. Skript priklad3 spustíme, pričom nezabudneme na parametre (pozri obrázok 20). V našom prípade sa skopíruje súbor s meno zoznam4 do nového súboru, ktorý dostane meno zoznam5.

```
jskrinarova@labs:~/priklady$ ./priklad3 zoznam4 zoznam5
```

Obrázok 74
Spôsob spustenia skriptu vetvenia

Výpis po spustení skriptu je uvedený na obrázku 21. Z výpisu vidíme, že príkaz kopírovania skončil úspešne. Vypísal sa reťazec „OK“.

```
Tento vyhodnoti chybove hlasenia pri kopirovani
OK
Spustenie: priklad3 subor1 subor2
```

Obrázok 75
Výpis po spustení skriptu s vetvením

ÚLOHA – RIEŠTE!

Zistite, čo sa stane, ak spustíme skript priklad3 bez parametrov za menom skriptu.

ÚLOHA – RIEŠTE!

Zistite, čo sa stane, ak spustíme skript priklad3 s chybnými parametrami za menom skriptu.

Pomocou príkladu 15 sa naučíme sa naučíme v skriptoch používať cyklus s pevným počtom opakovaní.

CVIČENIE – POUŽITE!

Príklad 16: Vytvorme skript, ktorý vypíše obsah aktuálneho adresára pomocou príkazu s pevným počtom opakovaní. Majme skript s názvom priklad4.

```
for i in *
do
echo $i
done
```

Obrázok 76

Obsah skriptu s cyklom s pevným počtom opakovaní

```
V skripte používame cyklus for. [6]

Syntax:

for premenná in zoznam
    do
    príkazy
    done
```

Pomocou príkazu for sa prehľadávajú všetky položky definované v zozname. Tieto položky sa postupne (pri každom opakovaní cyklu) priradia do premennej. Príkazy, ktoré sa vykonávajú v cykle, medzi výrazmi do a done, urobia určitú operáciu s položkou so zoznamu.

V našom prípade (pozri obrázok 22) * nahrádza zoznam všetkých položiek v aktuálnom adresári a následne urobí príkaz s každou položkou tak, že položku vypíše. Postupne vypíše všetky položky.

Spustíme skript s názvom priklad4 a výpis skriptu, ktorý vypisuje všetky položky aktuálneho adresára vidíme na obrázku 23.

```
priklad1

priklad2

priklad3

priklad4

zoznam4

zoznam5
```

Obrázok 77
Výpis po spustení skriptu s príkazom cyklu

ZHRNUTIE

Naučili sme sa, aké sú hlavné úlohy operačných systémov. Vieme pracovať v systéme a vytvárať jednoduché skripty v Bash. Rozumieme, čo sú prístupové práva k súborom a vieme ich nastavovať.

KĽÚČ K ÚLOHÁM

1	Aké parametre program vypíše?
2	Nepodarilo sa.
3	Nepodarilo sa.

BIBLIOGRAFIA

[1] MARTINCOVÁ, P., GRONDŽÁK, K.: Operačné systémy. 1.vyd. Žilina: EDIS, 2004, 294 s. ISBN 80-8070-242-X.

[2] BROOKSHEAR, J. G.: Computer Science: An Overview. 11.vyd. New Jersey: Prentice Hall, 2012, 624 s. ISBN 0132569035.

[3] STALLINGS, W.: Operating Systems: Internals and Design Principles. 7 vyd. Prentice Hall, 2012. ISBN 978-0-13-230998-1.

[4] SKOČOVSKÝ, L.: Princípy a problémy operačního sysytému UNIX. Science, 1993

[5] JODAS, P. KNOPP, M.: Pracujeme se soubory. Softwarové noviny 10/1993

[6] LIŠKA, R.: Úvod do Unixu. http://kfe.fjfi.cvut.cz/~liska/unix/

7 RIADENIE PROCESOV V OPERAČNOM SYSTÉME

V predchádzajúcich kapitolách sme sa naučili ako pracuje procesor. My, používatelia v okne operačného systému klikneme na ikonu programu alebo aplikácie a tá začne bežať. *Čo sa deje vo vnútri operačného systému?*

CIEĽ

Čo budeme vedieť a čo budeme vedieť urobiť

- vysvetliť ako sa vytvorí a čo je proces v systéme,
- poznať štruktúru procesu,
- orientovať sa v štruktúre systému,
- zobrazovať procesy a pracovať v systéme,
- chápať životný cyklus procesu,
- chápať princíp multiúlohového systému,
- vedieť modelovať činnosť plánovacieho algoritmu.

Najskôr si zopakujeme ako pracuje procesor. Naučíme sa ako sa z programu urobí proces. Pomocou praktických príkladov budeme vedieť zistiť ako môžeme vidieť procesy v systéme. Porozumieme, či je možné, aby na počítači s jedným procesorom bežalo viac procesov a tiež ako pracujú procesy v multiúlohovom operačnom systéme. Nachvíľu sa ponoríme do fantastického sveta Harryho Pottera a ten nám pomôže porozumieť princíp užitočného algoritmu pre multiúlohové systémy.

Kľúčové slová

proces, procesor, štruktúra procesu, životný cyklus procesu, prepínanie procesov, plánovací algoritmus

MOTIVÁCIA

V súčasnom svete sa stretávame s rôznymi počítačmi a rôznymi operačnými systémami. V každom operačnom systéme môžeme spustiť program, ktorý je postupne vykonávaný procesorom počítača.

CVIČENIE – DISKUTUJTE!

Čo sa deje vo vnútri operačného systému?

Môžeme sa pozrieť ako vyzerá spustený počítačový program?

Ak je operačný systém multiúlohový znamená to, že nám na počítači, v jednom okamihu, môže bežať viac úloh súčasne?

7.1 Procesy

Jednou z úloh operačného systému je riadiť procesy v systéme. Zabezpečuje, aby správne bežali naše programy. Viete, že program, ktorý sme napísali v programovacom jazyku, je potrebné najskôr preložiť, aby sa z neho stal vykonávateľný program. Zjednodušene povedané, prekladom sa príkazov programovacieho jazyka stanú inštrukcie. Pri spustení programu sa inštrukcie zavedú do tzv. operačnej pamäti počítača (pamäť RAM). Do pamäti sa zavedú aj niektoré údaje a tieto údaje budeme nazývať operandy.

Pretože inštrukcie vykonáva procesor, zopakujme si čo procesor robí, (pozri kapitolu Pokročilé architektúry počítačov, pozri tab. 1 - Kroky spracovávania inštrukcie).

Tabuľka 5 Kroky spracovávania inštrukcie

Krok	Význam
1	Výber inštrukcie
2	Dekódovanie inštrukcie
3	Výpočet adresy
4	Výber operandu
5	Vykonanie inštrukcie
6	Uloženie výsledku

VÝKLAD

Keď sme si zopakovali ako pracuje procesor, poďme sa pozrieť na to, ako sa vykonávajú programy. Program je súbor uložený na disku, ktorý môžeme spustiť. Keď program spustíme, vykonáva určitú činnosť. Každý program, ktorý spustíme v systéme, vytvorí jeden alebo viac procesov. Inak povedané proces je program v činnosti [1].

ZAPAMÄTAJ SI!

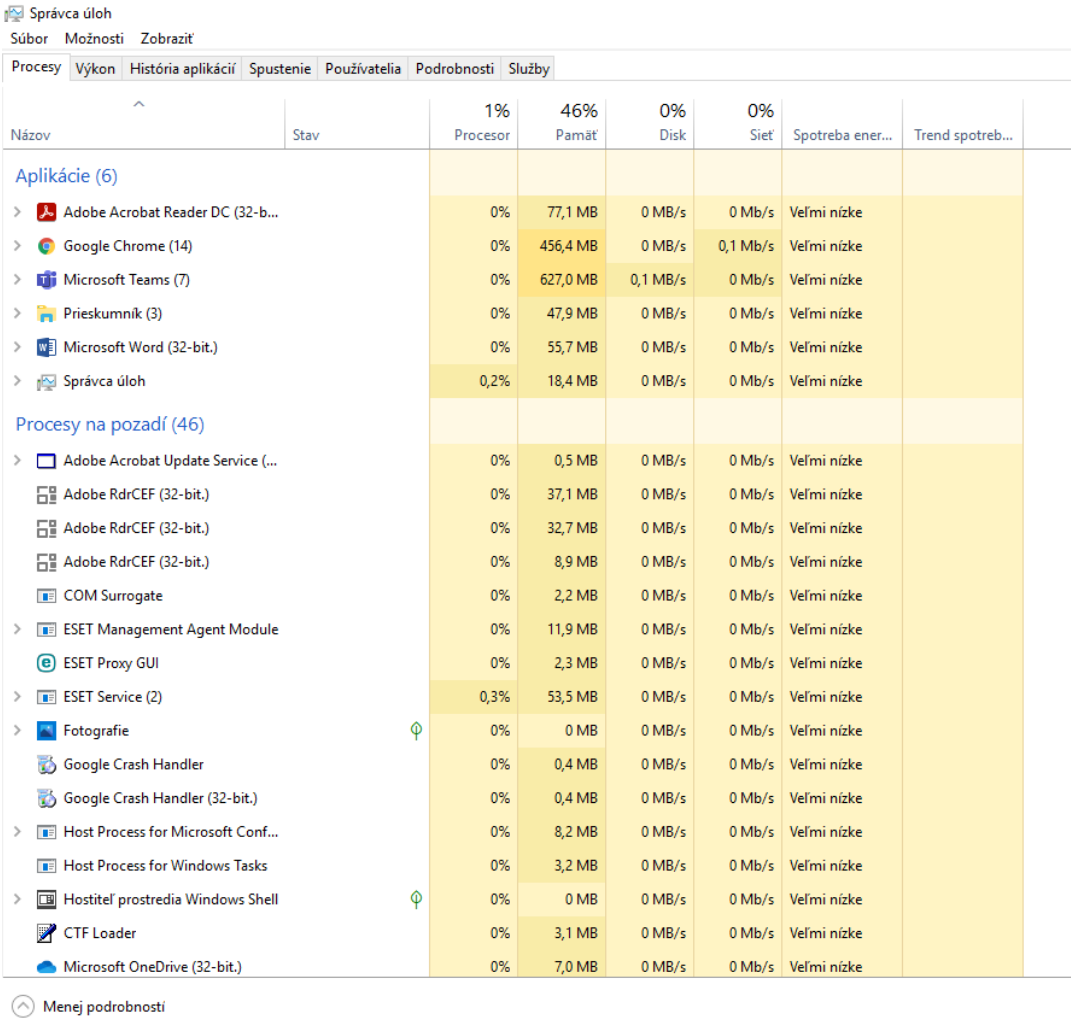
Proces je spustený počítačový program.

Každý program, ktorý spustíme v systéme, vytvorí jeden alebo viac procesov. Inak povedané proces je program v činnosti.

CVIČENIE – POUŽITE!

Ako zistíme, ktoré procesy máme spustené v operačnom systéme Windows?

Jednoducho tak, že spustíme Správcu úloh operačného systému. Použijeme napr. klávesové skratky CTRL + SHIFT + ESC. Správca úloh zobrazí všetky bežiace procesy. Pozri obrázok 1. V ľavom stĺpci vidíme najskôr zoznam spustených programov (aplikácií) a potom zoznam procesov.



Obrázok 78
Príklad zobrazenia všetkých procesov, ktoré bežia v operačnom systéme Windows, zobrazené pomocou správcu úloh

CVIČENIE – POUŽITE!

Ako zistíme, ktoré procesy máme spustené v operačnom systéme Linux?

Všetky spustené programy (procesy) sa nachádzajú v operačnom systéme Linux v priečinku s menom proc, (pozri obrázok 2).

Na zobrazenie výpisu môžeme použiť príkaz `ls /proc`.

```
jskrinarova@labs: ~
login as: jskrinarova
jskrinarova@labs.fpv.umb.sk's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb  7 14:42:22 2018 from 194.160.34.43
jskrinarova@labs:~$ ls /proc
1      1434   2      3      32693  640  759   96          misc
10     1451   20     30     32694  644  760   98          modules
101    1452   21     30035  32724  647  761   acpi        mounts
102    1453   21799  30064  32725  648  762   buddyinfo   mpt
103    1454   21896  30811  32727  649  763   bus         mtrr
104    1472   22     3098   3273   652  764   cgroups     net
10422  15     2205   31     32753  653  765   cmdline     pagetypeinfo
10678  1501   2221   31654  32754  654  766   consoles    partitions
108    1510   2227   31774  3280   655  7754  cpuinfo     sched_debug
10903  1534   22528  31778  33     656  7765  crypto      self
11     1535   23     31903  34     657  777   devices     slabinfo
113    1559   2311   31999  35     658  778   diskstats   softirqs
1167   1560   2320   32     36     659  779   dma         stat
118    1561   2345   32241  3873   664  780   driver      swaps
12     1562   24187  32242  39     665  781   execdomains sys
122    1563   24203  32243  40     685  782   fb          sysrq-trigger
1242   16     24276  32256  41     687  783   filesystems sysvipc
1261   1628   24290  32337  47     692  784   fs          timer_list
1262   1629   25     32338  48     7    785   interrupts  timer_stats
1274   1630   25675  32339  49     704  786   iomem       tty
1283   1631   25712  32365  5      717  787   ioports     uptime
1291   1664   25732  32428  506    721  788   irq         version
13     17     25734  32429  515    722  789   kallsyms    vmallocinfo
1336   170    25735  32430  520    723  8     kcore       vmstat
1338   17750  258    32525  522    724  851   keys        zoneinfo
134    178    26     32526  529    725  854   key-users
135    1798   26681  32527  530    732  9     kmsg
1360   18     27     32614  531    754  90    kpagecount
1362   181    28     32615  532    755  91    kpageflags
1382   1842   28832  32617  534    756  92    loadavg
1383   1846   29     32624  551    757  93    locks
1384   1904   29114  32691  595    758  9420  meminfo
jskrinarova@labs:~$
```

Obrázok 79
Príklad všetkých procesov, ktoré bežia v systéme počítača (/proc)

ZAPAMÄTAJTE SI!

Každý proces v systéme má svoje meno (číslo). Je to jedinečné číslo. To znamená, že žiadny iný proces v systéme nie je označený takým istým číslom.

CVIČENIE – POUŽITE!

Ako zistíme čísla procesov, ktoré sme spustili?

Pomocou príkazu ***ps*** vypíšeme čísla (PID) procesov, ktoré sme spustili (obrázok 3). Vidíme, že sme spustili 2 procesy s číslami ***3132*** a ***32694***.

```
jskrinarova@labs:~/priklady$ ps
  PID TTY          TIME CMD
 3132 pts/0    00:00:00 ps
32694 pts/0    00:00:00 bash
jskrinarova@labs:~/priklady$
```

Obrázok 80
Použitie príkazu ps, pomocou ktorého zistíme čísla našich procesov

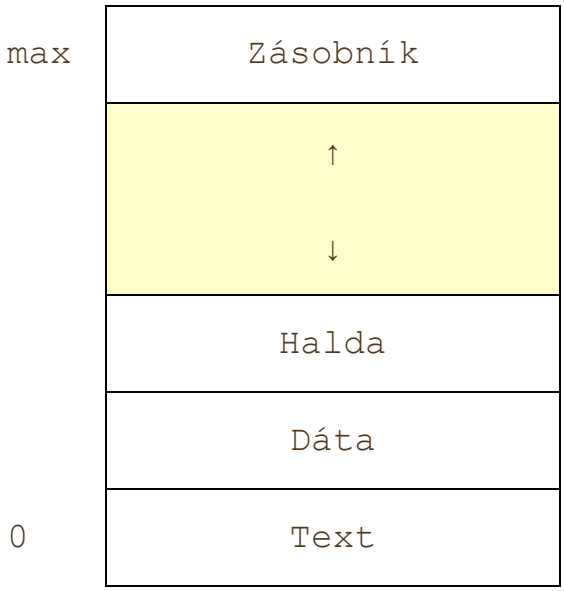
VÝKLAD

Program je pasívny súbor, ktorý je uložený na disku a je ho možné spustiť. Program obsahuje zoznam inštrukcií. Až keď program spustíme, vytvorí sa jeden alebo viac procesov. Proces je aktívny. Má počítadlo inštrukcií. Toto počítadlo ukazuje na ďalšiu inštrukciu, ktorá sa bude vykonávať.

Proces, ktorý je uložený v pamäti má nasledovnú štruktúru [1]:

- kód programu (tzv. text),
- aktuálny stav procesora a pamäte (počítadlo inštrukcií a obsah registrov procesora), ktoré súvisia s procesom,
- zásobník, v ktorom sú uložené dočasné údaje (napríklad parametre funkcie, návratové adresy a lokálne premenné),
- dátová časť, ktorá obsahuje globálne premenné,
- halda (nie vždy), ktorá je pamäťou a je dynamicky prideľovaná počas procesu spustenia.

Štruktúra procesu, ktorý je spustený a umiestnený v pamäti je zobrazená na obrázku 4.



Obrázok 81
Štruktúra procesu, uloženého v pamäti

Údaje sa ukladajú na spodok zásobníka (vždy na nižšiu adresu od poslednej zapísanej hodnoty) a v prípade, že sa údaje zapisujú do haldy, ukladajú sa vždy na vyššiu adresu od poslednej zapísanej hodnoty v halde. Preto sa pri zapisovaní údajov vždy zmenšuje voľný priestor medzi zásobníkom a haldou.

CVIČENIE – POUŽITE!

Ako vytvoríme proces?

Majme jednoduchý program s názvom ***priklad1.*** Tento jednoduchý program vypíše vetu ***„AHOJ SVET“.***

Spustíme program s názvom ***priklad1*** v operačnom systéme Linux.

Najskôr sa presvedčíme, že taký program naozaj máme. Použijeme príkaz ***cat priklad1*** (pozri obrázok 5). V prípade, že takýto program nemáme, ľahko si ho napíšeme, program obsahuje len 2 riadky.

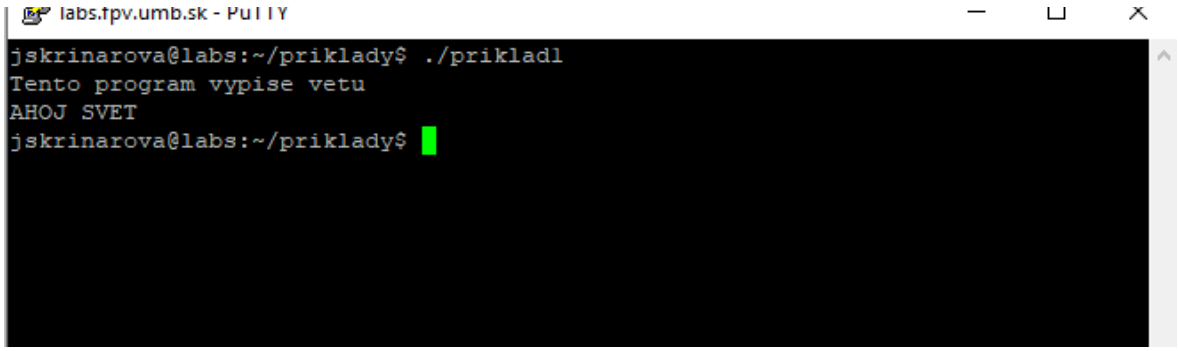
```
jskrinarova@labs:~/priklady$ cat priklad1
echo Tento program vypise vetu
echo AHOJ SVET
jskrinarova@labs:~/priklady$
```

Obrázok 82
Výpis programu priklad1

CVIČENIE – POUŽITE!

Teraz spustíme tento jednoduchý program.

Použijeme príkaz ***./priklad1*** (pozri obrázok 6).



Obrázok 83
Spustenie programu priklad1

CVIČENIE – POUŽITE!

Ako uvidíme aké procesy pri spustení vytvorí náš program ***priklad1***?

Potrebujeme súbežne zadať príkaz na spustenie programu aj príkaz na vypísanie procesov. Na to použijeme ***znak &***.

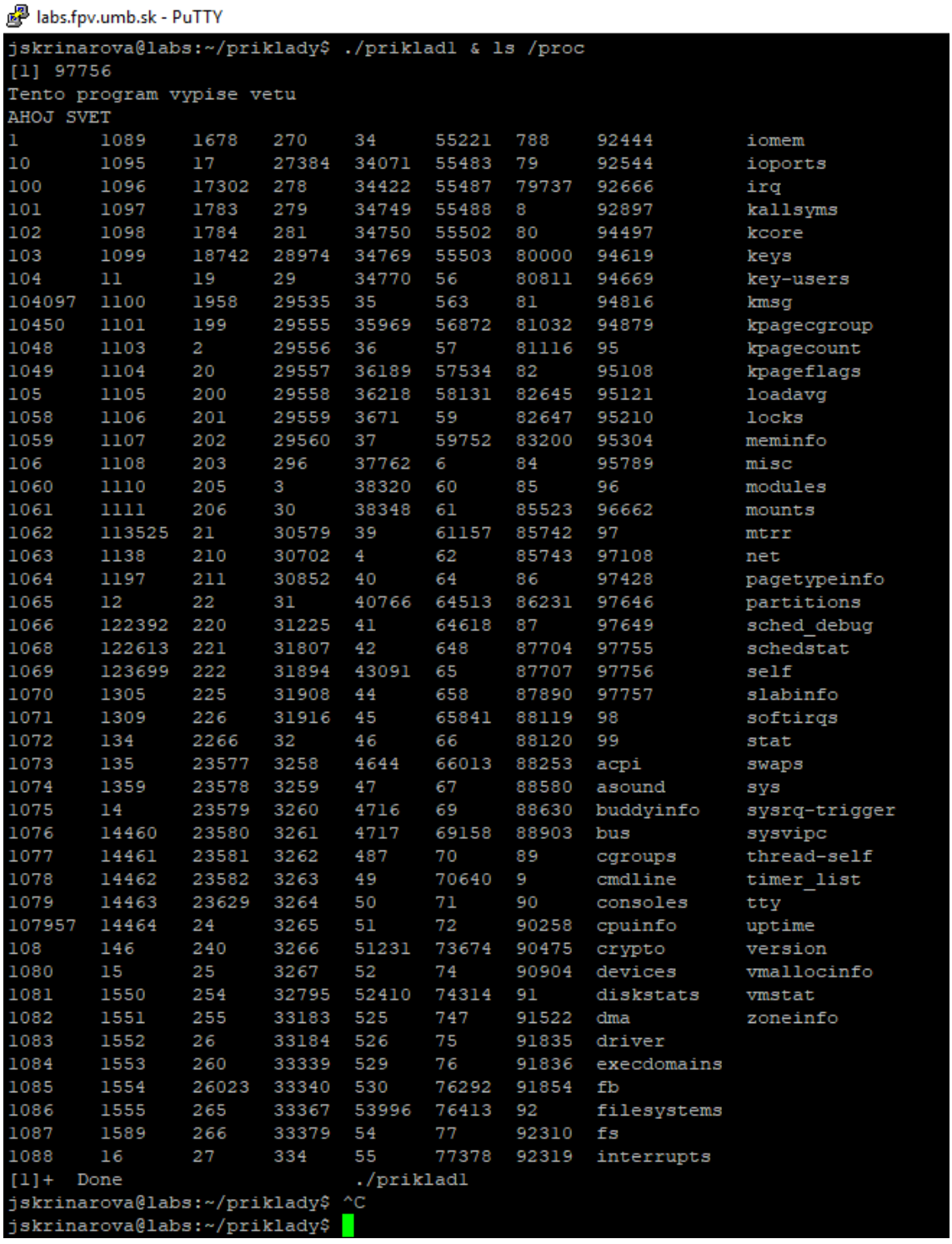
Spojíme dva príkazy, ktoré sme sa naučili používať v predchádzajúcich úlohách.

Použijeme príkaz ***./priklad1 & ls /proc*** (pozri obrázok 7).

Z výpisu vidíme, že spustením programu sa najskôr vytvoril proces a systém mu pridelil číslo ***97756***. Potom systém súbežne vykonával tento proces a vypísal zoznam všetkých bežiacich procesov. Vo ôsmom stĺpci vidíme číslo nášho bežiaceho procesu ***97756***.

ZHRNUTIE

Pomocou cvičení sme sa naučili ako spustiť program. Presvedčili sme sa, že proces je spustený počítačový program. Každý proces má svoje meno (číslo). Proces (jeho číslo) vidíme v systéme len počas behu programu.

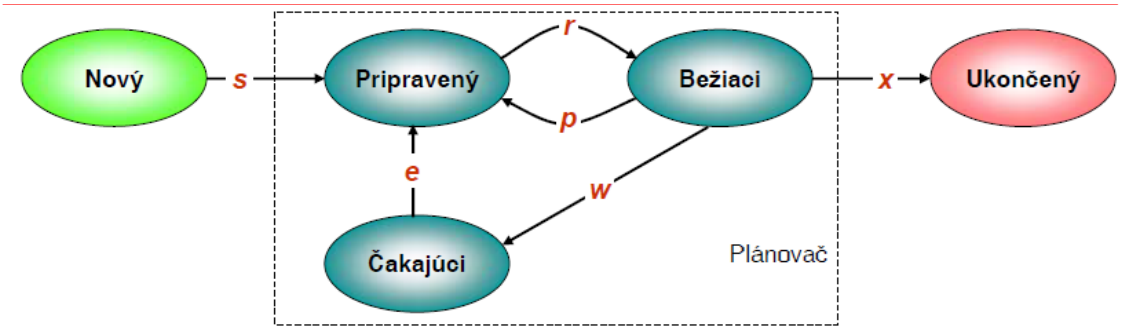


Obrázok 84
Zobrazenie aktuálne bežiaceho procesu, ktorý sa vytvorí spustením programu priklad1.

7.2 Procesy v multiúlohovom operačnom systéme

VÝKLAD

Ak má počítač iba jeden procesor (s jedným výkonným jadrom) môže v jednom okamihu vykonávať len jeden proces. Preto ***bežiaci*** proces (obrázok 8) môže byť len jeden a zvyšné procesy čakajú v rade ***pripravených*** procesov, alebo sú ***blokované***. ***Multitasking*** (beh viacerých úloh alebo procesov) je možný len prerušovaním behu procesu a rýchlym prepínaním sa medzi viacerými procesmi. Prepínanie procesov zabezpečuje operačný systém. Rýchle prepínanie vyvoláva dojem, že systém vykonáva súbežne viac procesov. Správanie sa procesu v systéme vyjadrujú stavy a prechody medzi stavmi procesu. Správanie sa procesu prebieha na základe životného cyklu procesu (obrázok 8) [2], [3].



Obrázok 85
Životný cyklus procesu v systéme

V okamihu vzniku proces (obrázok 8), nadobudne stav ***nový***. Keď je proces pripravený na vykonávanie, nadobúda stav ***pripravený***. Po pridelení procesora sa jeho stav mení na ***bežiaci***. Keď proces dokončí svoju činnosť (odíde zo systému – vymaže sa jeho číslo z ***/proc***) nadobudne stav ***ukončený***. Ak svoju činnosť nedokončí a z určitých dôvodov nadobudne stav ***čakajúci,*** čaká pokým nenastane určitá udalosť. Po výskyte udalosti opäť nadobudne *stav* ***pripravený***, [3].

Prechody medzi stavmi procesu sú uvedené v tabuľke 2.

Tabuľka 6 Prechody medzi jednotlivými stavmi procesu v systéme

Prechod	**Význam**
s	spustili sme program a tým sa zaviedol proces do systému, (štart procesu)
r	systém procesu pridelil procesor (proces beží, angl. run)
w	proces žiada o službu alebo údaje (čaká, angl. wait)
e	stala sa udalosť, proces dostal to, na čo čakal (udalosť, angl. event)
x	proces skončil svoju činnosť v systéme (koniec, angl. exit)
p	systém prerušil beh procesu (odobral procesu procesor) a prepol na iný proces (pridelil procesor inému procesu) – to je multiúlohovosť (angl. multitasking)

7.3 Plánovacie algoritmy

VÝKLAD

Naučili sme sa, že systém prepína procesy. Zaujímavé je vedieť, podľa čoho systém vyberá, ktorý proces bude bežať po prerušení aktuálne bežiaceho procesu. Na to slúžia tzv. plánovacie algoritmy. Úlohou plánovacieho algoritmu je stanoviť poradie, v ako budú procesy čakať v rade pripravených procesov. Pri najbližšom prepínaní sa prvý proces z radu pripravených procesov stane bežiacim procesom. Naučíme sa a budeme modelovať jeden z najpoužívanejších plánovacích algoritmov, ktorý má názov Round Robin.

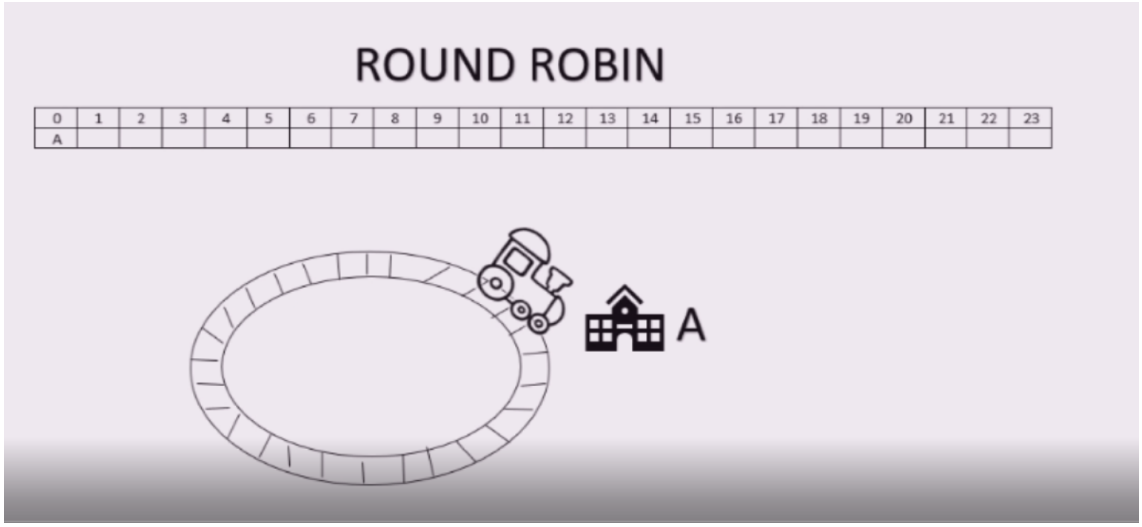
Algoritmus Round Robin a fantastický svet Harryho Pottera

Predstavme si fantastický svet Harryho Pottera, ktorý má železnicu s jednou kruhovou koľajou. Keď príde ***nový proces*** do systému vždy vznikne nová stanica a po skončení procesu stanica zmizne. Stanice predstavujú ilúzie našich procesov, ktoré sú do určitej miery rozpracované. Označme stanice (predstavte si, že sú to procesy) písmenami A až D. Vlak musí ísť stále po koľajniciach a zastaví sa pri každej stanici. Vždy, keď vlak zastane pri stanici, beží proces (systém mu pridelil procesor), po dobu jedného časového úseku. Pre lepšie porozumenie (pozri ilustračné video s algoritmom Round_Robin).

Tabuľka 7 Situácia v systéme - pre riešenie vizualizované na videu

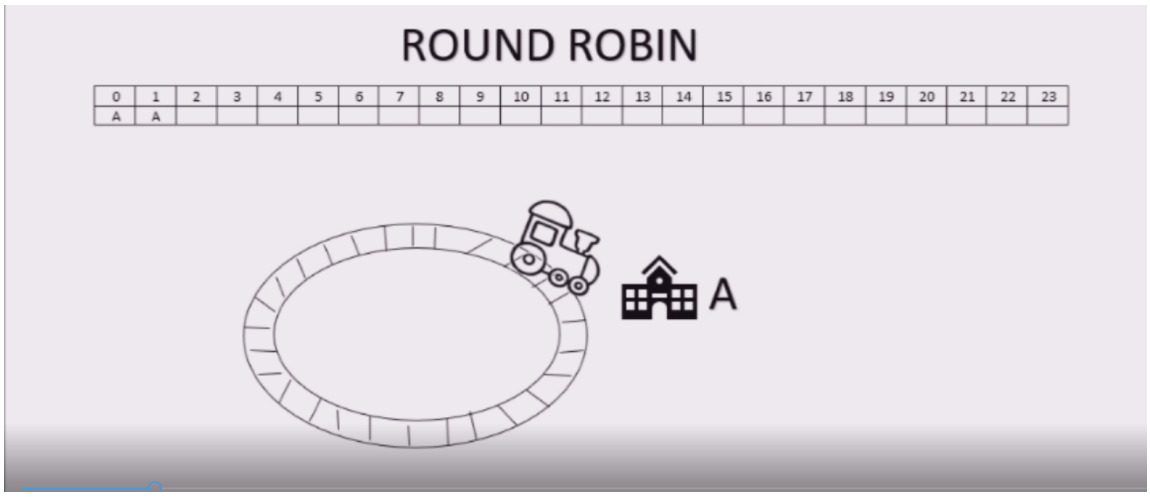
Proces	**Príchod do systému** **Na začiatku časového úseku**	**Dĺžka procesu** **(počet časových úsekov)**
A	0	7
B	2	4
C	4	2
D	6	1
E	8	10

V tabuľke 3 sme procesy (pre jednoduchosť) označili písmenami A až E. Príchod procesu do systému znamená, že počas časového úseku proces (napr. A) prišiel do systému a je pripravený na spracovanie (je v stave ***pripravený***). V našom príklade sú úseky, v ktorých je už možné vykonávať jednotlivé procesy, označené číslami 0 až 8. Dĺžka jednotlivých procesov je vyjadrená počtom časových úsekov. Inak povedané udáva, koľkokrát musí byť procesu pridelený procesor, aby bol dokončený celý proces.



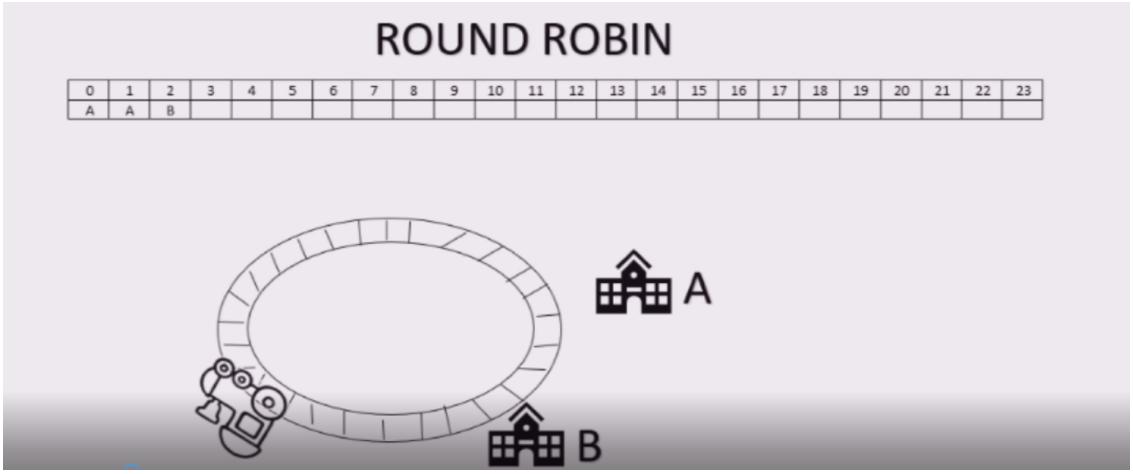
Obrázok 86
Železnica Harryho Pottera – vznik stanice A, podľa situácie uvedenej v tabuľke 3

Na obrázku 9 vidíme ako vznikla stanica A. Vlak sa zastaví na stanici A (procesu A je pridelený procesor a proces A sa vykonáva). Pretože na železnici nie je žiadna iná stanica, vlak prejde po celej kruhovej koľaji a znova sa zastaví na stanici A (pozri obrázok 10).



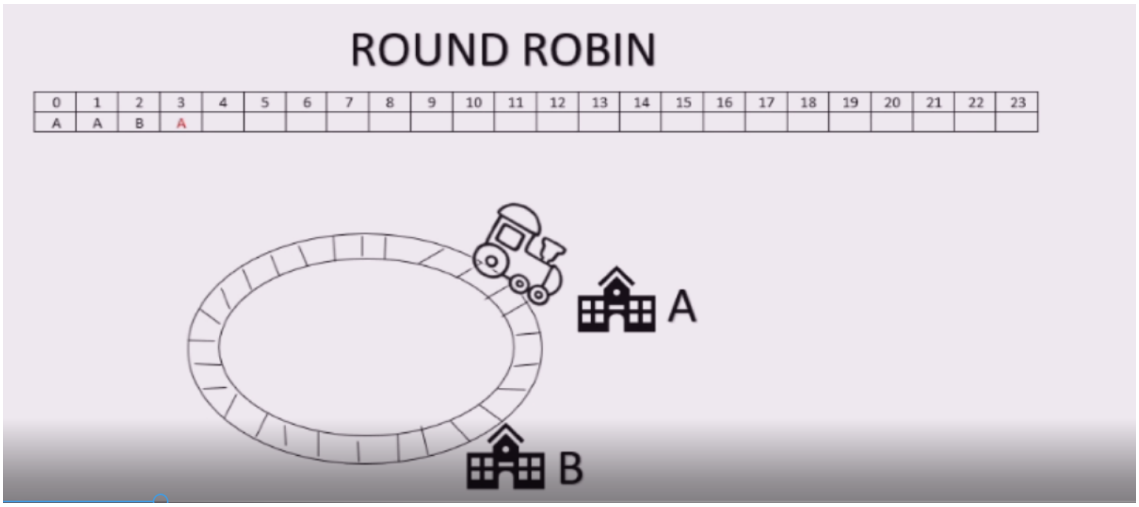
Obrázok 87
Železnica Harryho Pottera – vlak sa znova zastaví na stanici A, podľa situácie uvedenej v tabuľke 3

Podľa situácie v tabuľke 3 teraz vznikne stanica B (proces B je pripravený na spracovanie). Na železnici sa stanica B umiestni za stanicu A, v smere jazdy vláčika. Vláčik sa zastaví na stanici B (proces B sa vykoná).



Obrázok 88
Železnica Harryho Pottera – vlak sa znova zastavil na stanici B

Vlak pokračuje v jazde a pretože sa na koľaji, nachádza stanica A, vlak sa na nej zastaví .



Obrázok 89
Železnica Harryho Pottera – vlak sa znova zastaví na stanici A, podľa situácie uvedenej v tabuľke 3

CVIČENIE 5 – POUŽITE!

Sledujte video s postupom riešenia situácie uvedenej v tabuľke 3. Nakreslite na papier kruhovú koľaj. Postupne dokresľujte stanice, ktoré vznikajú a preškrtnite stanice, ktoré zanikajú. Vždy, keď vláčik zastane na stanici, zapíšte meno stanice.

Algoritmus Round Robin

Algoritmus Round Robin pracuje v systéme rovnako. Procesy bežia tak, že ich systém prerušuje (multitasking). Procesy sa nevykonávajú celé, ale vždy len určitá časť procesu. Jednotlivé procesy (ich čísla) sú usporiadané v rade pripravených procesov v poradí podľa času príchodu do systému. Po poslednom procese systém prepne znova na proces, ktorý je prvý v rade.

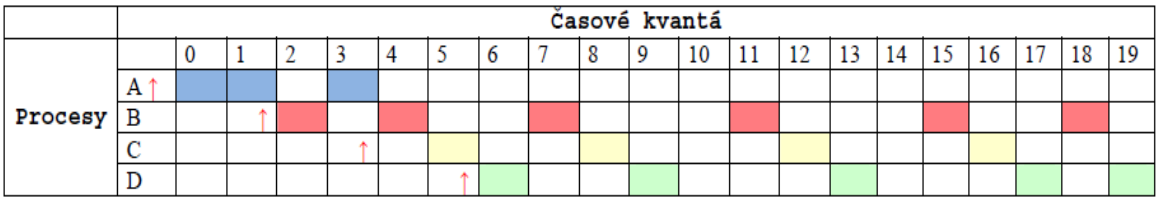
Tabuľka 8 Situácia v systéme – doba príchodu a požadovaná dĺžka trvania procesu v systéme

Proces	Príchod do systému Na začiatku časového úseku	Dĺžka procesu (počet časových úsekov)
A	0	3
B	2	6
C	4	4
D	6	5

ÚLOHA – RIEŠTE!

Aké je poradie vykonávania častí procesov, ak máme situáciu v systéme podľa tabuľky 4?

Riešme postupne úlohu. Najskôr si nakreslíme prázdnu tabuľku, akú vidíme na obrázku 13. Časové kvantá sú už spomínané časové úseky. Nakreslíme červené šípky pred začiatkom časových úsekov, ktoré znamenajú príchod procesu do systému. Potom postupne vyfarbujeme časové úseky pre procesy, ktoré bežia podľa algoritmu. Proces A je označený modrou, proces B červenou, proces C žltou a proces D zelenou farbou. Farba výplne časového úseku určuje, ktorý proces beží (má pridelený procesor) v tom časovom úseku.



Obrázok 90
Riešenie úlohy 1 - poradie spracovania častí jednotlivých procesov podľa situácie v systéme uvedenej v tabuľke 3

CVIČENIE – ANALYZUJTE!

Otázka 1: ***Vidíme, že proces A sa na začiatku vykonáva počas 2 časových úsekov. Viete prečo?***

Odpoveď***: Je to preto, lebo proces A je počas úsekov 0 a 1 zatiaľ sám v systéme.***

Otázka 2: ***Prečo v 5. časovom úseku nebeží proces A?***

Odpoveď: ***Lebo proces A už skončil. Jeho dĺžka bola 3 časové úseky (pozri tabuľku 3).***

ZHRNUTIE

Naučili sme sa, že sa procesy nevykonávajú celé, ale postupne len ich časti. Operačný systém postupne prepína medzi procesmi, ktoré sú pripravené. Naučili sme sa modelovať prácu plánovacieho algoritmu Round Robin, ktorý určuje poradie vykonávania procesov.

ÚLOHA – RIEŠTE!

Spustite program ***priklad1*** tak, aby ste videli číslo procesu v zozname vašich procesov.

ÚLOHA – RIEŠTE!

Majme situáciu v systéme, aká je uvedená v tabuľke 9.

Nájdite správne poradie vykonávania častí procesov, podľa algoritmu Round Robin Správne poradie môžete nakresliť napr. vo forme tabuľky..

Tabuľka 9 Úloha 3 - situácia v systéme – doba príchodu a požadovaná dĺžka trvania procesu v systéme

Proces	**Príchod do systému** **Na začiatku časového úseku**	**Dĺžka procesu** **(počet časových úsekov)**
A	0	7
B	2	2
C	4	4
D	6	3

KĽÚČ K ÚLOHÁM

1	Predtým ako začnete vyfarbovať časové úseky, nakreslite si železnicu Harryho Pottera a to Vám pomôže určiť, ktorý proces má nasledovať ako ďalší.
2	***ps & ./priklad1***
3	A A B A B C D A C D A C D A C A

BIBLIOGRAFIA

[1] MARTINCOVÁ, P., GRONDŽÁK, K.: Operačné systémy. 1.vyd. Žilina: EDIS, 2004, 294 s. ISBN 80-8070-242-X.

[2] BROOKSHEAR, J. G.: Computer Science: An Overview. 11.vyd. New Jersey: Prentice Hall, 2012, 624 s. ISBN 0132569035.

[3] STALLINGS, W.: Operating Systems. Prentice Hall, 2005. ISBN 0-13-147954-7

8 RIADENIE PAMÄTE V OPERAČNOM SYSTÉME

V predchádzajúcich kapitolách sme sa naučili ako systém z programov vytvára procesy. Tieto procesy sa pri spracúvaní nachádzajú v operačnej pamäti (RAM). *Ktorá časť počítača riadi ako sa procesy presúvajú do pamäte? Je to procesor, alebo za to zodpovedá operačný systém?*

CIEĽ

Čo budeme vedieť a čo budeme vedieť urobiť

- vysvetliť ako sa procesy umiestňujú v pamäti RAM,
- vedieť akú úlohu plní procesor a akú operačný systém pri riadení pamäte RAM,
- poznať algoritmy umiestňovania procesov tak, aby počítač pracoval efektívne,
- chápať princíp riadenia voľných úsekov pamäte,
- vedieť modelovať činnosť defragmentačných algoritmov.

Naučíme sa, že hlavnou úlohou riadenia pamätí je vypočítať fyzickú adresu miesta v pamäti RAM. Rovnako dôležité je riadiť umiestňovanie procesov v tejto pamäti. Preto je potrebné poznať voľné miesta v pamäti. Na základe poznania obsadených aj voľných úsekov pamäte je potom možné robiť defragmentáciu pamäte.

Kľúčové slová

proces, procesor, operačná pamäť, pamäť RAM, umiestňovanie procesov v pamäti, výpočet fyzickej adresy, riadenia voľných úsekov v pamäti, fragmentácia a defragmentácia pamäte

MOTIVÁCIA

V multiúlohových systémoch môžeme mať spustených viac programov (procesov). Všetky procesy, ktoré máme spustené musia byť umiestnené v operačnej pamäti. Niekedy je náš počítač pomalý. Je to preto, že je operačná pamäť má malú kapacitu? Vieme ako pracujú algoritmy, ktoré riadia pamäť?

CVIČENIE – DISKUTUJTE!

Ako operačný systém riadi pamäť? Spolupracuje procesor pri riadení pamäte so systémom?

Zostávajú niektoré časti pamäte voľné, alebo operačný systém dokáže obsadiť celú pamäť?

Ak zostanú niektoré úseky (fragmenty) voľné, vieme ich ešte využiť?

8.1 Úlohy riadenia pamäte

Operačný systém spolu s procesorom zabezpečujú riadenie operačnej pamäte (RAM). Hlavné úlohy riadenia pamäti sú:

- výpočet fyzickej adresy,
- efektívne riadenie umiestňovania procesov a tým aj riadenie voľných úsekov pamäte.,

VÝKLAD

Výpočet fyzickej adresy – programy sa v pamäti RAM dynamicky spracovávajú. Pri viacúlohových systémoch sa do pamäte RAM postupne prisúvajú programy, podľa toho ako sa vykonávajú. Preto sa môže stať, že sa náš program pri spracovávaní môže neskôr nachádzať v inej časti pamäte. Inak povedané programy (resp. ich procesy) môžu byť v rôznych časoch uložené v iných častiach pamäte.

Logická štruktúra programu a segmentovanie pamäte – programy často pozostávajú z modulov. Niektoré moduly obsahujú program (inštrukcie), niektoré údaje (dáta) a niektoré obsahujú dáta uložené v zásobníku. Každý modul je reprezentovaný časťou pamäte (pamäťovým segmentom).

Program, ktorý spracováva procesor používa rôzne adresy kódov (alebo inštrukcií) alebo ukazuje na rôzne adresy údajov (konštanty , premenné , údaje v poliach). Adresy, ktoré používa procesor sa nazývajú logické adresy a adresy, ktoré ukazujú na miesto v operačnej pamäti sa nazývajú fyzické adresy.

ZAPAMÄTAJ SI!

Fyzická pamäť je operačná pamäť (RAM). V operačnej pamäti musí byť uložené všetko, čo sa aktuálne spracováva (inštrukcie a dáta). Do pamäti RAM sa presúvajú dáta a inštrukcie z disku a prípadne pri nedostatku miesta v RAM sa znova odsúvajú na disk. Riadenie pamäte zabezpečuje tieto presuny.

Programy často pozostávajú z modulov. Niektoré moduly obsahujú program (inštrukcie), niektoré údaje (dáta) a niektoré obsahujú dáta uložené v zásobníku. Každý modul je reprezentovaný časťou pamäte (pamäťovým segmentom).

8.2 Výpočet fyzickej adresy

Program nepracuje so skutočnými fyzickými adresami, ale len s logickými. Je to preto, aby bolo možné efektívne využívať operačnú (RAM) pamäť. Logické adresy môžu nadobúdať hodnoty od 0 po max, kde max je poradie posledného (najvyššieho) miesta v pamäťovom segmente (pozri obrázok 1).

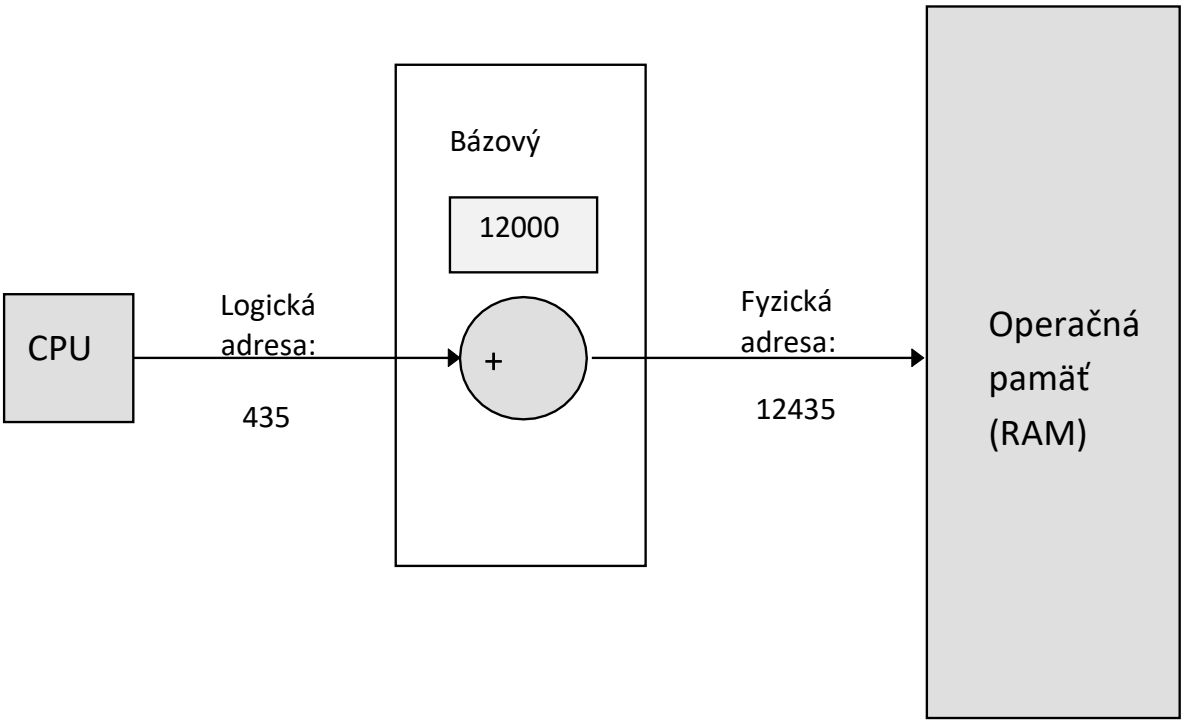
Aby efektívne pracoval počítač, treba pri behu programu, rýchlo prepočítavať logické adresy na fyzické a tým rýchlo ukazovať na miesta v operačnej pamäti. Preto procesory obsahujú hardvérovú jednotku riadenia pamätí (angl. Management Memory Unit, MMU) a operačný systém obsahuje softvérovú časť, ktorá riadi pamäte.

Jednotka riadenia pamäte prepočítava logické adresy na fyzické (a naopak). Bázový (relokačný) register obsahuje adresu, ktorá ukazuje začiatok časti pamäte (segmentu pamäte), kde sa nachádzajú inštrukcie nášho programu. K bázovému registru má prístup len operačný systém. Konkrétne poradie inštrukcie (logickú adresu) produkuje procesor.

Výpočet fyzickej adresy je daný súčtom logickej adresy a bázového registra (pozri obrázok 1). [1]

Nech má bázový register hodnotu 12000. Prvé adresovateľné miesto tohto pamäťového segmentu je 12000. Ak chceme prečítať alebo zapísať hodnotu, na ktorú ukazuje náš program (nech je táto hodnota 435), MMU pripočíta k hodnote bázového registra hodnotu logickej adresy 435. Potom je možné prečítať, alebo zapísať hodnotu na fyzické pamäťové miesto v pamäti RAM (t.j. na adresu 12435).

Pri riadení pamäte sa často používajú stránkované segmenty pamäte. Stránkovanie „rozdelí“ segment pamäte na rovnako veľké úseky tzv. stránky. Veľkosť stránky je zvyčajne 4KB.



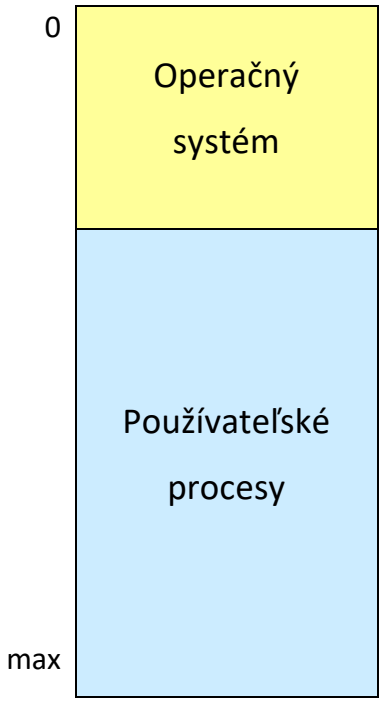
Obrázok 91
Výpočet fyzickej adresy

8.3 Umiestňovanie procesov v pamäti

Programy (procesy), ktoré chceme spracovávať pomocou procesora musia byť uložené v pamäti RAM. Ale aj operačný systém, musí byť uložený v pamäti RAM.

Pamäť je obyčajne rozdelená na dve časti - do jednej sa umiestni rezidentný operačný systém a do druhej sa umiestnia užívateľské procesy.

Operačný systém môže začínať hneď od adresy 0 (ako to ukazuje obrázok 2), alebo môže byť umiestnený na konci. [2]



Obrázok 92
Základný spôsob umiestňovania procesov v pamäti RAM

8.4 Prideľovanie súvislých procesov s rôznou dĺžkou do pamäti RAM

V praxi sa používa algoritmus prideľovania procesov do pamäti RAM. Tieto procesy obsadzujú úseky pamäte, ktoré sú súvislé a majú rôznu dĺžku.

ÚLOHA – RIEŠTE!

Predpokladajme, že na pridelenie pamäte čakajú procesy, ktoré sú pripravené v rade pripravených procesov (tabuľka 1). Z dôvodu ilustrovania dynamického spracovania procesov v systéme uvádzame časy spracovania procesu.

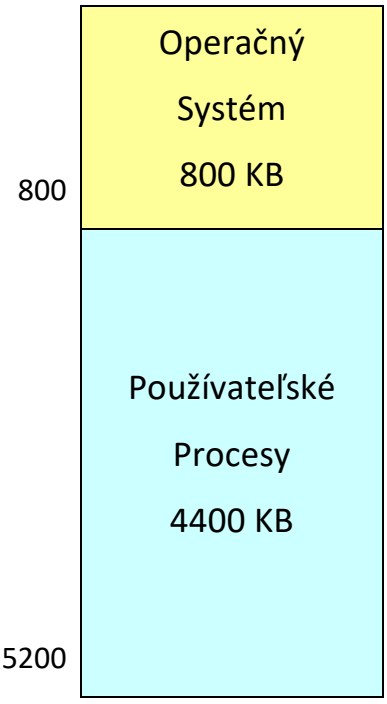
Na obrázku 3 vidíme situáciu v pamäti RAM, kde operačný systém zaberá 800 KB a priestor o kapacite 4400 KB je určený na používateľské procesy.

Riešme úlohu operačného systému a umiestňujme procesy do pamäte.

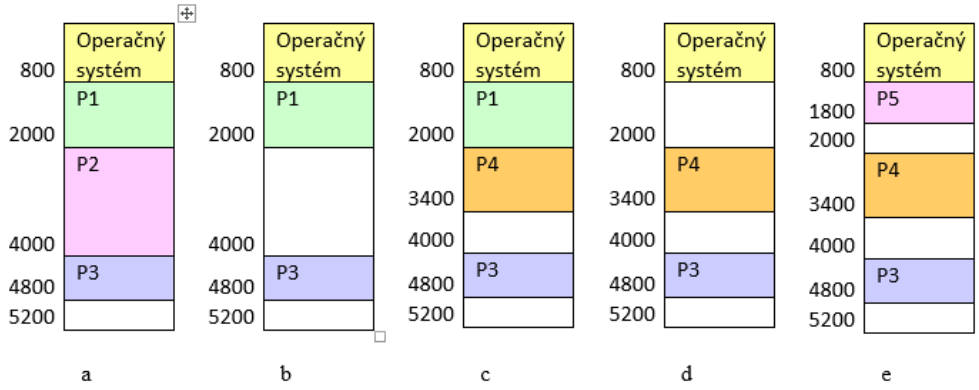
Postupne sa začnú vykonávať procesy P1, P2 a P3, ktoré sa umiestnili do pamäte (pozri obrázok 4a). Po uplynutí 5 sekúnd sa proces P2 odsunie z pamäte (pozri obrázok 4b). Vznikol priestor o kapacite 2000 KB a na toto miesto je možné umiestniť proces P4 (obrázok 4c). Za procesom P4 vznikol priestor o kapacite 600 KB a za procesom P3 je priestor o kapacite 400 KB, ale proces P5 nemôžeme umiestniť do pamäte, lebo tieto 2 úseky netvoria súvislú oblasť. Až po uplynutí ďalších 5 sekúnd (t.j. spolu 10) sa odsunie proces P1 (obrázok 4d) z pamäte a môže sa na jeho miesto umiestniť proces P5 (obrázok 4e)..

Tabuľka 10 Rad pripravených procesov

Proces	Požadovaná veľkosť pamäte	Požadovaný čas [s]
P1	1200 KB	10
P2	2000 KB	5
P3	800 KB	20
P4	1400 KB	8
P5	1000 KB	15



Obrázok 93
Prideľovanie pamäte procesom



Obrázok 94
Príklad prideľovania pamäte procesom

Na obr. 3 a 4 vidíme niekoľko situácií, ktoré môžu vzniknúť pri prideľovaní súvislých procesov do pamäte RAM. Rozmery úsekov sa menia dynamicky podľa veľkosti procesov, ktoré prichádzajú do pamäte. Úlohou systému je správne vybrať voľný úsek pamäte tak, aby sa do tohto priestoru vošiel prichádzajúci proces a zároveň bude efektívne využitá pamäť. [3]

Prideľovaním procesov do úsekov s premenlivou dĺžkou vznikajú časti pamäte, ktoré nie sú využiteľné (sú menšie ako prichádzajúce procesy). Tento problém sa dá riešiť striasaním pamäte.

Metóda prideľovania súvislých častí procesom tvorí základ pre techniku segmentovania pamäte.

ZHRNUTIE

Systém si dynamicky ukladá aktuálne informácie o voľných úsekoch pamäte. Z radu pripravených procesov vyberie proces a hľadá vhodný úsek pre jeho umiestnenie.

Ak systém nájde väčší úsek, proces sa tam umiestni a zvyšok úseku sa zaznamená do zoznamu voľných úsekov. Ak systém nenájde dostatočne veľký úsek, proces musí počkať, kým sa vhodný úsek uvoľní (skončí dostatočne veľký proces). Následne systém vyberie proces z radu pripravených procesov a umiesti ho do pamäte.

Keď proces skončí, jeho úsek sa vráti do zoznamu voľných úsekov. Následne systém urobí kontrolu, pomocou ktorej zistí či úseky v zozname neležia vedľa seba, aby sa mohli spojiť do väčšieho úseku.

8.5 Algoritmy vyhľadávania vhodného voľného úseku v pamäti

Na efektívnosť práce operačného systému vplýva optimálny výber vhodného voľného úseku pre umiestnenie procesu v pamäti, [1], [3].

Najčastejšie sa používajú algoritmy, ktoré prehľadávajú zoznam voľných úsekov:

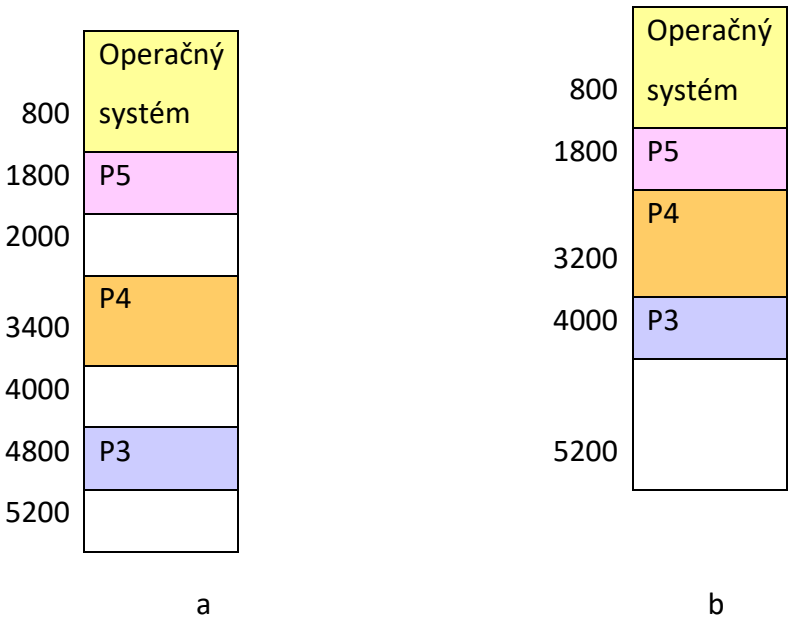
- Algoritmus Prvý vhodný (angl. First-fit, FF) - proces sa umiestni do prvého úseku v pamäti, ktorý je dostatočne veľký. Prehľadávanie sa zvyčajne začína na mieste, kde skončilo predchádzajúce hľadanie.

- Algoritmus Najlepší vhodný (angl. Best-fit, BF) – prehľadáva sa celý zoznam voľných úsekov. Pričom sa proces umiestni do pamäte do najmenšieho úseku, do ktorého sa vojde.

- Algoritmus Najhorší vhodný (angl. Worst-fit, WF) – prehľadáva sa celý zoznam voľných úsekov. Pričom sa proces umiestni do pamäte do najväčšieho voľného úseku. Je to vhodné vtedy, keď sú úseky v pamäti veľké a procesy malé. Po umiestnení takéhoto procesu ešte zostane dostatočne veľký priestor na umiestnenie ďalšieho procesu

8.6 Fragmentácia a defragmentácia pamäte RAM

Algoritmus prideľovania pamäte procesom do úsekov s premenlivou dĺžkou spôsobuje tzv. vonkajšiu fragmentáciu. Môžeme to vidieť na obrázku 4e, kde medzi procesmi P5, P4 a P3 vznikli nevyužiteľné úseky (fragmenty) pamäte. Súčet jednotlivých fragmentov dostatočne veľký na umiestnenie procesu, ale netvorí jeden celok. Preto do týchto fragmentov nemôžeme umiestniť proces. Fragmentáciu pamäte môžeme riešiť napr. striasaním pamäte. Striasaním sa snažíme spojiť neobsadené fragmenty pamäte do jedného väčšieho úseku. Pozri obrázok 5a ukazuje situáciu v pamäti pred striasaním a 5b po striasaní, [1], [3].



Obrázok 95
Príklad odstránenia fragmentácie striasaním pamäte

Striasanie sa uskutočňuje v dvoch krokoch: V prvom kroku sa presunú dáta a inštrukcie. V druhom kroku sa zmenia hodnoty v bázových registroch, aby korešpondovali s novým umiestnením procesov. Striasanie je časovo náročná operácia, preto je potrebný efektívny algoritmus striasania. Efektívny algoritmus defragmentácie sa vyznačuje tým, že:

- presúva najmenší počet procesov,
- presúva celkovú najmenšiu kapacitu procesov,
- čas presúvania procesov je najkratší.

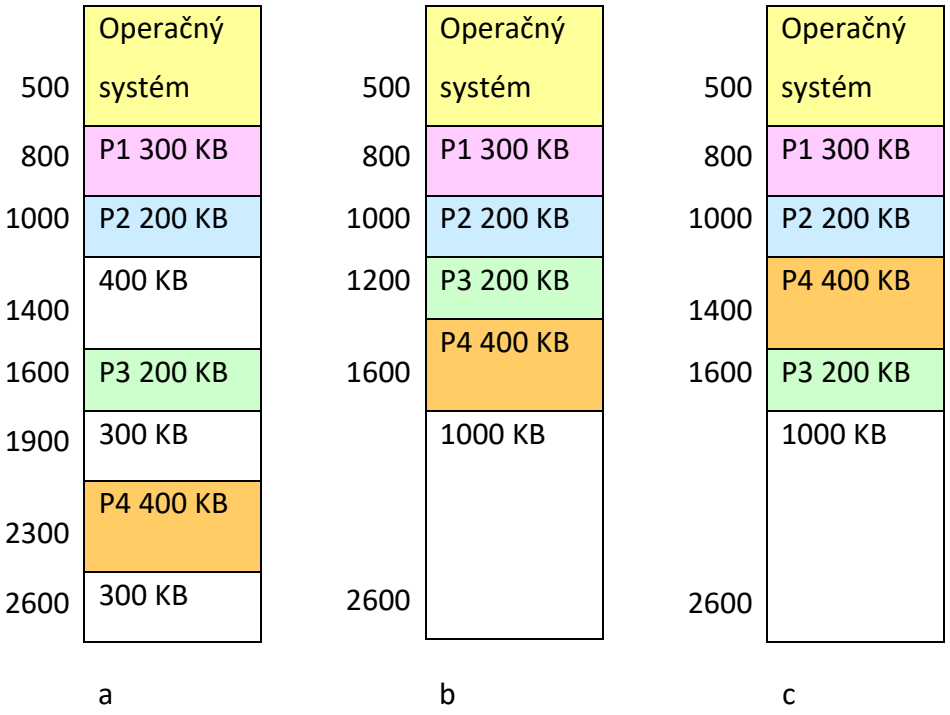
ÚLOHA – RIEŠTE!

Majme situáciu v pamäti RAM (pozri obrázok 6a). Riešme defragmentáciu pomocou 2 algoritmov. Porovnajme oba algoritmy.

Nech prvý deframentačný algoritmus presúva procesy tak, aby boli usporiadané v pamäti podľa doby príchodu do systému. Dobu príchodu vyjadrujú mená procesov (prvý prišiel P1, za ním P2, ...). Riešenie vidíme na obrázku b.

Nech druhý defragmentačný algoritmus) hľadá taký proces, ktorý má rovnakú veľkosť ako prvý voľný úsek (alebo najlepší vhodný proces) v pamäti. Operačný systém následne presunie proces do voľného úseku. Algoritmus opakuje celý postup dovtedy, kým nie je odstránená fragmentácia.

Pri algoritme na obrázku b, bolo potrebné presúvať 2 procesy (P3 a P4) s celkovou kapacitou 600 KB. Pri algoritme na obrázku c stačilo presunúť 1 proces s kapacitou 400 KB. Riešenie je znázornené na obrázku c.



Obrázok 96
Príklad porovnania dvoch defragmentačných algoritmov

ZHRNUTIE

Naučili sme sa, že hlavnou úlohou riadenie pamätí je vypočítať fyzickú adresu miesta v pamäti RAM a tiež riadiť umiestňovanie procesov v tejto pamäti. Preto je dôležité poznať voľné miesta v pamäti. Na poznania obsadených aj voľných úsekov pamäte je potom možné robiť defragmentáciu pamäte.

Tabuľka 11 Príklad3 - Rad pripravených procesov

Proces	Požadovaná veľkosť pamäte	Požadovaný čas [s]
P1	800 KB	10
P2	1400 KB	5
P3	2000 KB	20
P4	900 KB	8
P5	400 KB	15

Tabuľka 12 Príklad 4 – Obsadené o voľné úseky v operačnej pamäti

Proces	Požadovaná veľkosť pamäte
Operačný systém	800 KB
P1	900 KB
Voľný úsek	300 KB
P3	2000 KB
P4	900 KB
Voľný úsek	300 KB

ÚLOHA – RIEŠTE!

Majme operačnú pamäť s kapacitou 5200 KB. Nech operačný systém v pamäti zaberá kapacitu 800KB. Majme rad pripravených procesov, ktorý je uvedený v tabuľke2. Riešte úlohu umiestnenia procesov do pamäte. Nekreslite postupné umiestnenie procesov P1 až P5.

ÚLOHA – RIEŠTE!

Majme situáciu v pamäti, ako je uvedené v tabuľke 3. Riešte defragmentáciu pomocou 2 algoritmov z príkladu 2. Porovnajme oba algoritmy. Hľadajte alternatívne spôsoby umiestnenia. Nakreslite situácie v pamäti.

KĽÚČ K ÚLOHÁM

3	Riaďte sa postupom, ktorý sme sa naučili pomocou riešenia úlohy 1.
4	Riešte defragmentáciu pomocou 2 algoritmov z príkladu 2.

BIBLIOGRAFIA

[1] MARTINCOVÁ, P., GRONDŽÁK, K.: Operačné systémy. 1.vyd. Žilina: EDIS, 2004, 294 s. ISBN 80-8070-242-X.

9 RIADENIE SÚBOROV V OPERAČNOM SYSTÉME

Na počítači pracujeme so súbormi. *Vieme, čo obsahuje súbor, ktorý sme uložili na disk? Aké informácie si uchováva operačný systém o súboroch?*

CIEĽ

Čo budeme vedieť a čo budeme vedieť urobiť

- vytvoriť súbor,
- chápať pojem súbor,
- vedieť uložiť súbor na disk,
- zobrazovať informácie o súbore,
- chápať, ako sa časti súboru, ukladajú na disk,
- chápať zjednodušený model práce operačného systému pri ukladaní súborov na disk, resp. čítaní údajov zo súboru,
- chápať úlohy súborových systémov.

Súbory ukladáme na disk, alebo iné externé pamäťové médiá. Súbory sa na disk ukladajú do blokov, ktoré nemusia nasledovať za sebou. Pomocou Prieskumníka, alebo iných nástrojov vieme zobraziť informácie o súboroch. Ako to funguje vo vnútri systému? Kde má systém uložené informácie o súboroch? Ako je možné, že súbor sa môže skladať z viacerých častí a ako ich systém dokáže nájsť?

Kľúčové slová

súbor, obsah súboru, súborový systém, bloky súboru, informácie o súbore, prístupové práva k súboru

MOTIVÁCIA

Poznáme rôzne operačné systémy. Každý z nich umožňuje vytvárať súbory a ukladať ich na disk. Každý systém nám umožňuje robiť operácie so súbormi.

CVIČENIE – DISKUTUJTE!

Čo je to súbor? Čo obsahuje súbor?

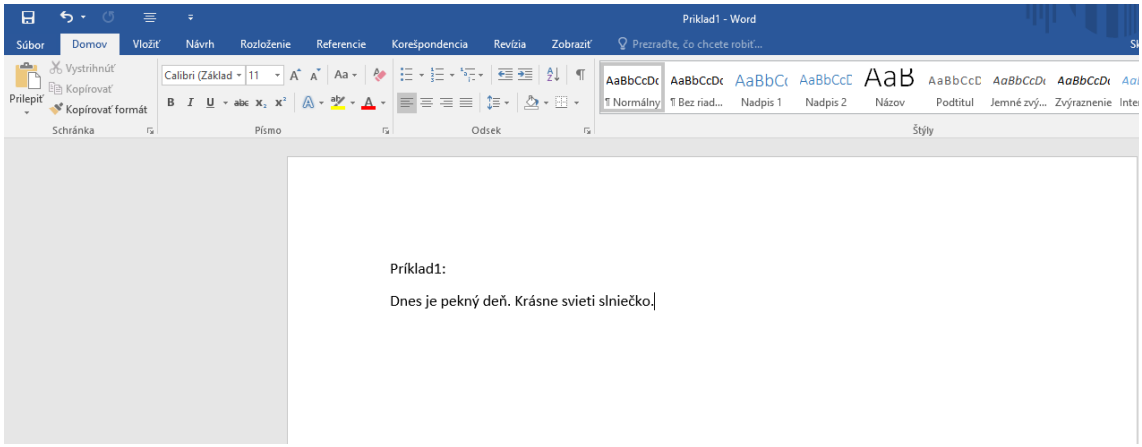
Ako uložíme súbor na disk? Aké operácie môžeme robiť so súborom?

Čo vieme povedať o súbore, ktorý máme uložený na disku?

Ako vyzerá súbor, ktorý je uložený na disku? Uloží sa súbor na disk ako jeden celok, alebo obsahuje niekoľko častí, ktoré nemusia byť uložené za sebou?

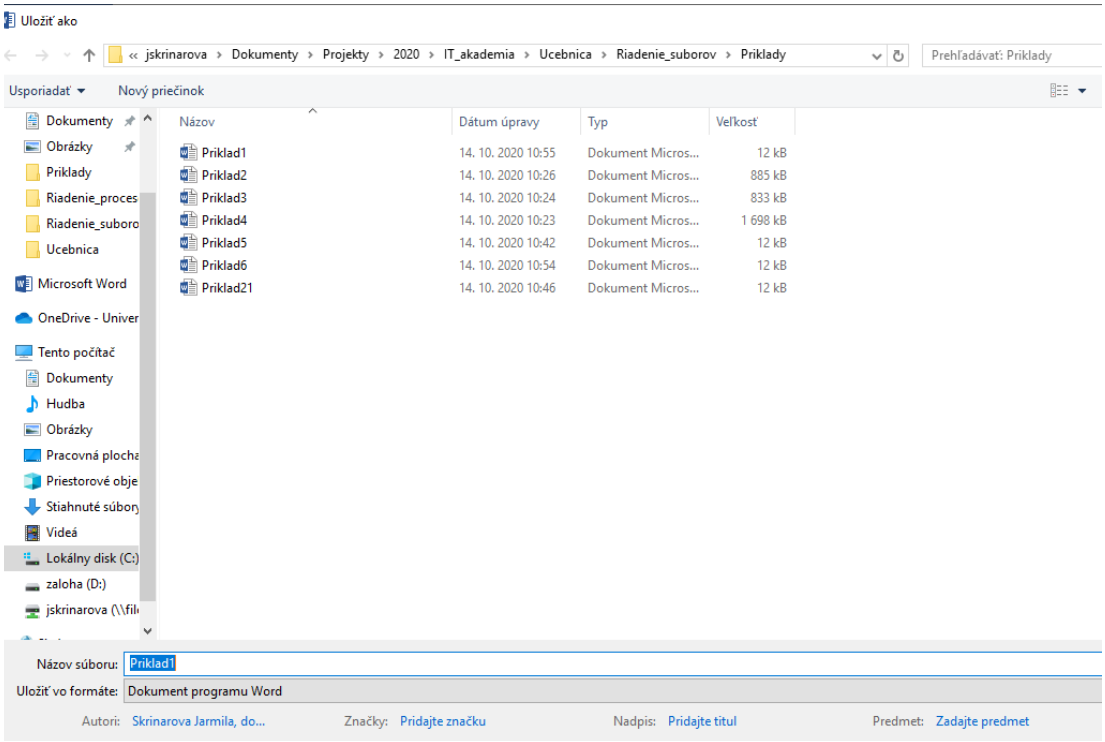
9.1 Súbor

Vieme, že keď pracujeme na počítači používame rôzne aplikácie. Napríklad používame textový editor Word, alebo tabuľkový kalkulátor Excel, alebo inú aplikáciu. Pomocou týchto aplikačných programov vytvárame súbory. Pozri obrázok 1, ktorý zobrazuje príklad tvorby súboru s názvom Priklad1 v aplikačnom programe Word.



Obrázok 97
Príklad tvorby súboru v aplikačnom programe Word

Takto vytvorený súbor ukladáme do určitého priečinku (adresára) na disk, alebo na iné externé pamäťové zariadenie. Pozri obrázok 2, ktorý ilustruje uloženie súboru s názvom Priklad1 do priečinka Priklady.



Obrázok 98
Príklad uloženia súboru na disk počítača

VÝKLAD

Na základe praktických príkladov, ktoré sú na obrázkoch 1 a 2, môžeme povedať čo je súbor. Súbor obsahuje informácie, ktoré spolu súvisia, sú uložené na disku a je označený menom. Zjednodušene povieme, že súbor je uložený na disku a máme na mysli, aj akékoľvek iné externé pamäťové médiá, na ktoré je možné ukladať súbory. Súbory zvyčajne ukladáme na pamäťové médiá (napr. pevný disk, USB kľúč, ...).

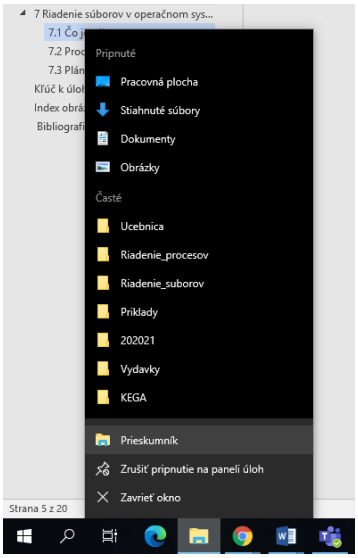
ZAPAMÄTAJTE SI!

Súbor obsahuje informácie, ktoré spolu súvisia. Súbor je uložený na disku a je označený menom.

CVIČENIE – POUŽITE!

Čo vieme povedať o súbore, ktorý máme uložený na disku?

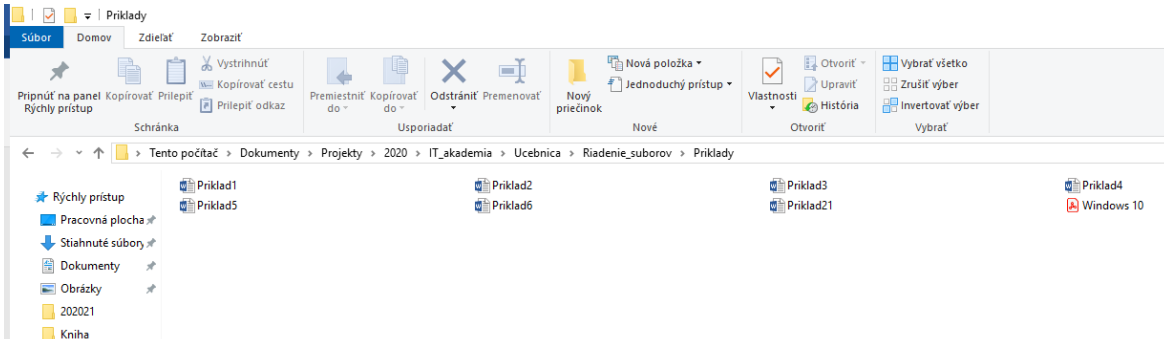
V operačnom systéme Windows 10, na prácu so súbormi, používame nástroj Prieskumník. Prieskumník otvoríme napríklad tak, že klikneme na žltú ikonu priečinka  , ktorá sa nachádza na spodnej lište. Ak chceme otvoriť ďalšie okno prieskumníka, klikneme na túto ikonu pravým tlačidlom a zvolíme Prieskumník.



Obrázok 99
Spôsob otvorenia ďalšieho okna nástroja Prieskumník

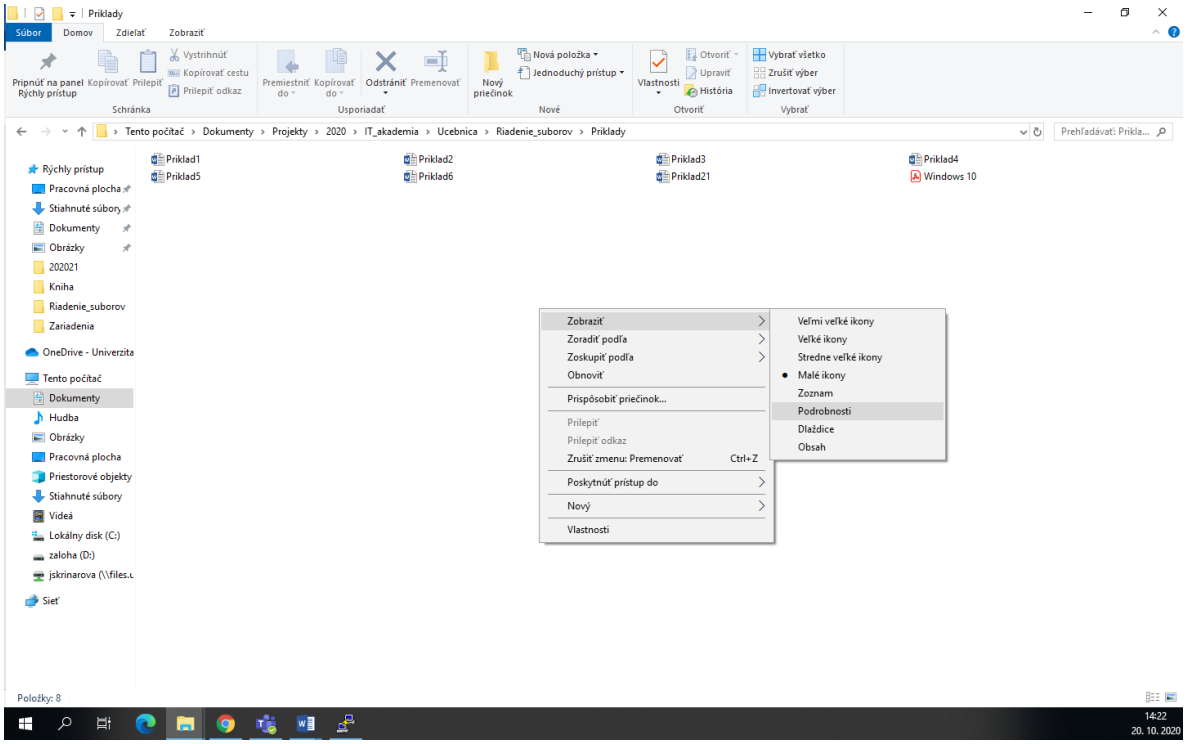
Pomocou nástroja Prieskumník zobrazíme súbory.

Aké ďalšie údaje o súboroch môžeme zobraziť Prieskumník?



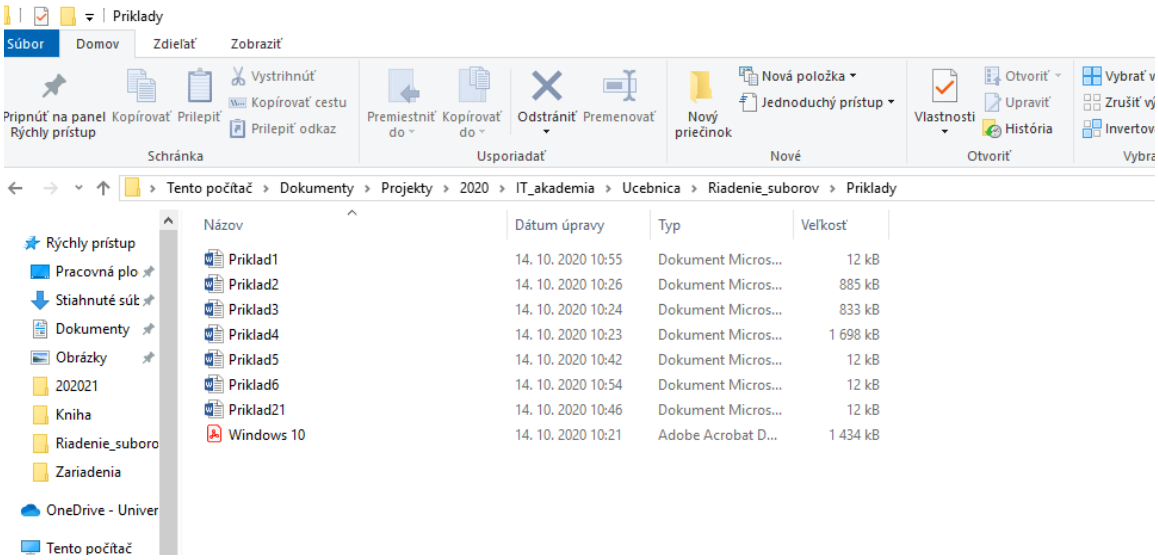
Obrázok 100
Obsah aktuálneho priečinka vo Windows 10

Klikneme pravým tlačidlom na aktuálne okno a z menu vyberieme položku Zobraziť a následne položku Podrobnosti.



Obrázok 101
Spôsob zobrazenie podrobností o súboroch

Výsledok vidíme vo výpise podrobností o súboroch.



Obrázok 102
Výpis podrobností o súboroch v systéme Windows 10

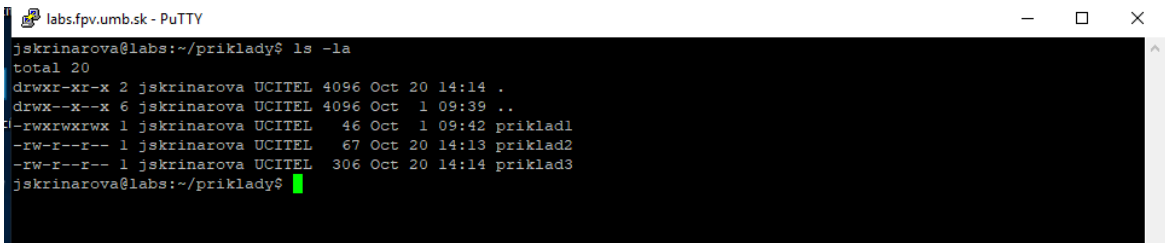
CVIČENIE – POUŽITE!

Zobrazujú všetky operačné systémy rovnaké informácie o súboroch?

Skúsme preto v operačnom systéme Linux zobraziť obsah aktuálneho adresára. Na zobrazenie výpisu môžeme použiť príkaz `ls.`

Následne, ak chceme zobraziť podrobnosti o súboroch, použijeme príkaz `ls - la.` Výpis vidíme na obrázku.

Obrázok 103
Obsah aktuálneho priečinka v Linuxe



Obrázok 104
Výpis podrobností o súboroch v systéme Linux

CVIČENIE – ANALYZUJTE!

Aké informácie o súboroch sme získali pri používaní systému Windows a systému Linux ?

Pozrime sa opäť na obrázky.

V ***systéme Windows*** sme o každom súbore zistili meno súboru, dátum vytvorenia resp. poslednej zmeny súboru, typ súboru a jeho veľkosť v kilobajtoch.

V ***systéme Linux*** sme sa o každom súbore dozvedeli (čítajme zľava):

Prvý znak – (pomlčka) označuje súbor.

Ďalších 9 znakov označuje prístupové práva k súboru.

Meno vlastníka súboru a skupiny, do ktorej vlastník patrí.

Veľkosť súboru v bytoch.

Dátum a čas poslednej zmeny súboru.

Názov súboru.

Vidíme, že systémy vypisujú podobné informácie o súboroch. Linux navyše zobrazuje informácie o prístupových právach k súborom.

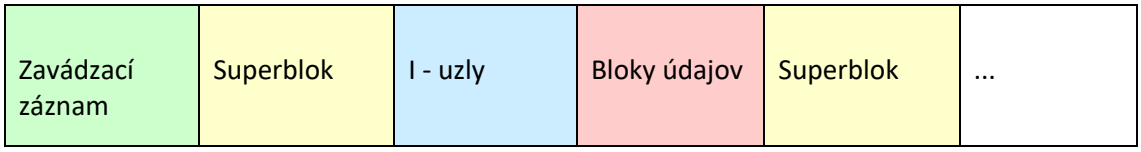
9.2 Súborový systém

MOTIVÁCIA

Zaujímavá je otázka: ***„Ako si systémy uchovávajú informácie o súboroch?“***

VÝKLAD

Operačné systémy si uchovávajú informácie o súboroch v tzv. súborových systémoch. Súborové systémy uchovávajú mená všetkých súborov (alebo ich čísla), ich veľkosti, dátum a čas ich vytvorenia, resp. zmeny súboru, a prístupové práva (Linux) [1]. Okrem týchto informácií ukazujú na bloky údajov, v ktorých sú uložené všetky bloky súboru. V operačnom systéme Linux sa takýto súborový systém volá ***EXT*** (pozri obrázok 9) [2].



Obrázok 105
Celková štruktúra súborového systému EXT.

Na začiatku súborového systému sa nachádza zavádzací (angl. boot) záznam. Úlohou zavádzacieho záznamu je pri štarte systému zaviesť operačný systém do pamäte počítača.

Záznam zvyčajne zaberá 1 sektor, preto sa často nazýva boot sektor. Po zavádzacom zázname sa opakuje základná štruktúra súborového systému.

Základná štruktúra súborového systému EXT [3]:

- superblok,
- tabuľka i- uzlov,
- bloky údajov (v údajových blokoch sa nachádzajú skutočné obsahy súborov).

Každý superblok obsahuje informácie o častiach, ktoré za ním nasledujú. Ide najmä o tieto informácie:

- počet i - uzlov,
- počet blokov,
- číslo prvého bloku,
- veľkosť bloku.

Informácie o každom súbore sú uložené v štruktúre, ktorá sa volá ***i – uzol***.

Základné informácie o súbore obsahuje tzv. i - uzol:

I - uzol obsahuje informácie o súbore alebo o priečinku. Typické informácie obsahujú typ súboru (najmä či ide o súbor alebo adresár) a prístupové práva k súboru, kto je vlastník súboru a do ktorej skupiny vlastník patrí, dátum a čas vytvorenia, alebo poslednej zmeny súboru, zoznam blokov údajov a odkazy na tieto údaje (časti súborov). Pozri obrázok 10 [3].

Typ a prístupové práva
Vlastník
Skupina vlastníkov
Časové informácie (dátum a čas)
Veľkosť
Počet odkazov
Priame odkazy na bloky údajov ...
Nepriame odkazy na bloky údajov jednocestné
Nepriame odkazy na bloky údajov dvojcestné
Nepriame odkazy na bloky údajov trojcestné

Obrázok 106
Obsah i - uzla súboru

Súbory sú uložené v blokoch údajov. Bloky údajov majú vždy rovnakú veľkosť. Tieto bloky zvyčajne nejdú pekne za sebou, ale sú rozhádzané po disku. Preto i – uzol obsahuje odkazy na tieto bloky. Priame odkazy obsahujú adresy na konkrétne bloky. Nepriame jednocestné odkazy ukazujú na tabuľku, ktorá obsahuje priame odkazy. Nepriame odkazy dvojcestné ukazujú na tabuľku, ktorá obsahuje nepriame odkazy a tie ukazujú na priame odkazy a tie už na konkrétne bloky údajov. Priečinky sú v skutočnosti súbory, kde sú ku každému názvu súboru priradené čísla i - uzlov.

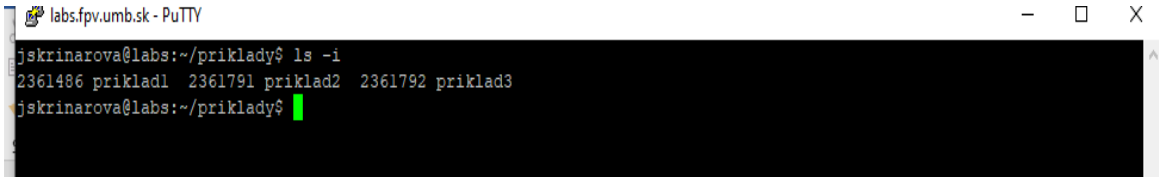
ZAPAMÄTAJ SI!

Súborový systém uchováva informácie o každom súbore. Najdôležitejšie informácie sú ukazovatele (adresy), ktoré ukazujú kde na disku na nachádzajú jednotlivé časti súboru. Súborový systém ďalej obsahuje ako veľkosť, dátum a čas vytvorenia súboru, meno vlastníka a prístupové práva.

CVIČENIE 3 – POUŽITE!

V operačnom systéme Linux zistite čísla i – uzlov súborov vo vašom priečinku!

Riešenie úlohy je jednoduché (pozri obrázok 11). Použijeme príkaz `ls -i`.



Obrázok 107
Zistenie čísla i – uzla súboru

ÚLOHA – RIEŠTE!

Majme zjednodušený súborový systém EXT (pozri obrázok 10), ktorý používa len priame odkazy. Nech má blok na disku veľkosť 512 B a odkaz má veľkosť 2 B. Aká je maximálna veľkosť súboru [4] ?

Ak má odkaz veľkosť 2 B, t.j. 16 bitov, potom dokážeme adresovať maximálne 2^{16} blokov, t.j. 65536 blokov. Ak použijeme len priame odkazy, potom maximálna veľkosť súboru je 512*65536 = 32 MB.

ZHRNUTIE

Naučili sme sa, že súbory vytvárame jednoducho pomocou aplikácií. Ukladáme ich na disk. Súbory sú na disku uložené v blokoch, ktoré majú rovnakú veľkosť (napr. 512 B, 4kB, ...). Naučili sme sa, že bloky údajov často nie sú uložené za sebou. Vieme, že súborový systém uchováva informácie o jednotlivých súboroch. Na príklade súborového systému EXT operačného systému Linux sme porozumeli, ako systém dokáže ukázať na (adresovať) všetky bloky, v ktorých sa súbor nachádza.

ÚLOHA – RIEŠTE!

Majme súborový systém EXT. Kde sú uložené mená súborov a im prislúchajúce čísla i – uzlov? Kde nájdeme informácie o týchto súboroch, ktoré vypisuje systém?

ÚLOHA – RIEŠTE!

Čo myslíte, prečo sa meno súboru neukladá do i – uzla?

KĽÚČ K ÚLOHÁM

2	Mená súborov a im prislúchajúce čísla i – uzlov sú obsahom súborov, ktoré poznáme pod pojmom priečinok. Informácie o súboroch, ktoré vypisuje systém môžeme nájsť v i – uzloch.
3	Mená súborov a im prislúchajúce čísla i – uzlov sú v priečinkoch. Vyhľadávanie informácií (a odkazov na bloky údajov súborov), ak by sme vyhľadávali podľa mena súboru, by bolo oveľa zdĺhavejšie. Ďalším dôvodom je, že v i – uzle by mohli zaberať veľa miesta (pri dlhých menách súborov).

BIBLIOGRAFIA

[1] STALLINGS, W.: Operating Systems. Prentice Hall, 2005. ISBN 0-13-147954-7

[2] HEJDA, V.: Souborové systémy v Linuxu. Linuxexpres. 2011.
https://www.linuxexpres.cz/blog/souborove-systemy-v-linuxu

[3] Systém souborů v Unixu
https://cs.wikipedia.org/wiki/Syst%C3%A9m_soubor%C5%AF_v_Unixu

[4] ALVISI, L.:ihttps://www.cs.utexas.edu/~lorenzo/corsi/cs372/06F/hw/11sol.html

10 VIRTUÁLNE POČÍTAČE A VIRTUALIZAČNÉ NÍSTROJE

Zatiaľ sme si v učebnici hovorili o počítačoch ako o fyzických zariadeniach. Opakom slova fyzický je slovo virtuálny. Odpovedzme si potom na otázku: *„Dovoľujú súčasné technológie existujú virtuálnych počítačov?“*.

CIEĽ

Čo sa naučíme a čo si precvičíme

- vysvetliť podstatu virtualizácie,
- ozrejmiť si rozdiely v typoch virtualizácie,
- demonštrovať rozdiel medzi emuláciou a virtualizáciou,
- oboznámiť sa s vrstvami virtualizácie,
- rozlišovať druhy virtualizácie na báze hypervízora,
- prakticky porovnať emuláciu s virtualizáciou,
- vytvoriť virtuálny počítač s použitím vhodného hypervízora.

Kľúčové slová

virtualizácia, virtuálny počítač, hypervízor, emulácia, typy virtualizácie, vrstvy virtualizácie, Hyper-V, VirtualBox

MOTIVÁCIA

Pojem post-PC éra prvýkrát použil David D. Clark z Massachusetts Institute of Technology v roku 1999. [3]

Žijeme v *ére post-PC*, kde sa osobné počítače stávajú menej dôležitými, lebo ich stretávame. Mať k dispozícii osobný počítač ako taký nie je pre jednotlivca nevyhnutné, pretože mobilné telefóny, tablety, hodinky, atď. plnohodnotne ponúkajú podobné služby. Prostredníctvom internetu sa pripájajú k výkonným počítačom umiestneným v geograficky vzdialených dátových centrách, ktoré poskytujú rôzne softvérové služby (napr. e-mailové služby, chaty, IpTV, sociálne siete), úložiská údajov (napr. Google Drive, Box, OneDrive, DropBox), ale aj na mieru vytvorené počítače (napr. MS Azure, Amazon EC, IBM Cloud). Je zaujímavé vedieť, ako takéto dátové centrá fungujú.

CVIČENIE – DISKUTUJTE!

Čo využívate na posielanie e-mailov?

Aké typy sociálnych sietí využívate?

Kde sa nachádzajú vaše e-maily, príspevky na FaceBooku alebo Instagrame?

Poznámka pre učiteľa:

Predpokladáme, že žiaci už využívajú niektoré cloudové služby, o ktorých netušia, že sú to cloudové služby. Skúste rozvinúť debatu na tému, aké služby používajú napr. v smartfónoch. Skúste ich poznačiť na tabuľu a ukázať, že sú to služby cloudu.

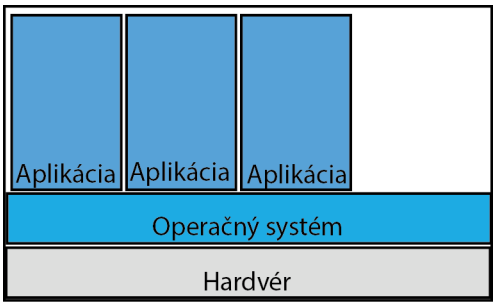
Iná problematika, ktorá sa nás týka možno viac, je dilema, ako odskúšať nejaký nový operačný systém bez toho, aby sme si museli zháňať nejaký „voľný" počítač, alebo aby sme museli vytvárať na disku novú partíciu s možnosťou výberu operačného systému pri bootovaní počítača, alebo dokonca preinštalovať svoj počítač. Riešením je vytvoriť si *„počítač v počítači"*. Môžeme si to predstaviť tak, že si spustíme v našom počítači (*hostiteľský počítač, ang. host*) aplikáciu, ktorá bude predstavovať plnohodnotný počítač (*hosťujúci počítač, ang. guest*). Keď budeme pracovať v okne aplikácie, tak budeme používať plnohodnotný hosťujúci počítač, ktorý však nie je fyzický, ale je len virtuálny. Mimo okna aplikácie budeme potom pracovať s hostiteľským počítačom.

Poznámka pre učiteľa:

Zvážte predvedenie vlastného virtuálneho počítača, ako motivácie pre študentov. Napríklad im ukážte počítač s OS Windows, v ktorom beží počítač s OS Linux (alebo naopak).

VÝKLAD

Uvedené možnosti by neboli možné realizovať bez existencie *virtualizácie*, *virtualizácie hardvéru*. Práve *virtualizácia* a výrazný *nárast výkonu počítačov* umožnil nástup post-PC éry. Virtualizácia je proces, pri ktorom sa vytvára virtuálna verzia fyzického objektu. V našom prípade je objektom počítač.



Obrázok 108
Schematické zobrazenie štruktúry fyzického počítača

Fyzický číslicový počítač sa skladá z hardvéru, operačného systém a nainštalovaných aplikácií (Obrázok 1). Význam slova fyzický treba v tomto prípade chápať ako hmotný. Pod pojmom fyzický počítač budeme ďalej v texte rozumieť buď osobný počítač alebo server. Cieľom virtualizácie je vytvoriť virtuálny hardvér. Virtualizácia hardvéru je vytváranie virtuálnej verzie fyzického hardvéru.

10.1 Virtualizácia

ZAPAMÄTAJTE SI!

Virtualizácia je simulácia softvéru alebo hardvéru, na ktorom beží ďalší softvér. Toto simulované prostredie sa nazýva virtuálny stroj (Virtual Machine – VM).

Virtualizácia môže mať veľa podôb, existuje:

- virtualizácia aplikácií,
- virtualizácia operačných systémov,
- virtualizácia hardvéru.

Typickým príkladom virtualizácie aplikácií je *Java Virtual Machine (JVM)*, ktorý je prostredníkom medzi aplikačným kódom, napísaným v jazyku Java a operačným systémom. Aplikácia je potom prenositeľná medzi všetkými platformami, na ktorých je nainštalovaný JVM.

Pri virtualizácii operačných systémov je k dispozícii virtuálna implementácia rozhrania operačného systému, ktoré možno použiť na spustenie aplikácií napísaných (skompilovaných) pre rovnaký operačný systém. Môžeme hovoriť o čiastočnej virtualizácii (podrobnejšie to bude vysvetlené ďalej).

Ak by sme sa pozreli na virtualizáciu z pohľadu spomínaných dátových centier, môžeme vo všeobecnosti virtualizáciu vnímať ako vytváranie virtuálnych výpočtových zdrojov (virtualizácia hardvéru) nad fyzickými výpočtovými zdrojmi (fyzickým hardvérom).

ZAPAMÄTAJTE SI!

Virtualizácia hardvéru je schopnosť modelovať hardvér pomocou softvérových metód. Virtualizačná technológia umožňuje spustiť viac operačných systémov na jedinom reálnom fyzickom počítači s izolovanými oddelenými výpočtovými procesmi s vyhradenými logickými zdrojmi, s využitím čiastočnej kapacity spracovania, čiastočnej kapacity RAM a časti súborového subsystému zo spoločného fondu.

Podľa typu virtualizácie výpočtových zdrojov existuje niekoľko možných *vrstiev virtualizácie hardvéru*. Aj keď si ich všetky nevysvetlíme, aspoň ich vymenujeme:

1. virtualizácia serverov,
2. virtualizácia osobných počítačov,
3. virtualizácia úložísk (vytvorenie jedného dátového úložiska z viacerých fyzických častí – DAS, NAS, SAN),
4. virtualizácia sietí (napr. vytváranie lokálnych virtuálnych sietí – VLAN),
5. manažovanie virtualizácie (horizontálne a vertikálne rozdelenie služieb do nezávisle spravovaných celkov),
6. virtualizácia prezenčnej vrstvy (napr. pri používaní tenkých klientov),
7. virtualizácia aplikácií (virtualizuje sa len aplikácia a nie celý operačný systém). [2]

Ďalej sa zameriame už len na vytváranie virtuálnych strojov, čo reprezentujú: *virtualizácia serverov* a *virtualizáciu osobných počítačov* (virtualizáciu na strane klienta).

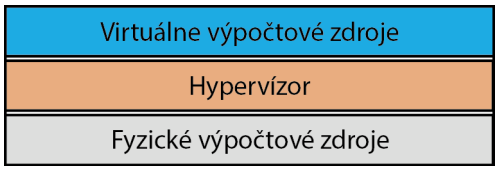
ZAPAMÄTAJ SI!

Virtuálny počítač je softvérová implementácia fyzického počítača bežiaceho v manažéri virtuálnych strojov.

10.2 Správca virtuálnych strojov

Softvér, ktorý umožňuje vytvárať virtuálne počítače, nazývame *virtualizačným softvérom*, ktorý sa tiež odborne nazýva *hypervízor* alebo *správca virtuálnych strojov* (ang. *Virtual machine manager* – *VMM*). Hypervízor umožňuje vytvárať, manažovať a spúšťať virtuálny počítač v prostredí fyzického počítača. Je to špeciálna softvérová vrstva, ktorá sprístupňuje fyzické výpočtové zdroje vo forme požívateľom požadovaných virtuálnych výpočtových zdrojov (Obrázok 2).

V roku 1974 boli definované základné vlastnosti virtuálneho stroja: *efektívnosť*, *riadenie zdrojov*, *rovnocennosť* [3]. V súčasnosti sú už zastaralé.



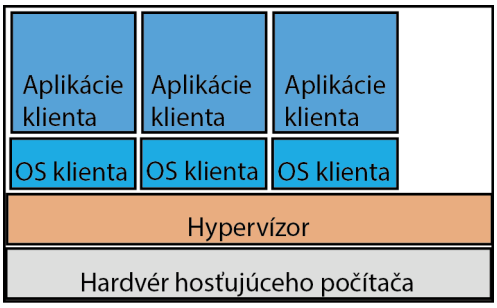
Obrázok 109
Vrstvy pri virtualizácii výpočtových zdrojov

Hypervízor *emuluje* (napodobňuje, predstiera) kompletne hardvér fyzického číslicového počítača. Obsahuje virtuálnu CPU, virtuálnu operačnú pamäť, virtuálne vstupno-výstupné zariadenia, atď. Podľa spôsobu prístupu hypervízorov k hostiteľským počítačom rozlišujeme dva základné *typy hypervízorov*:

1. *natívne* (alebo prirodzené, typu 1),
2. *hosťované* (alebo typu 2).[2]

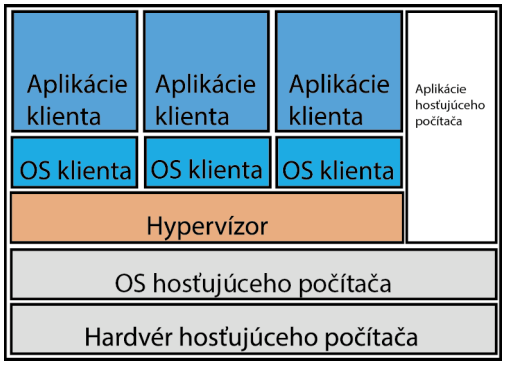
[2] Môžeme sa stretnúť, najmä pri operačnom systéme Linux s *vnoreným hypervízorom* (embedded hypervisor).

Natívny hypervízor je virtualizačný softvér, ktorý nepotrebuje k svojej funkčnosti úplný hostiteľský operačný systém a beží priamo na hardvéri hostiteľského počítača (Obrázok 3). Priamo nad hardvérom hostiteľského počítača beží hypervízor. Pomocou neho sú vytvárané virtuálne stroje vybavené vlastným operačným systémom a aplikáciami. Do tejto skupiny hypervízorov patria napr. *VMware vSphere / ESXi*, *KVM*, *Red Hat Enterprise Virtualization (RHEV)*, *Xen / Citrix XenServer*, *Microsoft Windows Server 2012 Hyper-V*, atď. Často sa používa pri virtualizácii serverov.



Obrázok 110
Schematické zobrazenie virtualizácie s natívnym hypervízorom

Hosťovaný hypervízor je nainštalovaný v hostiteľskom operačnom systéme. Nad hardvérom hostiteľského počítača beží jeho úplný operačný systém. V operačnom systéme je spustený hypervízor. Okrem hypervízora môžu pri tomto riešení na hostiteľskom počítači bežať aj iné aplikácie. V hypervízore sú vytvorené virtuálne stroje. Každý virtuálny stroj má nainštalovaný vlastný operačný systém a aplikácie (Obrázok 4). Do tejto skupiny hypervízorov patria napr. *Oracle Virtual Box*, *Oracle VM for x86*, *Parallels Desktop*, *QEMU*, *Solaris Zones*, atď.



Obrázok 111
Schematické zobrazenie virtualizácie s hosťovaným hypervízorom

Hypervízor potrebuje časť výpočtového výkonu fyzického počítača a časť súvisiacich zdrojov fyzického počítača. Navyše, hypervízor istým spôsobom znižuje výkon virtuálneho stroja oproti takému istému reálnemu stroju. Je to spôsobené dodatočným spracovaním (v zmysle „inštrukčného prekladu“), ktorý vykonáva. Pri porovnaní oboch typov hypervízorov, dosahuje prvý typ lepší výkon a druhý typ lepšiu hardvérovú kompatibilitu. Momentálne významné softvérové firmy ponúkajú širokú škálu hypervízorov prvého aj druhého typu. Výber toho správneho virtualizačného nástroja (hypervízora) je podmienený viacerými kritériami, ktorými sú najmä cena, druh licencie, spôsob použitia, hardvér a operačný systém hosťujúceho počítača, zložitosť inštalácie a údržby.

Poznámka pre učiteľa:

Pred samotnou inštaláciou ktoréhokoľvek hypervízora skontrolujte na počítačoch, či procesor podporuje virtualizáciu a či je v BIOS-e podpora aktivovaná. Prípadne študentom stručne objasnite čo v BIOS-e kontrolujete a aktivujete.

Počas inštalácie zvážte, ktorým detailom sa budete venovať a ktoré pojmete len ako „čiernu skrinku“.

Správca virtuálnych strojov v operačnom systéme Windows

Spoločnosť Microsoft ponúka vo svojom OS systéme Windows, vlastný hypervízor, ktorý sa nazýva *Hyper-V*. Hypervízor, môžeme alebo nemusíme mať aktivovaný.

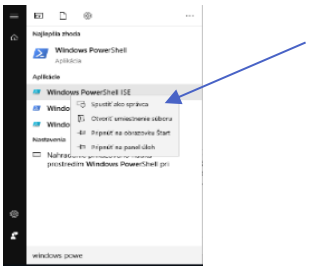
Poznámka pre učiteľa:

Cvičenie môžete robiť len s použitím OS Windows. Ak používate Linux, prejdite na ďalšiu úlohu. Aby ste mohli túto úlohu so študentmi riešiť, potrebujete minimálne[3]:
– OS Windows 10 Enterprise, Pro alebo Education
– 64-bitový procesor so SLTA (Second Level Address Translation), čo Intel nazýva Extended page tables (EPT) a AMD Nested page tables (NPT),
– podporu procesora pre VM Monitor Mode Extension,
– minimálne 4 GB pamäte.

Aktivovať Hyper-V môžeme tromi spôsobmi, *vždy v administrátorskom móde*:

1. buď cez konzolu *PowerShell*,
2. alebo cez konzolu *CMD*,
3. alebo pomocou okna *Nastavenia*.

Alternatíva 1 – Aktivovanie hypervízora Hyper-V cez konzolu PowerShell



Obrázok 112 Spustenie konzoly PowerShell-u

[3] https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v

1. Po *kliknutí* na ikonu *Štart* začneme písať *PowerShell*. Objaví sa nám ponuka, z ktorej vyberieme *Windows PowerShell* a pravým tlačidlom myšky otvoríme podmenu, v ktorom zvolíme možnosť *Spustiť ako správca* (Obrázok 5).

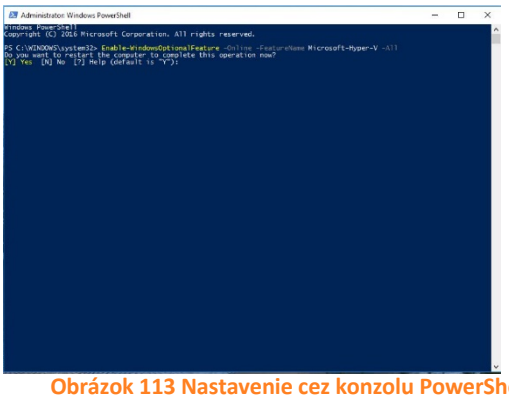
> **Poznámka pre učiteľa:**
> Ak ste ešte nevysvetľovali študentom, ako pracovať v prostredí OS Windows v režime administrátora, tak najskôr študentom ukážte, ako sa otvárajú aplikácie v režime administrátora, aby sa potom mohli zamerať na preberané učivo.

2. Otvorí sa okno aplikácie Windows PowerShell, v ktorom zadáme príkaz:

```
ENABLE-WINDOWSOPTIONALFEATURE -ONLINE -FEATURENAME MICROSOFT-HYPER-V -ALL
```

Po vykonaní príkazu sa aktivuje virtualizačný nástroj Hyper-V, ktorý môžeme začať používať až po reštartovaní počítača (Obrázok 6).

Alternatíva 2 – Nastavenie cez konzolu CMD

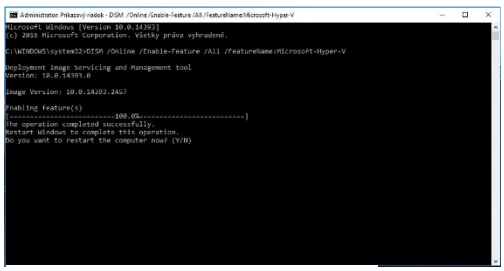


Obrázok 113 Nastavenie cez konzolu PowerShell

1. Podobne ako pri použití aplikácie PowerShell, po kliknutí na ikonu *Štart* začneme písať *CMD*. Objaví sa nám ponuka *Príkazový riadok*, ktorú potvrdíme pravým tlačidlom myšky a zvolíme možnosť *Spustiť ako správca*.
2. Otvorí sa nám okno Príkazového riadku, v ktorom zadáme príkaz (Obrázok 7):

```
DISM /ONLINE /ENABLE-FEATURE /ALL /FEATURENAME:MICROSOFT-HYPER-V
```

3. Počítač budeme musieť reštartovať.

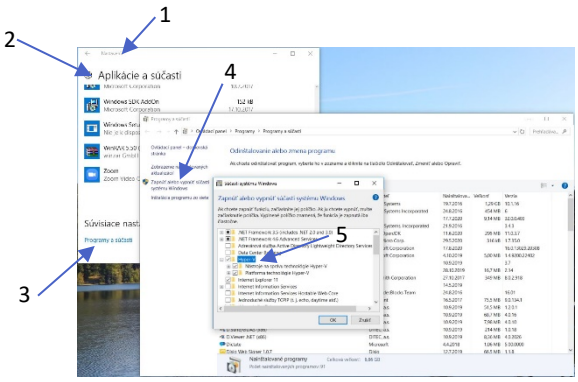


Obrázok 114 Nastavenie cez konzolu CMD

Alternatíva 3 – Nastavenie pomocou okna Nastavenia

1. Otvoríme si okno *Nastavenia*.
2. Vyberieme z ponuky *Aplikácie a súčasti*.

3. Otvorí sa nám okno Aplikácie a súčasti a na jeho samom spodku klikneme na odkaz *Programy a súčasti*.
4. Otvorí sa okno, kde z menu na jeho ľavej strane zvolíme kliknutím možnosť *Zapnúť alebo vypnúť súčasti systému Windows* (Obrázok 8).
5. Otvorí sa ďalšie okno, v ktorom nájdeme zaškrtávacie políčko *Hyper-V* a zaškrtneme ho. V prípade, že je zaškrtnuté, tak Hyper-V už na pozadí beží. (Týmto spôsobom môžeme Hyper-V aj deaktivovať, napr. keď budeme riešiť nasledujúci príklad použitia VirtualBoxu.)
6. Počítač budeme musieť znovu reštartovať.



Obrázok 115 Nastavenie cez okno Nastavenia

Poznámka pre učiteľa:

Ak nechcete ukázať študentom všetky tri možnosti, tak použitie tretí spôsob, lebo nemusíte vysvetľovať, čo znamenajú príkazy, ktoré používate v prvých dvoch prípadoch.

CVIČENIE – POUŽITE!

Ak máte k dispozícii počítač s OS Windows 10, aktivujte a deaktivujte v tomto počítači hypervízor *Hyper-V*.

Správca virtuálnych strojov VirtualBox

Hypervízor *VirtualBox* je od spoločnosti Oracle. Tento hypervízor je vhodný tak pre hosťujúci operačný systém Windows, ako aj pre hosťujúci operačný systém Linux. Ukážeme si postup inštalácie pre hosťujúci počítač, na ktorom je nainštalovaný operačný systém Windows.

Poznámka pre učiteľa:

Po tejto úlohe siahnite, ak nemáte k dispozícii počítač s potrebným OS Windows. Ak máte k dispozícii počítače s nainštalovaným OS Linux, upravte zadanie pre konkrétnu verziu OS Linux.

V prípade, že chcete len ukázať virtualizáciu, siahnite len po jednej z úloh 1 alebo 2.

Inštalácia VirtualBoxu

1. Potrebný inštalačný balík *VirtuálBoxu* (cca. 200 MB) získame z jeho webovej stránky.[4] Na stránke sa nachádza viacero inštalácií, ktoré sa viažu ku konkrétnemu typu operačného systému hostiteľského počítača (Windows, OS X, Linux, Solaris). Zvolíme si inštaláciu hypervízora VirtualBox pre OS Windows.

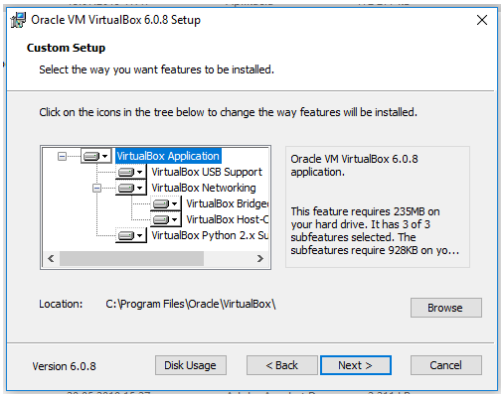
Obrázok 116 Stránka s inštalačnými balíkmi hypervízora VirtualBox pre rôzne operačné systémy

2. Po získaní inštalačného balíčka spustíme inštaláciu. Otvorí sa nám uvítacie okno inštalátora. Pre pokračovanie v inštalácii stlačíme tlačidlo *Next*.

Obrázok 117 Úvodné okno inštalátora VirtualBoxu

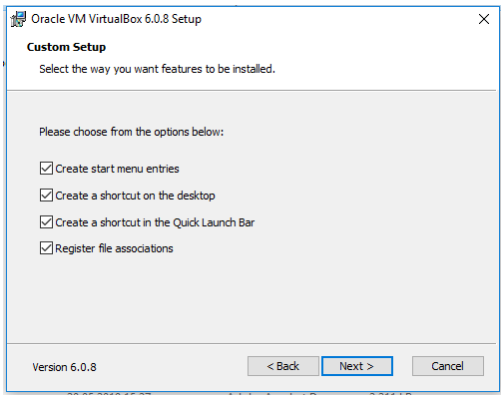
3. Otvorí sa nám okno pre používateľské nastavenie inštalácie VirtualBox. V ľavej časti okna je nám ponúknutá možnosť výberu častí hypervízora, ktoré sa majú nainštalovať. V časti *Location* vidíme cestu k adresáru, kde sa nainštalujú súbory. Ponecháme ponúknuté nastavenie a stlačíme tlačidlo *Next*.

[4] www.virtualbox.org



Obrázok 118 Okno nastavenia inštalácie VirtualBoxu

4. Zobrazí sa okno umožňujúce určiť, ktoré ikony prístupu k VirtualBoxu sa majú nainštalovať. Nastavenie nemeníme a stlačíme tlačidlo *Next*.

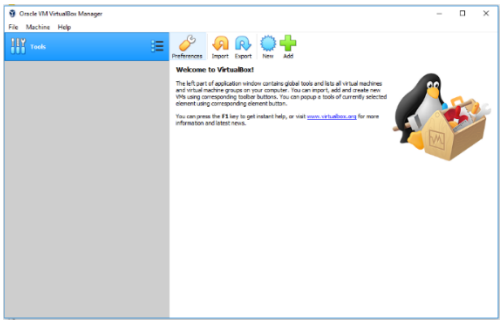


Obrázok 119 Okno nastavenie prístupových ikon

5. Zobrazí sa varovanie o dočasnom reštartovaní sieťového pripojenia. Stlačíme tlačidlo *Yes*. Po potvrdení začína proces inštalácie, ale predtým sa zobrazí ešte jedno okno s uistením, že chceme VirtualBox nainštalovať. Stlačíme tlačidlo *Install*. VirtualBox sa začne inštalovať. Inštalácia bude trvať niekoľko minút.

Obrázok 120 Okno s varovaním

6. Po úspešnej inštalácii sa zobrazí okno s informáciou o úspešnom inštalovaní VirtualBoxu, ktorý môžeme hneď spustiť. zaškrtnutím políčka *Start Oracle VM VirualBox x.x.x after installation* a stlačením tlačidla *Finish*.



Obrázok 121 Uvítacie okno po spustení VirtualBoxu

V Menu pribudla položka *Oracle VM VirtualBox*.

10.3 Typy virtualizácie na báze hypervízora

Virtuálny počítač má prístup k fyzickým zdrojom (hardvéru) hostiteľského počítača. Podľa spôsobu prístupu k hardvéru a softvéru hosťujúceho počítača virtuálnym počítačom rozlišujeme niekoľko základných typov virtualizácie[5]:

1. *úplnú*,
2. *čiastočnú,*
3. *paravirtualizáciu*.

Úplná virtualizácia

Pri *úplnej virtualizácii* je hosťovaný operačný systém je úplne oddelený od fyzického hardvéru. Nemusí byť upravený pre použitie vo virtuálnom prostredí. Pri tomto spôsobe virtualizácie beží hosťovaný operačný systém spolu s jeho aplikáciami len na virtuálnom hardvéri. Je úplne izolovaný od hosťujúceho operačného systému. Hardvérové zariadenia virtuálneho stroja sa simulujú hardvérovými zariadeniami softvérovo vytvorenými hypervízorom. Hypervízor priamo komunikuje s hardvérom hosťujúceho hardvéru. Každá inštancia operačného systému a jej aplikácie bežia v samostatnom virtuálnom stroji. Pri úplnej virtualizácii sú k dispozícii obe platformy hypervízorov, t. j. natívna aj hosťovaná. Treba poznamenať, že operačný systém virtuálneho stroja nie je modifikovaný, ale musí byť určený pre hardvér hosťujúceho počítača. Znamená to, že ak zvolený operačný systém hosťovaného počítača nie je určený pre hardvér hosťujúceho počítača, nedá sa použiť vo virtuálnom stroji. Ak by sme si napríklad vytvoriť virtuálny stroj typu Raspberry Pi (*RPi*) na počítači vybavenom procesorom typu Intel, tak by to pri tomto spôsobe virtualizácie nešlo, lebo počítač *Raspberry Pi* je vybavený procesorom s iným typom architektúry (procesor *ARM*). Príkladom je nainštalovaný hypervízor VirtualBox.

[5] Rôzni autori sa líšia v rozdelení typov virtualizácie.

Čiastočná virtualizácia

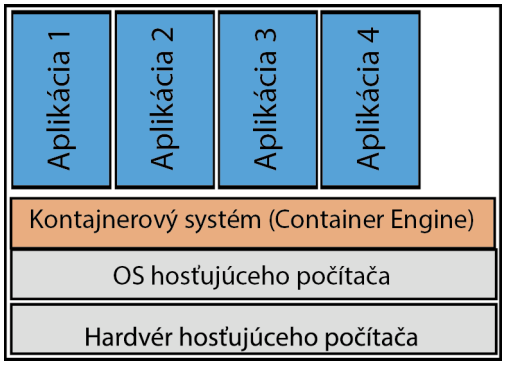
Čiastočná virtualizácia je medzistupňom pred vznikom úplnej virtualizácie. Virtuálny stroj simuluje niektoré hardvérové zariadenia, ale nie všetky. Najčastejšie sa virtualizuje adresný priestor. Potom každý virtuálny stroj pozostáva z nezávislého adresného priestoru. Znamená to, že celý hosťovaný operačný systém nemôže bežať na virtuálnom stroji, ako pri úplnej virtualizácii.

Paravirtualizácia

Pri *paravirtualizácii* je zvyčajne použitý operačný systém hosťovaného počítača modifikovaný. Keďže hosťovaný operačný systém vie, že beží nad hostiteľským hardvérom, niektoré ovládače využívajú virtuálne zariadenia a niektoré bez sprostredkovania hypervízorom priamo pracujú s reálnymi hardvérovými zariadeniami hostiteľského počítača.

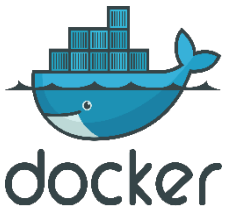
10.4 Kontajnery

Je potrebné spomenúť ešte jednu oblasť, ktorá sa úzko spája s vytváraním virtuálnych strojov – sú to *kontajnery*, ktoré nie sú v pravom slova zmysle virtualizáciou na báze hypervízora. Virtuálne počítače a kontajnery sa líšia niekoľkými spôsobmi, ale primárny rozdiel spočíva v tom, že kontajnery poskytujú spôsob virtualizácie operačného systému tak, že na jednej inštancii operačného systému môže bežať viacero pracovných úloh. Pri *kontajnerovej virtualizácii* sa virtualizuje na úrovni operačného systému. Nevytvára sa kompletný virtuálny počítač vybavený vlastným operačným systémom, ale len *kontajner*, ktorý obsahuje potrebné aplikácie (Obrázok 19). Kontajnery obsahujú iba požadované aplikácie a pre nich špecifické súbory, ale neobsahujú virtualizovaný operačný systém. Každý kontajner využíva jadro hostiteľského operačného systému a zvyčajne aj binárne súbory a knižnice hostiteľského počítača.



Obrázok 122 Schematické zobrazenie kontajnerovej virtualizácie

Príkladom takéhoto virtualizačného nástroja je softvér *Docker* (Obrázok 20). Docker pozostáva z troch častí:



Obrázok 123 Logo kontajnerového systémy Docker

1. *Klienta*, pomocou ktorého používateľ manažuje systém kontajnerov.
2. *Démona (deamon),* ktorý beží neustále na pozadí hostiteľského počítača.
3. *Repozitára*, ktorý obsahuje kontajnery, ktoré je možné prevziať do bežiaceho dockera a po spustení používať. Repozitár je riešený ako cloudová služba.

Koncepcia kontajnerov vychádza z technológie takzvaných *sandboxov*, ktoré sa používajú pri testovaní nebezpečného, škodlivého alebo podozrivého softvéru (malvéru). *Sandbox* využíva hardvér a operačný systém hostiteľského počítača v obmedzenom režime. Poskytuje v ňom bežiacim procesom prístup len k obmedzenej množine zdrojov (vybraným adresárom, portom, serverom, atď.). Sandbox nie je typický príklad virtuálneho stroja. Softvérové riešenia na vytváranie rôzne zameraných sandboxov ponúkajú napr. spoločnosti R&S® alebo *BufferZone*. Ďalej sa môžeme stretnúť s populárnymi softvérmi *Sandboxie*, *SHADE Sandbox*.

CVIČENIE – ANALYZUJTE!

Na experimentovanie s kontajnerovým systémom Docker existuje on-line laboratórium.[6] Prezrite si jeho stránku a video návody, ktoré sa na nej nachádzajú.

Poznámka pre učiteľa:

Ak zvýši čas, je vhodné ukázať študentom kontajnerový prístup na jednoduchom príklade.

10.5 Emulácia hardvéru

Hardvérová emulácia (niekedy nazývaná hardvérový preklad) je typ hosťovanej virtualizácie. Od úplnej virtualizácie sa líši v tom, že v emulácii hypervízor poskytuje iné hardvérové rozhrania ako sú rozhrania poskytované fyzickým hardvérom. Každá jednotlivá inštrukcia virtuálneho stroja je prekladaná (emulovaná) na inštrukciu hostiteľského stroja.

[6] https://training.play-with-docker.com/

Emulátory teoreticky umožňujú spustiť kód napísaný pre konkrétny druh hardvéru na stroji s úplne inou architektúrou a preto môže nad hypervízorom bežať nezmenený hosťovaný operačný systém určený pre platformy odlišné od platformy hostiteľa. Nutnosť prekladu spôsobuje značné spomalenie činnosti hosťovaného stroja.

Pri tomto spôsobe je možné na hostiteľskom systéme simulovať veľké množstvo rôznych architektúr, ktoré sú úplne odlišné od hostiteľskej. Výhodou je ľahká inštalácia a správa, nevýhodou je nízky výkon a vysoká réžia. Príkladom takéhoto emulátora je *Qemu (Quick EMUlator)*.

10.6 Virtuálny počítač

Virtualizácia serverov a virtualizácia osobných počítačov spočíva vo vytváraní *virtuálnych počítačov* (virtuálnych strojov) na konkrétnom fyzickom stroji (počítači). V prostredí hypervízora môže byť vytvorených viacero virtuálnych počítačov, pričom počet bežiacich virtuálnych počítačov je limitovaný hardvérovými parametrami hostiteľského počítača (možnosti procesora, veľkosť operačnej pamäte, kapacita pevného disku, atď.). Ako sme si uviedli, na vytvorenie virtuálneho počítača môžeme použiť rôzne hypervízory, či už platené alebo zo skupiny slobodného softvéru.

ZAPAMÄTAJ SI!

Virtuálny počítač je softvérová implementácia fyzického počítača bežiaceho v manažéri virtuálnych strojov (hypervízore).

Virtuálny počítač sa z pohľadu systémového prostredia javí ako fyzicky jestvujúci, aj keď v skutočnosti je jeho hardvér len abstraktný. Ak virtuálny počítač pracuje správne, tak by používateľ nemal pociťovať žiaden rozdiel medzi používaním fyzického počítača a virtuálneho počítača (vlastnosť rovnocennosti). Môžeme sa naň pozerať ako na *abstraktný hardvér*.

Z pohľadu operačného systému na fyzickom počítači beží len jeden operačný systém. To isté platí aj o virtuálnom počítači. Keďže na fyzickom počítači môže byť vytvorených viacero navzájom *nezávislých virtuálnych počítačov*, každý s vlastným operačným systémom, potom nad fyzickým počítačom môže bežať viacero rôznych operačných systémov.

Súbory, ktoré reprezentujú virtuálny počítač, obsahujú opis hardvéru virtuálneho počítača, operačný systém, dátové súbory a nainštalované aplikácie. Ak virtuálny počítač nie je spustený (nebeží), tak je v hostiteľskom počítači uložený vo forme súboru, takzvaného obrazu virtuálneho počítača. Tento obraz sa dá teoreticky prenášať medzi hypervízormi nainštalovanými na rôznych fyzických počítačoch a vytvárať tak klony (kópie) virtuálneho počítača.

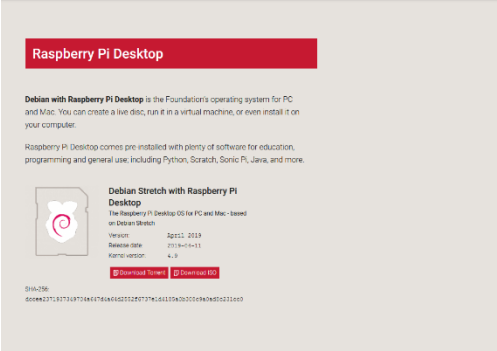
MOTIVÁCIA

Máme fyzický počítač vybavený procesorom triedy *AMD64* (*x86-64*) a chceme si vytvoriť virtuálny počítač, na ktorom si budeme skúšať prácu s mikropočítačom triedy *Raspberry Pi*. Ako operačný systém použijeme na to určený *Raspbian* (*Linux Debian* pre mikropočítače RPi). Mikropočítač RPi je však vybavený procesorom triedy *ARM*. Aký spôsob virtualizácie využijeme?

VÝKLAD

Zameriame sa na tri spôsoby. Využijeme už nainštalovaný hypervízor VirtualBox, potom Hyper-V, ktoré ponúkajú úplnú virtualizáciu a ukážeme si použitie emulácie pomocou Qemu.

V prvých dvoch prípadoch je problém s nekompatibilitou hardvéru. Napriek tomu existujú



Obrázok 124 Stránka so súborom Linux Debian s Desktopom Raspberry Pi

inštalácie operačného systému Linux Debian, ktoré dokážu napodobniť vizuálne prostredie počítača RPi na postačujúcej úrovni. Budeme inštalovať virtuálny stroj s operačným systémom Linux na hostiteľskom počítači s operačným systémom Windows. Obraz (inštalačný súbor) OS Linux s napodobením prostredia počítača RPi pre desktop získame z webovej stránky projektu Raspberry Pi.[7]

Poznámka pre učiteľa:
Nie je nutné inštalovať práve tento typ operačného systému. Ak zvolíte iný, upravte návod pre študentov. Priebehu inštalácie OS Linux Debian nevenujeme. Ak by mali študenti s inštaláciou problém, treba im pomôcť.

Poznámka pre učiteľa:

Úlohy voľte podľa toho, ktorý hypervízor ste si nainštalovali.

[7] https://www.raspberrypi.org/downloads/raspberry-pi-desktop/

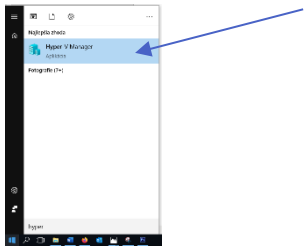
Virtuálny stroj v prostredí hypervízora Hyper-V

Poznámka pre učiteľa:

Skontrolujte, či majú študenti aktivovaný Hyper-V, alebo to pripomeňte.

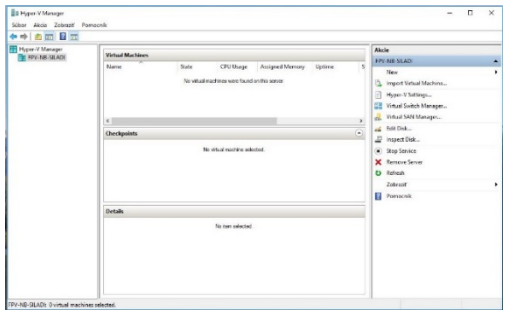
Vytvorenie virtuálneho stroja

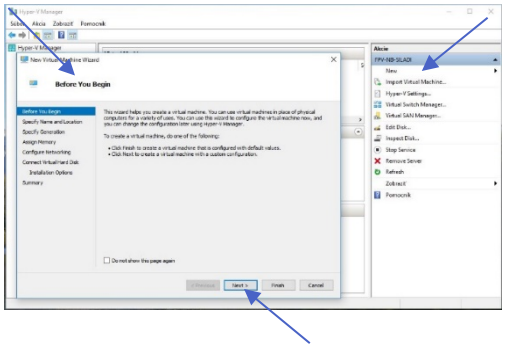
1. Otvoríme si okno hypervízora tak, že klikneme ľavým tlačidlom myšky na ikonku *Štart* a začneme písať *Hyper-V*. Okno hypervízora otvoríme kliknutím na odkaz *Hyper-V Manager*, ktorý sa nám objaví v ponuke. Otvorí sa nám okno *Hyper-V manažéra*.



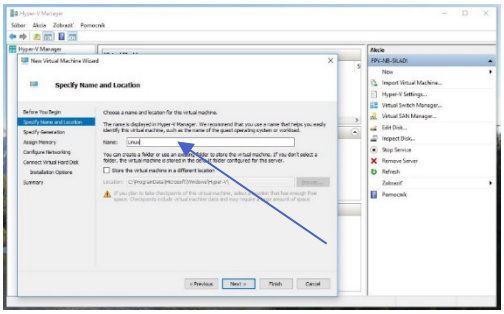
Obrázok 125 Odkaz v Menu na Hyper-V Manager

2. V časti *Akcie* klikneme na odkaz *New* a vyberieme možnosť *Virtual Machine*. Otvorí sa sprievodca, ktorý nás bude viesť tvorbou virtuálneho stroja. Na posúvanie sa na nasledujúce kroky vytvárania budeme používať tlačidlo *Next*.

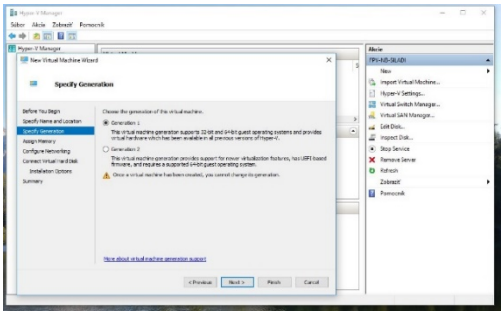




3. Otvorí sa okno na špecifikáciu názvu virtuálneho stroja a jeho umiestnenie. Zadáme názov, napr. Linux a pomocou tlačidla *Next* prejdeme k ďalšiemu kroku vytvárania virtuálneho stroja.



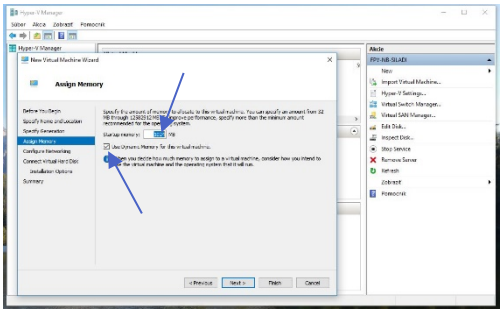
4. Zvolíme virtuálny stroj *1. generácie*.



Poznámka pre učiteľa:

Pre tento krok navrhujeme použiť princíp „čiernej skrinky“ a nevysvetľovať „prečo“.

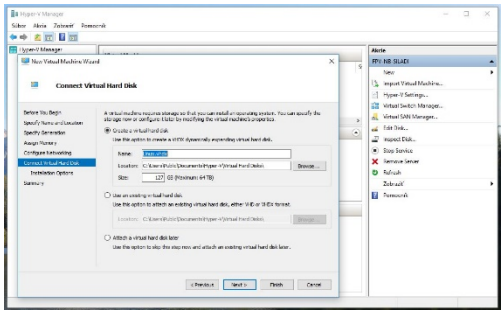
5. V nasledujúcom kroku nastavíme *veľkosť počiatočnej pamäte* pre virtuálny stroj na ponúkaných *1024 MB* a povoliť dynamické využívanie veľkosti pamäte pre virtuálny počítač.
6. Pripojenie na sieť v tomto prípade vynecháme.



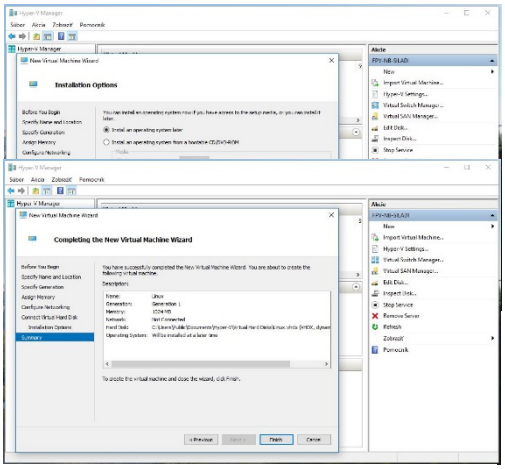
Poznámka pre učiteľa:

Pre tento krok navrhujeme použiť princíp „čiernej skrinky“ a nevysvetľovať „prečo“.

7. Vytvoríme *virtuálny disk* pre počítač a umiestnime ho na dostatočne veľké a rýchle médium v PC. Názov a veľkosť necháme tak, ako sú predvolené.



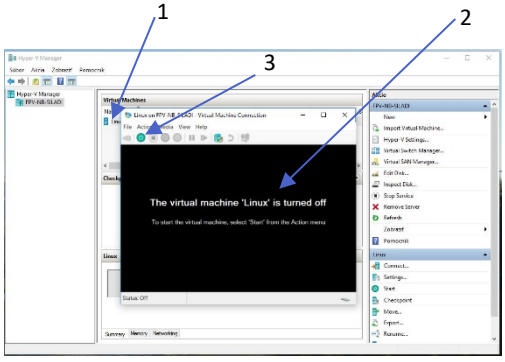
8. Zvolíme, či chceme, alebo *nechceme* inštalovať OS okamžite. Zvolíme „nie“, lebo budeme využívať *live verziu OS Linux Debian*.



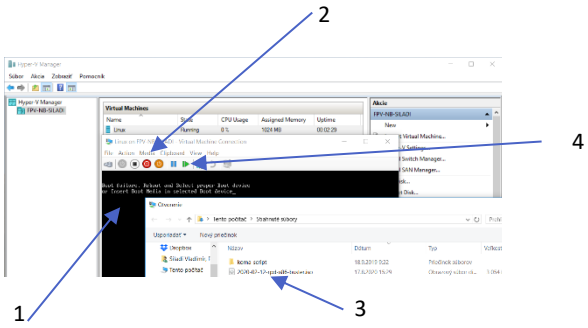
9. Prezrieme si súhrnné informácie o vytvorenom virtuálnom stroji a potvrdíme jeho vytvorenie kliknutím na tlačidlo *Finish*. Následne sa virtuálny počítač vytvorí a zobrazí sa v zozname virtuálnych strojov, ktoré máme k dispozícii.

Spustenie virtuálneho stroja

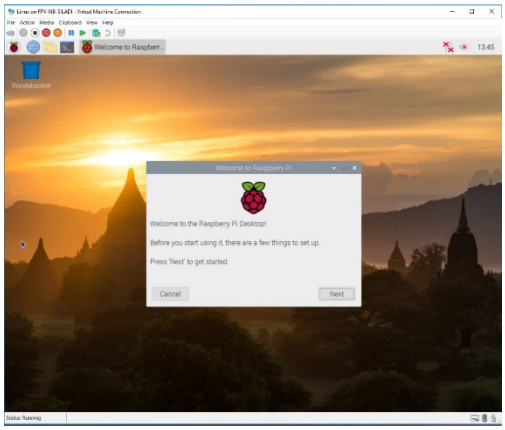
10. Spustíme virtuálny stroj kliknutím na odkaz naň (1) v *zozname virtuálnych strojov (Virtual Machines)*. Zobrazí sa nám informácia, že je vypnutý (2). Následne klikneme na ikonku Start (3) .



11. Zobrazí sa chybová obrazovka (1). K virtuálnemu stroju musíme pripnúť obraz požadovaného OS cez menu *Media/DVDdrive/Insert Disk* (2, 3), v našom prípade Linux Debian s obrazovkou Rspbianu, ktorý sme získali zo stránky uvedenej na začiatku tejto časti. Klikneme na ikonu *Reset* (4).



12. Objaví sa úvodná obrazovka inštalácie OS Linux Debian. Nemusíme nič vyberať, len počkáme kým sa rozbehne live verzia OS. Samozrejme, môžeme sa rozhodnúť aj pre inštaláciu OS na virtuálny stroj (bude uvedené ďalej).



Obrázok 126 Okno virtuálneho stroja s bežiacim OS

CVIČENIE – POUŽITE!

Ak máte k dispozícii počítač s OS Windows 10 a aktivovaným Hyper-V zopakujte opísaný postup.

ÚLOHA – RIEŠTE!

Zastavte bežiaci virtuálny počítač a znovu ho spustite.

Nájdite adresár so súborom virtuálneho pevného disku vytvoreného virtuálneho počítača.

Odstráňte exitujúci virtuálny počítač.

Nainštalujte ďalší virtuálny počítač s iným operačným systémom ako Linux Debian.

Virtuálny stroj v prostredí hypervízora VirtualBox

Poznámka pre učiteľa:

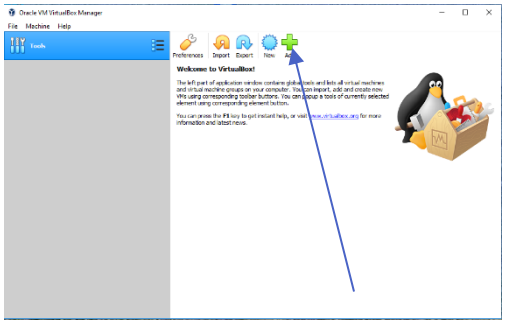
Ak máte k dispozícii počítače s nainštalovaným OS Linux, upravte zadanie pre konkrétnu verziu OS Linux.

Úloha je trochu problematická (najmä pri práci v OS Windows 10), lebo je potrebné niekedy deaktivovať niektoré bezpečnostné prvky v BIOS-e (napr. Configure Legacy Support and Secure Boot[8]) a Hyper-V vo Windows. Ak sa objaví pri pripájaní obrazu hlásenie *Invalid settings detection*, je potrebné vypnúť deaktivovať *Hyper-V*, ako sme si uviedli v texte vyššie.

Pomocou nainštalovaného hypervízora *VirtualBox* vytvorme v počítači virtuálny počítač s *OS Linux Debian*, ktorý sa vizuálne bude podobať na prostredie *OS Raspbian*. Túto úlohu možno riešiť na OS Linux aj OS Windows.

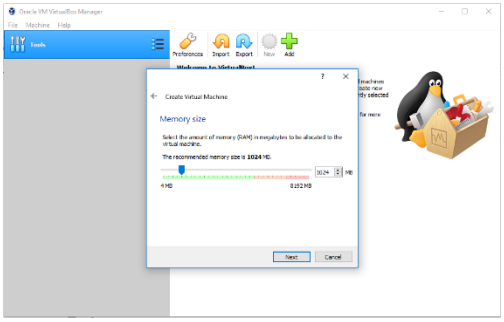
Vytvorenie virtuálneho stroja

1. Klikneme na ikonu hypervízora VirtualBox (v *Menu* alebo *Paneli úloh* alebo na obrazovke). Privíta nás okno grafického rozhrania manažéra virtuálnych strojov. Kliknutím ma na ikonu *Add* spustíme proces vytvárania nového virtuálneho stroja.



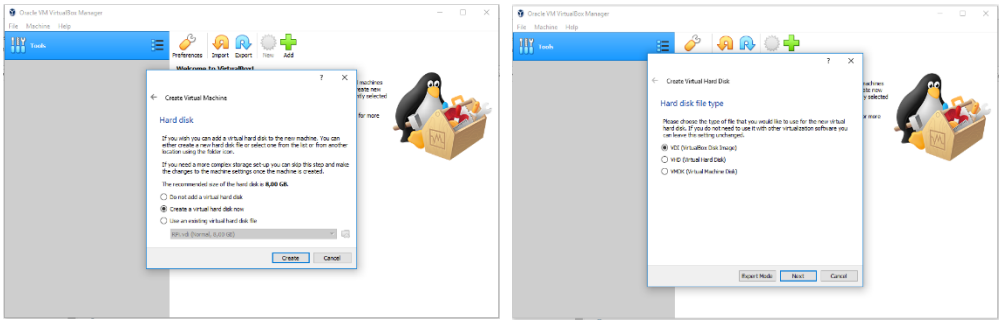
[8] Pri tvorení učebného textu to nebolo ptrebné.

2. Otvorí sa okno na zadanie názvu inštancie vytváraného virtuálneho stroja, zvolenie typu operačného systému virtuálneho stroja a jeho verzie. My zvolíme Linux Debian (32-bit). Názov zvoľte podľa ľubovôle. Stlačíme tlačidlo *Next*.
3. Ďalšie okno umožňuje zvoliť veľkosť RAM virtuálneho. Treba si dať pozor, aby RAM virtuálneho počítača bola dostatočne veľká pre hosťovaný operačný systém a softvér, ktorý naň chceme nahrať a používať (treba pozrieť dokumentáciu k vybranému OS). Netreba zabudnúť, že tým odkrajujeme z kapacity RAM. Ponecháme prednastavenú veľkosť 1024 MB.



Obrázok 127 Voľba veľkosti RAM a pevného disku virtuálneho počítača

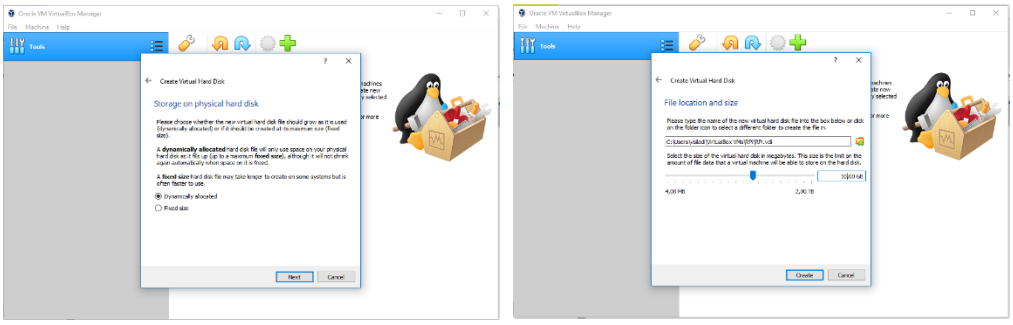
4. V nasledujúcom okne zvolíme tvorbu nového virtuálneho pevného disku (druhá voľba – *Create a virtual hard disk now*). A stlačíme tlačidlo *Create*. Tretia možnosť (*Use an existing virtual hard disk file*) slúži na použitie exitujúceho virtuálneho pevného disku, ktorý reprezentuje nainštalovaný virtuálny počítač. To sa používa napríklad pri klonovaní virtuálnych strojov.



Obrázok 128 Voľba nového virtuálneho pevného disku a voľba typu súboru reprezentujúceho virtuálny pevný disk

5. Ďalšie okno ponúka aký typ súboru sa bude používať ako virtuálny pevný disk. Ponecháme prednastavenú hodnotu (VDI).
6. Následne si zvolíme, či bude mať náš virtuálny počítač dynamicky sa meniacu veľkosť podľa potreby alebo pevnú veľkosť. V prvom prípade je proces vytvárania disku rýchlejší, ale pri používaní potom pracuje pomalšie. Alokuje sa preň len nevyhnutne potrebná veľkosť a počas používania virtuálneho počítača sa jeho veľkosť mení podľa potreby (je to ohraničené zadanou maximálnou veľkosťou). Pri použití druhej možnosti, trvá vytváranie disku dlhšie, ale potom pracuje disk rýchlejšie. Alokuje sa preň hneď jeho maximálna požadovaná veľkosť. My zvolíme dynamické alokovanie veľkosti disku a stlačíme tlačidlo *Next*.
7. Otvorí sa okno, v ktorom nastavíme veľkosť pevného disku virtuálneho počítača. Podobne ako pri určení veľkosti RAM je potrebné správne nastaviť jeho veľkosť podľa

dokumentácie a zamýšlanému použitiu virtuálneho počítača. Osvedčila sa veľkosť minimálne 10 GB. Na začiatku je veľkosť súboru pri dynamickom alokovaní priestoru 2 GB. Po nastavení veľkosti disku stlačíme tlačidlo *Create* a virtuálny počítač je pripravený k prvému spusteniu.

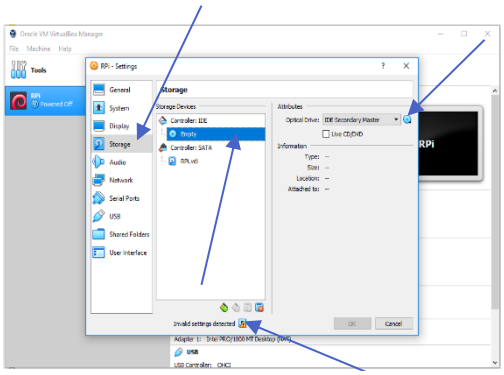


Obrázok 129 Nastavenie maximálnej veľkosti pevného disku

Súbor s virtuálnym pevným diskom sa nachádza v podadresári *VirtualBox VMs*, ktorý je štandardne umiestnený v domovskom adresári používateľa.

Spustenie virtuálneho stroja

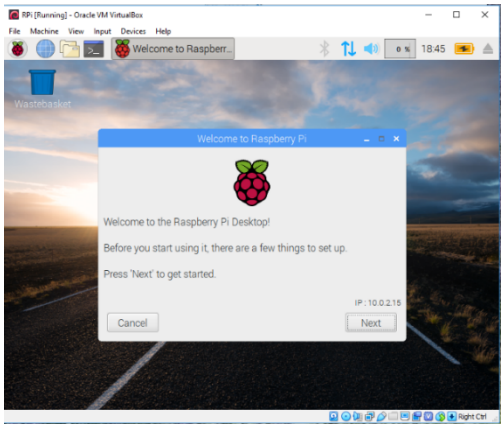
8. Teraz potrebujeme k virtuálnemu stroju pripojiť obraz inštalačného disku vybraného operačného systému. My si zvolíme, tak ako v predchádzajúcom príklade, OS *Linux Debian* (ako sme na začiatku nastavili vo vlastnostiach virtuálneho stroja) s *Desktopom Raspberry Pi*.
9. Klikneme na ikonu *Settings* (Obrázok 42). Otvorí sa ďalšie okno (Obrázok 43), v ktorom si zvolíme záložku Storage. Pripojíme súbor s inštalačným obrazom k virtuálnemu počítaču (kliknutím na ikonu disku – *Optical Device*). Ak nie je súbor pripojený, svieti označenie *Empty*. (Ak je v dolnej časti okna zobrazená informácia *Invalid settings detected* a nedá sa kliknúť na tlačidlo *OK*, požiadajte o pomoc vyučujúceho. Treba urobiť zásahy do nastavenia hostiteľského operačného systému.)



Obrázok 130 Pripojenie obrazu inštalačného disku hosťovaného operačného systému

Ak prebehlo pripojenie obrazu inštalačného disku úspešne, tak sa hláška *Empty* zmení na názov pripojeného súboru.

10. Môžeme prvýkrát spustiť virtuálny počítač tlačidlom *Start*. Začne prebiehať klasická inštalácia operačného systému. O úspešnej inštalácii nás presvedčí okno s obrazovkou virtuálneho počítača.



Obrázok 131 Úvodné obrazovka operačného systému Linux Debian Raspberry Pi

Poznámka pre učiteľa:

Ďalšie nastavenia (ako napríklad sieť) v texte nerozoberáme, lebo po tejto inštalácii bude prepojenie NAT fungovať. Nastavovania môžete využiť ako cvičenie k iným kapitolám o počítačových systémoch, sieťach a operačných systémoch.

CVIČENIE – POUŽITE!

Zopakujte na počítači uvedený postup.

ÚLOHA – RIEŠTE!

Zastavte bežiaci virtuálny počítač a znovu ho spustite.

Nájdite adresár so súborom virtuálneho pevného disku vytvoreného virtuálneho počítača.

Odstráňte exitujúci virtuálny počítač.

Skúste nainštalovať ďalší virtuálny počítač s iným operačným systémom ako Linux Debian.

Emulácia počítača RPi pomocou emulátora Quemu

Poznámka pre učiteľa:

Ak máte k dispozícii počítače s nainštalovaným OS Linux, upravte postup pre konkrétnu verziu OS Linux.

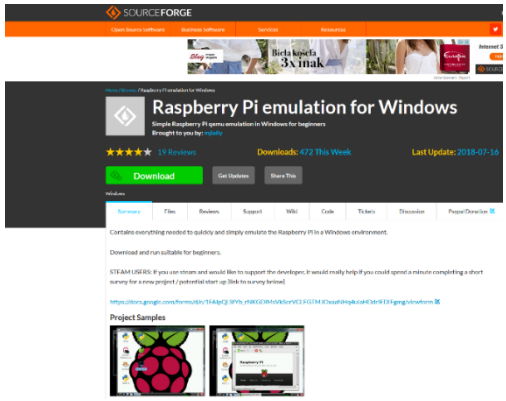
Teraz si vyskúšame vytvoriť virtuálny stroj pomocou emulácie. Teda všetky inštrukcie pre procesor ARM budú prekladané do inštrukcií pre procesor Intel. Použijeme na to nástroj

Qemu. Pomocou hypervízora Qemu vytvorme virtuálny počítač, ktorý bude emulovať Raspberry Pi. Využijeme už hotové riešenie, ktoré je k dispozícii pre OS Windows na webe.[9]

Poznámka pre učiteľa:

V texte už nebudeme zachádzať do detailov procesu inštalácie. Upozorníme len na podstatné časti. Pomoc poskytujte podľa schopností a zručností študentov. Inštalácia nie je zložitá, lebo podstatné časti sú už predpripravené na webe. Šikovní študenti sa môžu pokúsiť podľa návodov na webe o vlastnú inštaláciu.

1. Nainštalujeme si do počítača hotové riešenie z webovej stránky.[10] Toto riešenie obsahuje v sebe tri zložky, ktoré by sme v prípade, ak by sme sa pokúsili o vlastnú samostatnú inštaláciu, získať a nainštalovať:
   - Qemu[11],
   - OS Raspbian,
   - Linuxový kernel Qemu kompilovaný pre procesor ARM[12].
   - V takomto prípade je potrebné dať pozor na to, aby sedela skompilovaná verzia kernelu s verziou OS Raspbian (napr. *Wheezy*, Jessie, Stretch alebo Buster).
2. Stiahneme inštalačný súbor zo spomínanej webovej stránky kliknutím na tlačidlo *Download* a rozbalíme ho napr. na Ploche (alebo v domovskom adresári). Sťahovanie sa začne automaticky po niekoľkých sekundách.

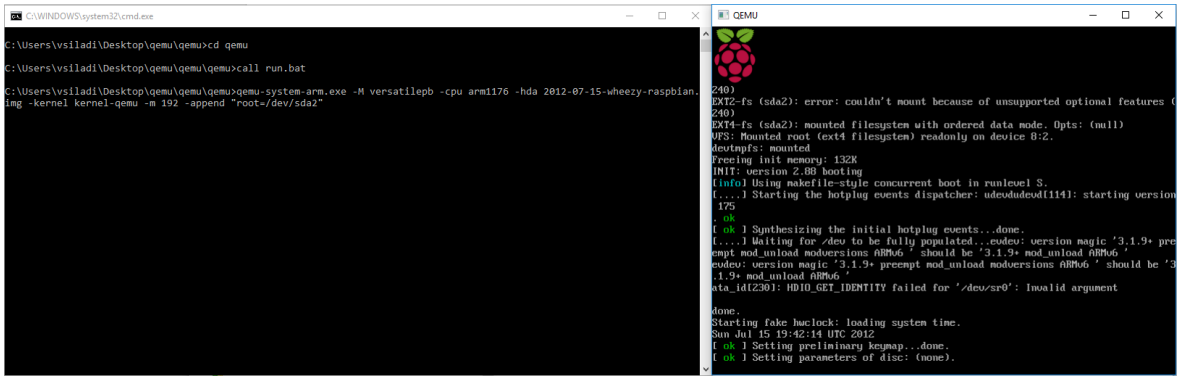


Obrázok 132 Webová stránka s upravenou inštaláciou Qemu Raspbianom

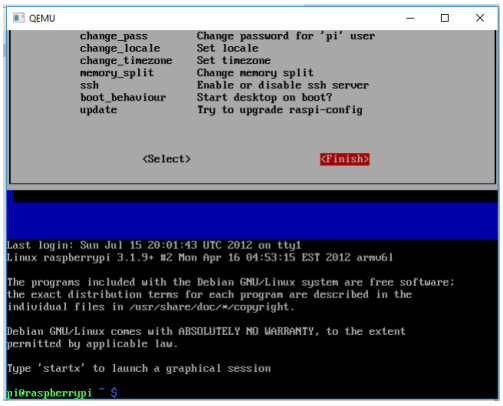
3. Spustíme emuláciu dávkovým súborom `run.bat`
4. Všetky ďalšie informácie k emulátoru sa nachádzajú v súbore `README.txt`. Pozor, pri prvom spustení sa ešte objaví inštalačné okno z Linuxu. Stačí len pomocou tabulátora

[9] https://sourceforge.net/projects/rpiqemuwindows/
[10] https://sourceforge.net/projects/rpiqemuwindows/files/latest/download
[11] https://www.qemu.org/
[12] napr. tu https://github.com/dhruvvyas90/qemu-rpi-kernel

kliknúť na *Finish*. Pri ďalšom spúšťaní emulátora sa toto okno už nezobrazí a hneď nás privíta prostredie Raspbianu.



Obrázok 133 Spúšťanie emulácie Raspbianu v prostredí Qemu

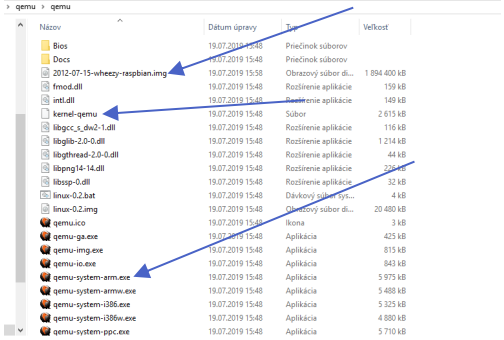


Obrázok 134 Prvé spustenie emulácie Raspbianu v Qemu

CVIČENIE – ANALYZUJTE!

V adresári Qemu si všimnite vyššie spomínané súbory: emulátora konkrétnej architektúry (napr. `qemu-system-arm.exe`), skompilovaného kernelu (`kernel-qemu`) a obrazu inštalácie Raspbianu (`2012-07-15-wheezy-raspbian.img`).

Obrázok 135 Súbory v domovskom adresári Qemu



CVIČENIE – POUŽITE!

Zopakujte na počítači uvedený postup.

ZHRNUTIE

Čo sme sa naučili

- Základné pojmy z oblasti virtualizácie hardvéru a operačných systémov.
- Typy hypervízorov.
- Rozdiel medzi virtualizáciou na báze hypervízora, emuláciou a kontajnermi.
- Aktivovať hypervízor Hyper-V.
- Nainštalovať hypervízor VirtualBox.
- Vytvoriť a pracovať s virtuálnym počítačov v prostredí Hyper-V.
- Vytvoriť a pracovať s virtuálnym počítačov v prostredí VirtualBoxu.
- Pracovať s emuláciou Raspbinau v Qemu.

BIBLIOGRAFIA

[1] LOHR, Steve. Is Mr. Gates Pouring Fuel On His Rivals' Fire? New York . Times, ECONOMIC VIEW. s. 3003004. (18. 04 1999).

[2] RUEST, Danielle – RUEST, Nelson. Virtualizace : podrobný průvodce . [překlad Pavel Vaida]. 1. vyd. Brno : Computer Press, 2010. 408 s. ISBN 978-80-251-2676-9.

[3] POPEK, Gerald J. – GOLDBERG, Robert P. Formal requirements for virtualizable third generation architectures . Commun. ACM 17, 7, 1974. s. 412-421

[4] BHOWMIK, Sandeep. Cloud computing. 1st ed. Cambridge : Cambridge University Press, 2017. xxiii, 407 s. ISBN 978-1-316-63810-1.

11 ARCHITEKTÚRA POČÍTAČOVEJ SIETE A INTERNETU

CIEĽ

Cieľom tejto kapitoly je spoznať základné sieťové prvky počítačovej siete. Pasívne prvky tvoria sieťovú kabeláž, aktívne prvky sú sieťové zariadenia. Žiak bude vedieť popísať vlastnosti jednotlivých prvkov a bude sa vedieť orientovať v ich špecifikácii. Naučí sa tiež základné ovládanie programu Cisco Packet Tracer.

MOTIVÁCIA

Určite ste sa už stretli s tým, že si chcete pripojiť počítač (notebook) k domácej sieti, rozšíriť domácu WiFi sieť pretože chcete lepšie pokrytie signálom alebo proste len opraviť niečo v sieti čo sa pokazilo. Aby sme vybudovali alebo opravili počítačovú sieť potrebujeme poznať z čoho sa počítačová sieť skladá, čo máme k dispozícii na jej budovanie. Jednotlivé komponenty môžeme rozdeliť do dvoch kategórií – káble a sieťové zariadenia.

VÝKLAD

11.1 Pasívne sieťové prvky - káble v počítačovej sieti

Nebudeme sa zaoberať kabelážou, ktorá sa používala v počítačových sieťach v minulosti. Sieťovú kabeláž môžeme rozdeliť do dvoch skupín – metalickú a optickú. Základom metalickej kabeláže je „metal“ čiže kov, v našom prípade meď, teda medený kábel. Sieťový kábel (voláme ho aj ethernet kábel) sa skladá zo štyroch párov vodičov („káblikov“), ktoré sú v ňom riadne medzi sebou poprepletané ako lano.

Prípad zo života – predajňa elektro

Zákazník: Dobrý deň, prosím vás, máte internetový kábel?

Predajca: Myslíte sieťový - ethernetový kábel? Tak ten máme.

Zákazník: Prosím si 20 metrov.

Predajca: Želáte si lanko alebo drôt? Chcete tienený alebo netienený? Akej kategórie ten kábel chcete? Na vnútorné alebo vonkajšie použitie?

Zákazník: Prosím si UTPčko, 5e, lanko. A k tomu 10 ks netienených RJ45.

Predajca: Hneď to bude.

Ethernetové káble sa vyrábajú v prevedení „lanko“ alebo „drôt“. Prevedenie drôt znamená, že v izolácii sa nachádza len jeden medený vodič. Keďže v celom kábli je týchto vodičov osem, takýto kábel je pevný, ťažšie ohybný a viac náchylný na zlomenie. Druhou alternatívou je, že vodič je zložený z veľa veľmi tenkých medených vlasov. Takýto kábel je ohybný a nie je náchylný na prelomenie.

ZAPAMÄTAJTE SI!

Lanko používame na prepojenie počítača do sieťovej zásuvky. Drôt sa umiestňuje do stien alebo žľabov a ukončuje sa zásuvkou.

Káble odborne nazývame **prenosové média** delíme ich na **medené a optické**. Špeciálny typom prenosu je bezdrôtové spojenie, pri ktorom prenosové médium nedefinujeme.

Poznámka pre učiteľa:

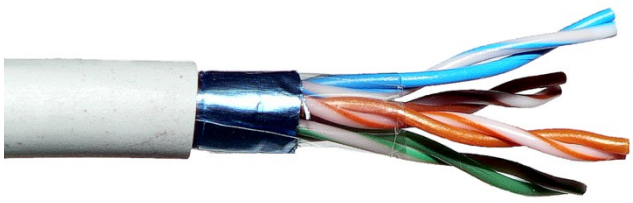
V bezdrôtovej dobe je vhodné zohnať reálne ukážky kabeláže. Predpokladáme, že akákoľvek predajňa výpočtovej techniky alebo elektro spotrebného materiálu s touto požiadavkou nebude mať problém. Na kábloch je vytlačené ich typové označenie s množstvom ďalších technických informácií.

11.1.1 Medené prenosové médiá

V dnešnej dobe ja najpoužívanejším prenosovým médiom TP kábel - (Twisted Pair) – krútená dvojlinka. Je tvorený ôsmimi vodičmi, ktoré sú rozdelené do štyroch párov a každý vodič má vlastné farebné označenie. Vodiče v každom páre sú krútené okolo seba a páry sú krútene navzájom. Výhoda zakrútenia je, že znižuje vzájomné rušenie prenášaných signálov, čím viac je zakrútení, tým je rušenie menšie. Na pripájanie sa používa konektor RJ45 (8P8C) a maximálna dĺžka TP kábla je 100 m. Používa sa na tvorenie sietí typu ethernet 10BASE-T, Fast Ethernet 100BASE-T a 1000BASE-T. Označenie možno vyzerá komplikovane, no nie je. Číslo na začiatku predstavuje prenosovú rýchlosť (10 Mbit/s, 100 Mbit/s, 1000 Mbit/s), slovo Base znamená, že sa jedná o fyzické médium a T je označenie pre „Twisted Pair“.

CVIČENIE – DISKUTUJTE!

Prečo sú jednotlivé páry zakrútené? Ako to môže brániť rušeniu? Spomeňte si na definíciu ampéra ako jednotky SI a analyzujte ju.



Obrázok 136 TP kábel
(https://sk.wikipedia.org/wiki/Kr%C3%BAten%C3%A1_dvojlinka#/media/File:FTP_cable.jpg)

Podľa vyhotovenia delíme TP káble na:

- Unshielded Twisted Pair (UTP), netienený kábel
- Screened Unshielded Twisted Pair (S/UTP), Foiled Twisted Pair (FTP)
- Shielded Twisted Pair (STP), Screened Shielded Twisted Pair (S/STP)

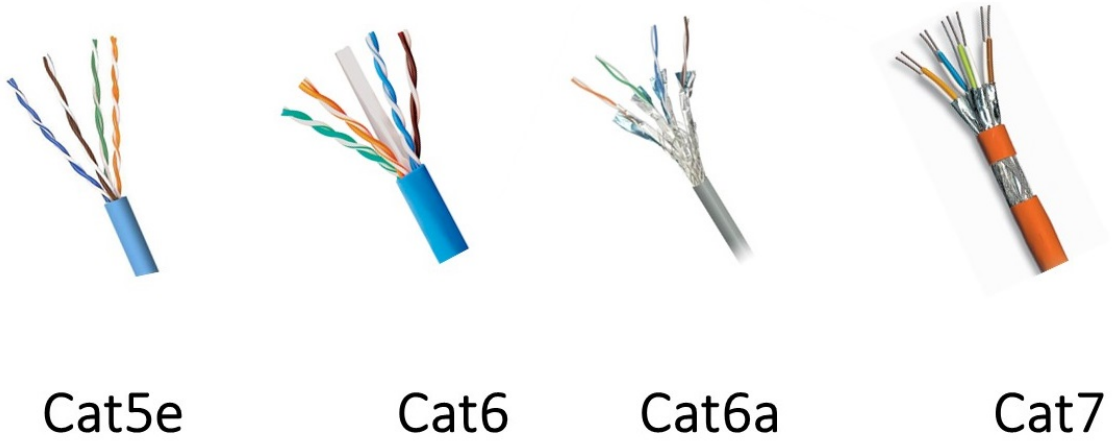
Unshielded Twisted Pair (UTP), netienený kábel je tvorený len 4 pármi a vonkajšou izoláciou. Jeho výhodami sú jednoduchá inštalácia, ohybnosť, nízka cena. Nevýhodou je, že pokiaľ máme vedenie tvorené UTP káblami v blízkosti elektrických rozvodov, môže nastáva rušenie signálu. Používame ho najmä na pripájanie koncových zariadení (Patch cable) na kratšie vzdialenosti, aj keď jeho dosah je 100 m.

Screened Unshielded Twisted Pair (S/UTP) , tiež známy ako **Foiled Twisted Pair (FTP)** je základnou tienenou verziou UTP. Spoločné tienenie je vo forme kovovej fólie a musí byť uzemnené, inak sa správa ako anténa a spôsobuje ďalšie rušenie. Tento typ tienenia chráni je pred vonkajšími zdrojmi rušenia.

Shielded Twisted Pair (STP) a **Screened Shielded Twisted Pair (S/STP)** má každý pár tienený kovovou fóliou. Tienením každého páru sa redukuje prípadné rušenie párov navzájom. Tento typ káblu je najdrahší a používa sa v ťažkom priemysle (zdroje rušenia môžu byť napríklad elektromotory, turbíny a pod.)

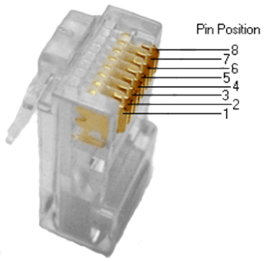
Obrázok 137 Vyhotovenia TP káblov – UTP, FTP, STP (https://www.universalnetworks.co.uk/faq/copper/what-does-utp-ftp-stp-or-sftp-mean)

Okrem verzie tienenia pri TP kábli určujeme ešte jeho kategóriu. Čím je kategória vyššia, tým rýchlejšie vieme prenášať údaje. Dnes používané sú Cat5E, Cat6 a Cat7. Káble kategórie 5E prenášajú údaje maximálnou rýchlosťou 1 GB/s. Káble kategórie 6 prenášajú údaje rýchlosťou až 10 GB/s (aj keď s obmedzením dĺžky na 50 metrov) a káble kategórie 7 dosahujú teoreticky aj vyššie rýchlosti. Samozrejme platí, čím vyššia kategória, tým vyššia cena. V domácnosti aj malej firme nám postačuje aj kategória 5E.



Obrázok 138 Kategórie TP káblov (http://www.fiberopticshare.com/guide-choosing-suitable-ethernet-cables.html)

Pri nákupe káblov sa tiež stretneme s označením priamy (straight) alebo krížený (crossover) kábel. Závisí to od spôsobu akým sú na koncoch kábla zapojené koncovky, ktoré majú označenie RJ45. Na zhotovenie káblu existujú dva štandardy zapojenia: TIA/EIA-568-A a TIA/EIA-568-B.

Pin	T568A pár	T568B pár	T568A farba	T568B farba	Piny na konektore RJ45 (8P8C)
1	3	2	bielo-zelená	bielo-oranž.	
2	3	2	zelená	oranžová	
3	2	3	bielo-oranž.	bielo-zelená	
4	1	1	modrá	modrá	
5	1	1	bielo-modrá	bielo-modrá	
6	2	3	oranžová	zelená	
7	4	4	bielo-hnedá	bielo-hnedá	
8	4	4	hnedá	hnedá	

Obrázok 139 Poradie vodičov v TP kábli

Priamy kábel (straight cable) je najčastejšie používanou verziou. Používame ho na pripojenie počítača k prepínaču alebo domácemu Wi-Fi smerovaču. Koncovka je na oboch koncoch zapojená rovnako (buď A, alebo B).

Krížený kábel (crossover cable), používame ak spájame dve rovnaké sieťové zariadenia na priamo, napríklad ak chceme prepojiť priamo dva počítače. Kábel má na jednom konci koncovku zapojenú podľa A na druhom podľa B. Dnešné sieťové zariadenia sú inteligentné, vedia si krížený kábel „vyrobiť“ aj keď použijeme priamy kábel.

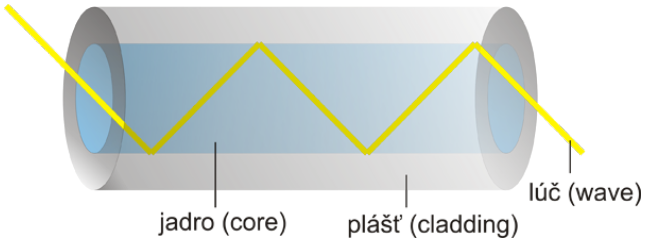
11.1.2 Optické prenosové médiá

Optické káble sa používajú najmä v počítačových sieťach poskytovateľov služieb internetu aj v internete ako takom. Ich hlavnou výhodou je vysoká prenosová rýchlosť (10 GB/s a viac) a to

že nie sú ovplyvnené rušením a netreba ich tieniť. Dnes nie je neobvyklé, keď vám poskytovateľ privedie optiku až domov.

Obrázok 140 Optické káble (http://www.chinacablesbuy.com/wp-content/uploads/2017/04/cropped-2-1.jpg)

Optické vlákna sa skladajú z jadra okolo ktorého je plášť. Jadro aj plášť sú vyrobené z takých materiálov aby sa svetelný lúč dobre odrážal a postupoval tak vláknom (a hlavne neprechádzal z jadra do plášťa). Určite si spomeniete na zákon odrazu a zákon lomu z fyziky. Lúč sa dostáva na koniec vlákna postupnými odrazmi. Jedno vlákno môže prenášať údaje len jedným smerom, teda na komunikáciu medzi dvoma bodmi potrebujeme dve vlákna. V optických kábloch nájdeme vedľa seba naukladaných niekoľko desiatok optických vlákien.



Obrázok 141 Princíp optického vlákna

Lúč sa dostáva na koniec vlákna postupnými odrazmi. Jedno vlákno môže prenášať údaje len jedným smerom, teda na komunikáciu medzi dvoma bodmi potrebujeme dve vlákna. V optických kábloch nájdeme vedľa seba naukladaných niekoľko desiatok optických vlákien.

Podľa počtu ciest, ktorými môže lúč v optickom vlákne prechádzať rozlišujeme vlákna na:

- **Jednovidové optické vlákno** (singlemode optical fiber), je tenké vlákno s vysokou prenosovou kapacitou. Na generovanie svetelného lúča sa používa LASER a jadro vlákna má hrúbku 8-10 µm. Maximálna dĺžka sa pohybuje v desiatkach kilometrov.
- **Mnohovidové optické vlákno** (multimode optical fiber), na generovanie svetelného lúča sa používajú LED. Keďže dióda vysiela všetkými smermi, tak sa generovaný lúč pri ceste vláknom odráža od okrajov jadra. Kvôli lámaniu lúču sa skracuje vzdialenosť dosahu, ktorá je pri tomto type vlákna stovky metrov. Priemer jadra u viacvidových káblov je zvyčajne 50 alebo 62,5 µm

11.1.3 Bezdrôtové prenosové médiá

Dnes používané technológie na bezdrôtový prenos sú Bluetooth, WiFi, 2G, 3G, 4G, LTE a LiFi. V minulosti sa používal aj Infračervený prenos, tomu sa však venovať nebudeme, no nájdeme ho doma napríklad v diaľkových ovládačoch.

Bluetooth

Bluetooth je rádio štandard a komunikačný protokol vytvorený pre komunikáciu pri nízkej spotrebe energie na krátku vzdialenosť, založený na lacných komunikačných mikročipoch. Zariadenia môžu spolu komunikovať ak sú v dosahu. Keďže sa používa rádiový prenos, zariadenia sa nemusia fyzicky vidieť a teda nemusia byť v tých istých miestnostiach. Bluetooth bolo formalizované 20. mája 1998.

Harald "Bluetooth" Gormsson (993 – 986) bol kráľ Dánska a neskôr aj Nórska. Zjednotil dánske kmene, vytvoril kráľovstvo a obyvateľov priviedol na kresťanstvo. V meste Jelling (Dánsko) sa nachádza runový kameň s nápisom:

ᚼᚾᚱᚾᛚᛏᚱ : ᛦᚢᚾᚢᛦᚴ : ᛒᚾᚦ : ᛦᚾᚢᚱᚢᚾ
ᛦᚢᛒᛚ : ᚦᚾᚢᛋᛁ : ᚾᛒᛏ : ᛦᚢᚱᛘ ᛒᚾᚦᚢᚱ ᛋᛁᚾ
ᚾᚢᛦ ᚾᛒᛏ : ᚦᚼᚢᚱᚢᛁ : ᛘᚢᚦᚢᚱ : ᛋᛁᚾᚾ : ᛋᚾ
ᚼᚾᚱᚾᛚᛏᚱ (:) ᛁᚾᛋ : ᛋᚼᚴ • ᚢᚾᚾ • ᛏᚾᚾᛘᚾᚱᛦ

ᚾᛚᚾ • ᚾᚢᛦ • ᚾᚢᚱᚢᛁᚾᛦ

• ᚾᚢᛦ • ᛏ(ᚾ)ᚾᛁ (• ᛦᚾᚱᚦᛁ •) ᛦᚱᛁᛋᛏᚾᚼ

haraltr : kunukʀ : baþ : kaurua kubl : þausi : aft : kurm faþur sin auk aft : þąurui : muþur : sina : sa haraltr (:) ias : sąʀ * uan * tanmaurk

ala * auk * nuruiak

* auk * t(a)ni (* karþi *) kristną

ÚLOHA – VYHĽADAJTE!

Zistite, čo je na kameňoch v meste Jelling napísané.

Iste ste si všimli, že písmeno H v runovom písme sa zobrazuje ako ᚼ a písmeno B ako ᛒ. Preložením týchto písmen cez seba dostávame logo technológie bluetooth. Technológie, ktorá zjednocuje komunikáciu medzi rôznymi zariadeniami (ako kráľ zjednotil rôzne kmene).

H (ᚼ) + B (ᛒ) =

Využitie Bluetooth je v prepojení mobilného telefónu s handsfree; bezdrôtová sieť medzi počítačmi na malom mieste a kde stačí malá šírka pásma (bandwidth); bezdrôtová komunikácia počítaču so vstupnými a výstupnými zariadeniami (myš, klávesnica, tlačiareň, …); v diaľkových ovládačoch; pripojenie ovládačov k herným konzolám; a ďalšie.

Maximálne môžeme prepojiť osem zariadení, jedno zariadenie je master a ostatné sú slave. Takúto sieť nazývame piconet, ide vlastne o ad-hoc sieť tvorenú Bluetooth zariadeniami (Na jeden master, pripadá 7 slave aktívnych zariadení). Neaktívne zariadenia sú „zaparkované“ a počúvajú na aktivačný signál. V jednej picone sieti môže byť až 255 zaparkovaných zariadení.

Wi-Fi (Wireless Fidelity)

Wi-Fi definuje podstatu bezdrôtových LAN (WLAN) založených na štandarde IEEE 802.11 vydaného v roku 1997. Predchodca Wi-Fi bol vynájdený v roku 1991 a produkty boli na trhu pod

značkou WaveLAN s rýchlosťami od 1 Mbit/s do 2 Mbit/s. Dnešné rýchlosti sú rádovo v 100 Mbit/s.

S Wi-Fi zariadením ako počítač, mobilný telefón, tablet a pod. sa môžeme pripojiť do siete LAN ak sme v blízkosti Access Pointu (AP). Oblasť pokrytá jedným alebo viacerými AP sa nazýva hotspot. Tie môžu byť veľkosťou pokrytia od pár metrov až po kilometre štvorcové (pomocou prekrývania hotspotov). Wi-Fi nám taktiež umožňuje spojenie typu peer-to-peer (tiež označované ad-hoc). Vtedy môžeme zariadenia prepojiť priamo medzi sebou bez použitia AP.

V začiatkoch predaja boli malo Wi-Fi problémy s nekompatibilitou zariadení od rôznych výrobcov a to bolo podnetom pre vznik Wi-Fi Alliance, ktorá zaviedla označenie Wi-Fi CERTIFIED. Certifikáciou je garantované, že produkty takto označené sú navzájom kompatibilné. V zozname nižšie uvádzame rôzne štandardy pre Wi-Fi (802.11), spolu s frekvenciou na ktorej sa komunikuje (uvádza sa v GHz) a maximálnou teoretickou prenosovou rýchlosťou.

- 802.11b (1999), 2,4 GHz, prenos 11 Mb/s
- 802.11a (1999), 5 GHz, prenos 54 Mb/s
- 802.11g (2002), 2,4 GHz, prenos 54 Mb/s
- 802.11n (2009), 2,4 GHz, prenos 300 Mb/s (označujeme ako WIFI 4)
- 802.11ac (2013), 2,4 aj 5 GHz, prenos 7 Gb/s (označujeme ako WIFI 5)
- 802.11ax (2020), 2,4 aj 5 GHz, 6 GHz, prenos 9,6 GB/s (označujeme ako WIFI 6)

Štandard Wi-Fi je dôležitý pre vzájomnú kompatibilitu zariadení. Všetky Wi-Fi zariadenia sú spätne kompatibilné z nižšími verziami. Pokiaľ si však zakúpime telefón s WIFI 6 a náš domáci Wi-Fi prístupový bod bude len WIFI 5, na vysokú rýchlosť môžeme zabudnúť.

Na pripojenie do WIFI siete potrebujeme poznať názov siete SSID (Service Set Identifier) a zvyčajne aj „heslo“ (WPA kľúč). Vždy volíme najvyššiu možnosť zabezpečenia, tou je v súčasnosti WPA2 (pripravuje sa WPA3). Ochránime tak našu vlastnú sieť pred prístupom nechcených zariadení.

11.2 Aktívne sieťové prvky

Poznámka pre učiteľa:

Aj tu je vhodné získať na ukážku reálne produkty. Nemusia byť funkčné.

Opakovač (Repeater)

Opakovač je aktívne sieťové zariadenie používané na predĺženie vzdialenosti medzi dvoma sieťovými zariadeniami. Jeho úlohou je zosilniť a zregenerovať signál. Signál sa zoslabuje z dôvodu útlmu signálu. Útlm (degradácia a skreslenie) signálu vzniká pri zvyšujúcej sa vzdialenosti medzi stranami komunikácie. Ide o situáciu, kedy nie je možné určiť, či signál určuje logickú nulu alebo logickú jedna. Útlm vzniká ak prepojíme zariadenia, ktoré sú ďalej ako je maximálny dosah prenosového média (TP kábel má dosah 100 m, wifi v závislosti od okolia). Opakovač jednoducho prevezme signál na vstupe pošle ho na výstup. Opakovače dnes väčšinou vidíme v domácich WiFi sieťach aby sme zvýšili pokrytie signálom.

Obrázok 142 WiFI opakovač (https://cdn.alza.sk/ImgW.ashx?fd=f3&cd=TP633g)

Prepínač (Switch)

Prepínač je zariadenie, ktoré nájdeme v domácich a firemných sieťach. Je to základný prvok počítačovej siete. Prepínač si vedie tabuľku MAC adries (jedinečné číslo sieťového zariadenia), spolu s číslom portu, ku ktorému je stanica s danou adresou pripojená.

Obrázok 143 Prepínač (switch) (http://focus.sk/netgear-gs348-48-port-gigabit-switch-fanless-19487)

Údaje zo vstupu neposiela na všetky výstupy, ale len na ten kam patria (podľa MAC adresy). Pokiaľ nastane prípad, že sa cieľová adresa v tabuľke nenachádza, alebo sú údaje určené pre všetkých (MAC adresa ff:ff:ff:ff:ff:ff), tak sa údaje pošlú na všetky porty, okrem toho, z ktorého prišli. MAC adresa sa zapisuje ako 6 skupín čísel v šestnástkovej sústave. Prvé tri čísla identifikujú výrobcu zariadenia.

ZAPAMÄTAJTE SI!

MAC adresa je celosvetovo unikátna fyzická adresa sieťového zariadenia, teda existuje iba jedno zariadenie na svete s danou MAC adresou. Na zariadení ju nájdeme vytlačenú alebo si ju vieme pozrieť cez príkazy operačného systému. Určuje ju výrobca zariadenia.

ÚLOHA – POUŽITE!

Na svojom mobilnom telefóne pohľadajte informácie o WiFi pripojení a zobrazjte si o ňom podrobnejšie informácie. Mali by ste sa dostať až k informácii o MAC adrese. Na stránke https://macvendors.com/ zadajte zobrazenú MAC a overte vyhľadaného výrobcu.

Smerovač (router)

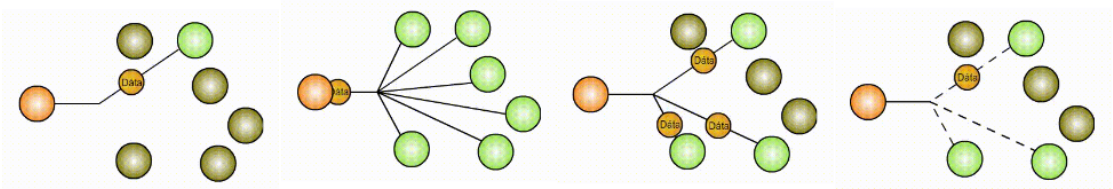
Smerovače nájdeme najmä v sieťach poskytovateľov služieb internetu, pretože ich hlavnou úlohou je prepájať dve a viac rôznych sietí. Pod rôznymi chápeme jednak rôzne technológie prenosu (káble), rôznych vlastníkov, rôzne adresy. Smerovač máme aj doma a pomocou neho sa pripájame k svojmu poskytovateľovi Internetu. Domáce smerovače prepájajú dve siete, majú dve rozhrania. Vo väčšine prípadov je na domácom rozhraní integrovaný aj niekoľkoportový prepínač.

Obrázok 144 Domáci bezdrôtový smerovač (https://cdn.alza.sk/ImgW.ashx?fd=f3&cd=TP640o)

11.3 Spôsoby komunikácie v počítačových sieťach

Zapojenie aktívnych zariadení medzi sebou nám vytvára topológiu alebo mapu siete. Vieme presne povedať, ktoré zariadenia si navzájom vymieňajú údaje ako ďaleko sú od seba. Keď sa však pozrieme na to, akým spôsobom sa prenášajú údaje, môžeme definovať nasledovné spôsoby sieťovej komunikácie (a nezáleží či sú zariadenia priamo prepojené):

- Komunikácia jeden s jedným – **unicast**
- Komunikácia jeden so všetkými – **broadcast**
- Komunikácia jeden s niektorými – **multicast**
- Komunikácia jeden s najbližším (smerovačom, DNS serverom, serverom, ...) – **anycast**



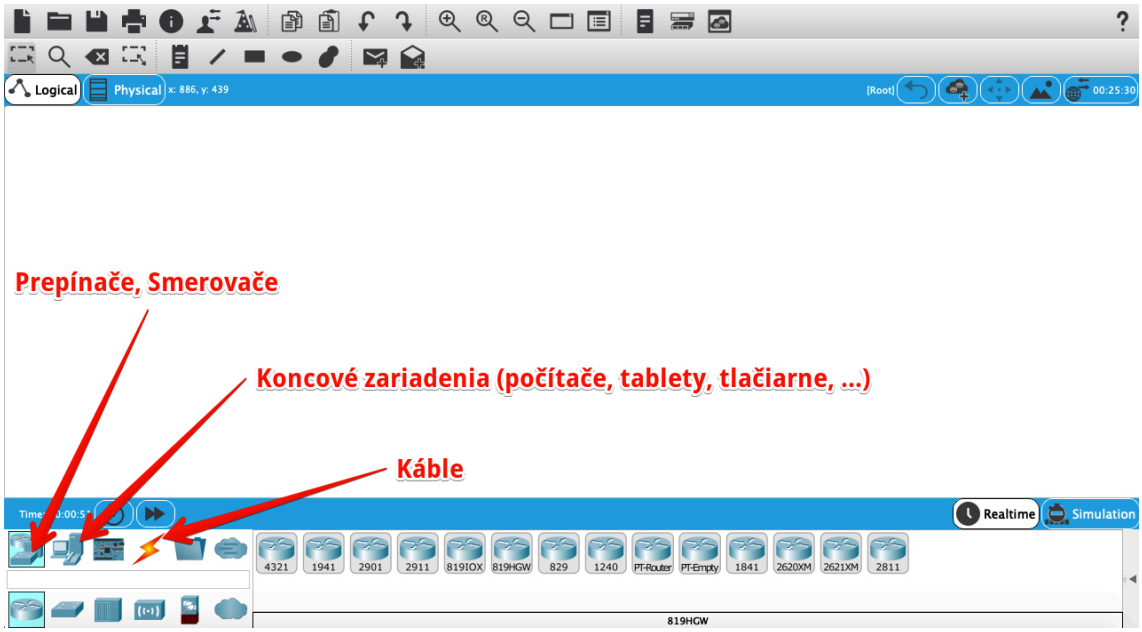
Obrázok 145 Unicast - Broadcast - Multicast – Anycast

11.4 Cisco Packet Tracer

Cisco Packet Tracer je emulačný a simulačný nástroj pre výučbu počítačových sietí. Je „voľne“ šíriteľný a dostupný pre windows, mac aj linux. Existuje tiež v mobilnej verzii. Aby sme ho mohli používať je potrebné vytvoriť si bezplatný účet na v portáli Cisco akadémie a prihlásiť sa do bezplatného kurzu pre Packet Tracer (https://www.netacad.com/courses/packet-tracer) Registrácie je jednoduchá a bez nej nie je možné Packet Tracer používať.

Poznámka pre učiteľa:

Registrácia do Netacad neprináša žiadne záväzky. Škola nemusí byť súčasťou programu Netacad. Registráciou žiaci získajú možnosť používať Packet Tracer aj doma. Pri spustení si program vyžiada zadanie mena a hesla použitého pri registrácii, preto je nutné aby si ho žiaci zapamätali. Po registrácii ponúka Netacad krátky kurz, ktorý naučí používať Packet Tracet. Je na zvážení, či učiteľ nechá na prvej hodine prejsť žiakov týmto kurzom, alebo si ho prejde učiteľ a podľa neho si upraví hodinu. Prejsť kurzom výrazne odporúčame.

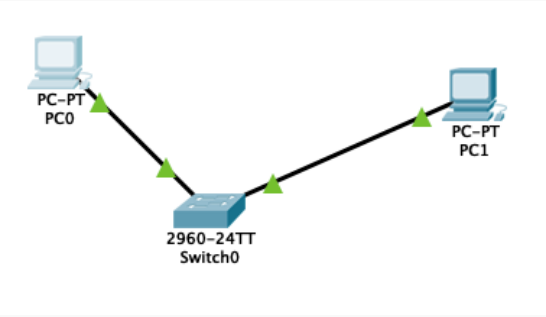


Obrázok 146 Základná obrazovka Cisco Packet Tracer

V ľavej dolnej časti obrazovky sa nachádzajú rôzne skupiny sieťových zariadení. Po ich rozkliknutí vidíme jednotlivé druhy zariadení a môžeme ich metódou chyť a ťahaj ukladať na pracovnú plochu. Rovnako máme k dispozícii aj kabeláž, ktorú sme si už spomínali.

ÚLOHA – RIEŠTE!

Vytvorte v packet tracer jednoduchú sieť zloženú z 2x PC a jeden krát prepínač. Zistite princíp akým sa jednotlivé komponenty medzi sebou zapájajú. Ako prepínač môžete použiť 2960-24TT. Aktívne a správne zapojenie je v prípade, že „LEDky“ svietia nazeleno.



Vyskúšajte zapojiť rôzne druhy káblov. Prepojte počítače aj priamo medzi sebou (bez prepínača). Aké typy káblov ste museli použiť?

IP adresa

Aby sme mohli overiť funkčnosť takto vytvorenej siete je potrebné nakonfigurovať sieť aj v operačnom systéme jednotlivých počítačov = je potrebné prideliť počítačov IP adresy.

IP adresa nám jednoznačne určuje sieťové rozhranie systému. Taktiež to znamená, že pokiaľ je v systéme viacej sieťových rozhraní, tak na každom, kde sa používa IP protokol je iná IP adresa. Je možné aj to, že na jednom sieťovom rozhraní použijeme niekoľko IP adries.

Namiesto pojmu IP adresa rozhrania sa používa označenie IP adresa počítača, je to z dôvodu, že väčšina systémov má jedno sieťové rozhranie. IP adresa je tvorená štyrmi bajtmi a v zápise ich oddeľujeme bodkou. Podľa použitej číselnej sústav používame dva zápisy IP adries:

- v dvojkovej sústave … 10101010.01010101.11111111.11111000
- v desiatkovej sústave … 170.85.255.248

Poznámka pre učiteľa:

Zápis IP adries dáva učiteľovi možnosť zopakovať prevody medzi číselnými sústavami.

Nie všetky IP adresy, ktoré by sme vedeli na 32 bitoch vyrobiť sú aj použiteľné na koncové zariadenia. Podľa spôsobu komunikácie používame nasledovné adresy

- Unicast, Anycast: 1.0.0.0 – 223.255.255.255
- Multicast: 224.0.0.0 – 239.255.255.255

Adresy od 240.0.0.0 do 255.255.255.255 sa nepoužívajú, sú odložené ako rezerva. Adresa 0.0.0.0 predstavuje „toto rozhranie“ a adresa 255.255.255.255 predstavuje „všetky zariadenia“, alebo tiež všeobecný broadcast. Poslednou špeciálnou adresou sú tie z intervalu 127.0.0.0 – 127.255.255.255. Adresu 127.0.0.1 má každé zariadenie a voláme ju **localhost**. Slúži len na komunikáciu v samotnom operačnom systéme a údaje poslané na túto adresu sa nedostávajú do počítačovej siete.

Každé sieťové zariadenie má pridelenú **unicast** adresu a tá musí byť v sieti jedinečná. Výnimku tvorí **anycast** spôsob komunikácie (viacero serverov má tú istú IP adresu), kedy smerovače v internete zabezpečia, že aby poslali údaje na iba jeden server.

IP adresy môžeme deliť aj podľa toho ako sú zadané v systéme a to na **statickú** IP- zadávame ju ručne, to znamená, že ju napevno zapíšeme do systému. Potom takto zadanú IP adresu bude sieťové rozhranie používať stále, pokiaľ ju nezmeníme.

Druhou možnosťou je **dynamická** IP – je pridelená DHCP serverom na určitý čas (hodiny, dni, týždne). IP adresu je možné na DHCP serveri rezervovať pre určité sieťové rozhranie a vtedy bude mať stále takú istú IP adresu, teda bude mať dynamicky pridelenú IP adresu ale bude sa tváriť ako statická. Toto je napríklad výhodné ak v rámci LAN máme server a chceme aby mal stálu IP adresu, ale netrváme na pevných IP adresách ostatných počítačov v sieti. V malých sieťach nám služby DHCP servera väčšinou poskytuje smerovač, ktorým sme pripojený k providerovi.

V prípade, že sa v sieti DHCP server nenachádza, počítače si vygenerujú **lokálnu (link-local) adresu** z intervalu 169.254.0.0 – 169.254.255.255.

IP adresy delíme na **verejné a súkromné (privátne)**. Verejné sa používajú na komunikáciu v Internete a v jednom čase nemôžu mať dve sieťové rozhrania pridelenú takú istú IP adresu. Na rozdiel od toho privátne IP adresy sa používajú v rámci lokálnych sietí v domácnostiach alebo vo firmách.

Výhodou je, že tieto privátne adresy sú nepoužiteľné v Internete (poskytovateľ ich nesmie do Internetu pustiť) a tým pádom sa ani z internetu nedostaneme priamo na privátnu adresu. Dosiahneme tým akúsi bezpečnosť domácej siete. Druhou výhodou je, že o pridelenie týchto adries nie je treba nikoho žiadať (verejnú si žiadame o poskytovateľ a platíme za ňu)

Máme tri rozsahy privátnych IP adries:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255

ÚLOHA – RIEŠTE!

Na ľubovoľnom zariadení otvorte web https://www.moja-ip.sk/. Aká je vaša IP adresa? Porovnajte ju s IP adresou, ktorú vidíte v nastaveniach sieťového rozhrania.

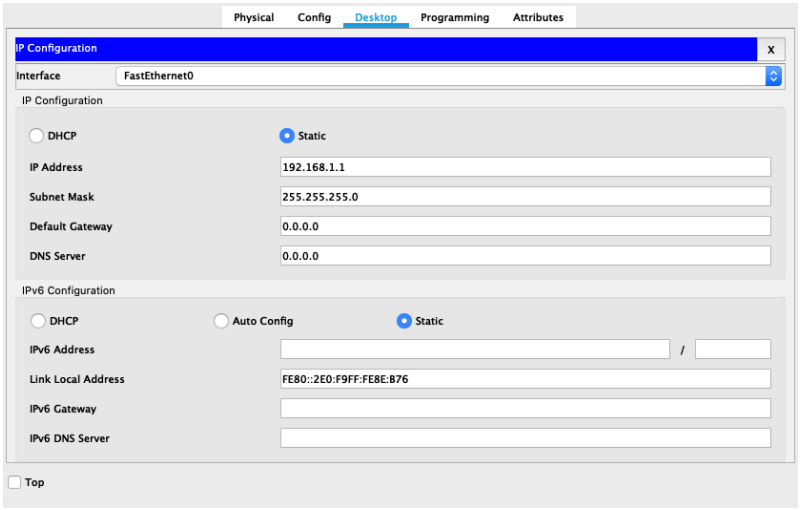
Sieťová maska

Okrem IP adresy nastavujeme na rozhraní aj sieťovú masku. Sieťová maska je štvorbajtové číslo, ktoré nám určuje, ktorá časť IP adresy je spoločná pre všetky zariadenia v jednej sieti (adresa siete) a ktorá určuje už konkrétne sieťové zariadenie. Príkladom sieťovej masky je 255.255.255.0

- Počítač 1 má adresu 192.168.1.1 s maskou 255.255.255.0
- Počítač 2 má adresu 192.168.1.2 s maskou 255.255.255.0
- Počítač 3 má adresu 192.168.2.1 s maskou 255.255.255.0

- Počítač 4 má adresu 192.168.2.2 s maskou 255.255.255.0

Ak by boli počítače 1 až 4 pripojené na spoločný prepínač, komunikovať by mohli len PC1 a PC2, alebo PC3 a PC4. Iná komunikácie nie je možná, lebo PC1 a PC2 sú v sieti 192.168.1.0 a počítače PC3 a PC4 sú v sieti 192.168.2.0. IP adresu v Pakcet Tracer nastavíme po kliknutí na počítač, záložka *Desktop*, ikona *IP Configuration*.



Obrázok 147 Nastavenie IP adresy

ÚLOHA – RIEŠTE!

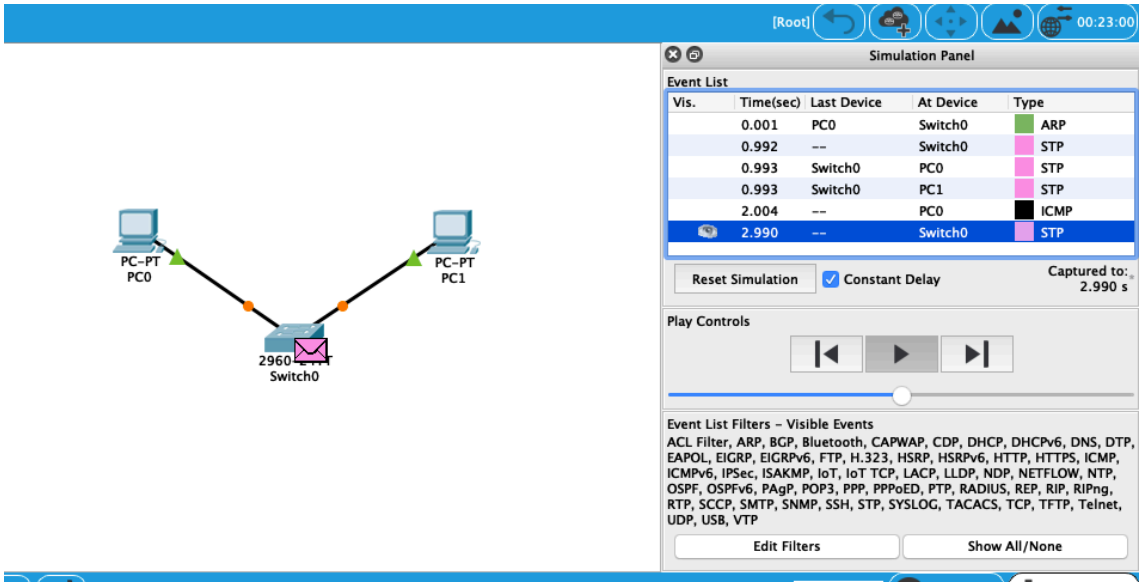
Vytvorte sieť so štyrmi počítačmi. IP adresy nastavte nasledovne:

- Počítač 1 má adresu 192.168.1.1 s maskou 255.255.255.0
- Počítač 2 má adresu 192.168.1.2 s maskou 255.255.255.0
- Počítač 3 má adresu 192.168.2.1 s maskou 255.255.255.0
- Počítač 4 má adresu 192.168.2.2 s maskou 255.255.255.0

Na jednom z počítačov otvorte *„Command prompt“* na záložke *Desktop*. Pomocou príkazu **ping** otestujte dostupnosť ostatných počítačov. Sledujte rozdiel vo výpise.

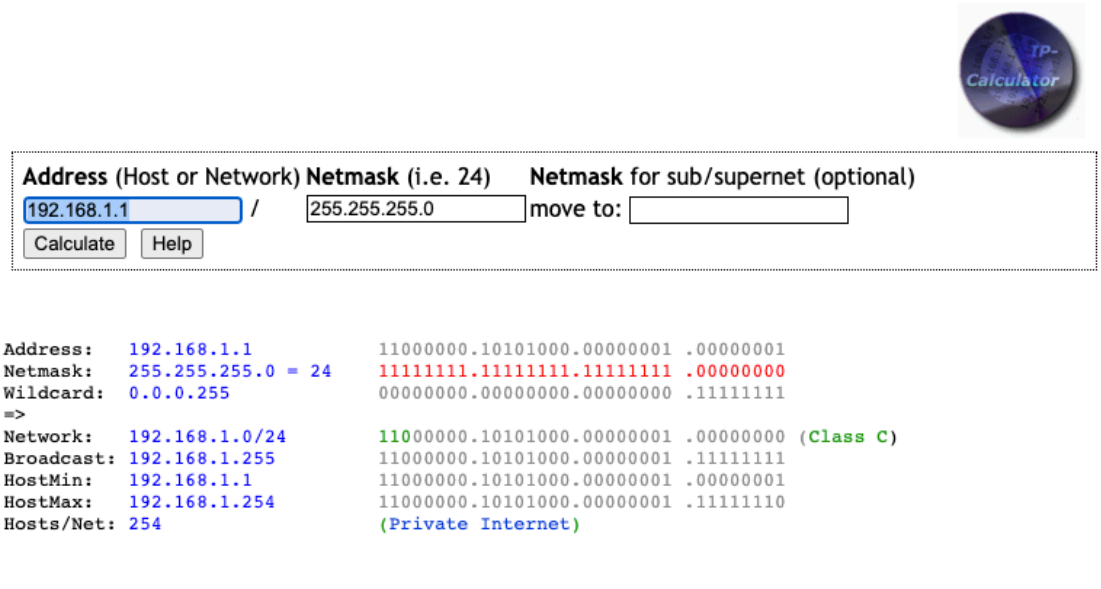
Príklad: **ping 192.168.1.2**

Packet tracer nám umožňuje aj podrobne sledovať ako sa údaje v sieti správajú. Slúži na to simulačný režim. Ten zapíname v pravom dolnom rohu cez tlačidlo „Simulation“. Zopakujte prechádzajúcu úlohu v simulačnom režime. Po skončení sa prepnite do „Realtime“.



Obrázok 148 Simulačný režim Packet Tracer

V našej fyzickej sieti sme vytvorili dve logické siete. Majú rôzne IP adresné rozsahy (intervaly). V počítačovej sieti môžu medzi sebou priamo komunikovať len zariadenia, ktoré patria do jednej siete. Aby sme zistili, či dve adresy patria do jednej siete je potrebné poznať hranice adresných rozsahov siete. Môžeme si pomôcť kalkulátorom dostupným na http://jodies.de/ipcalc.



Obrázok 149 IP kalkulátor http://jodies.de/ipcalc

Všimnime si riadok Broadcast. V predchádzajúcom texte sme si už broadcast spomínali. Tentokrát je však adresa iná, voláme ju sieťový broadcast, avšak jej význam je rovnaký ako všeobecný broadcast. Slúži na adresovanie všetkých počítačov v danej IP sieti. Všeobecný broadcast adresuje všetky počítače vo všetkých sieťach až po najbližší smerovač.

ZAPAMÄTAJTE SI!

Sieťová maska zapísaná v binárnom tvare je vždy postupnosť jednotiek, za ktorou nasleduje postupnosť núl.

Ak by sme zmenili masku na všetkých počítačoch na 255.255.0.0, patrili by nám všetky IP adresy do jedného rozsahu a naše počítače by vedeli medzi sebou komunikovať.

ÚLOHA – RIEŠTE!

Zmeňte masku na všetkých počítačoch na 255.255.0.0 a otestujte dostupnosť počítačov.

Príklad: **ping 192.168.1.2**

Poznámka pre učiteľa:

Do siete môžeme pridať aj server. Adresu mu pridelíme rovnako ako počítačom. V základnom nastavení funguje ako webový server a z počítača môžeme cez prehliadač otvoriť stránky, ktoré sú na ňom uložené. Prehliadač je na záložke *Desktop*.

Zmena masky aby nám počítače komunikovali nie je štandardným riešením. Aby sme prepojili dve siete, potrebujeme smerovač.

ÚLOHA – RIEŠTE!

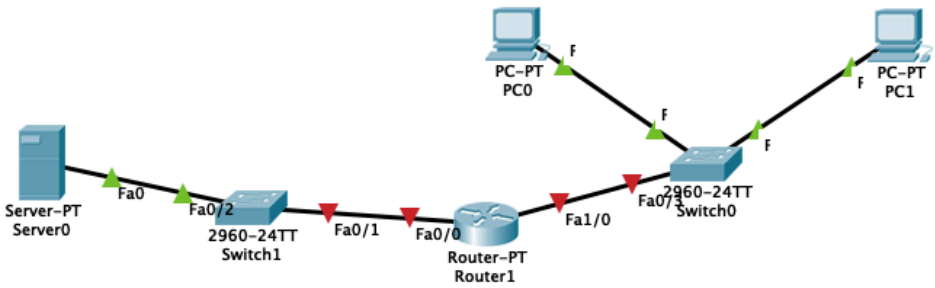
Pridajte do siete smerovač (PT-Router), prepínač a server. Serveru prideľte nasledovné adresy

- IP: 10.0.0.10
- Maska: 255.255.255.0
- Brána (gateway): 10.0.0.1

Na všetkých ostatných počítačoch pridajte bránu s hodnotou 192.168.100.100. Káble zapojte podľa obrázka nižšie.

Poznámka pre učiteľa:

V tomto momente je vhodné nastaviť zobrazovanie mien sieťových rozhraní. Nastavenie nájdeme v ponuke *Options – Preferences – Always show port labels in logical workspace*



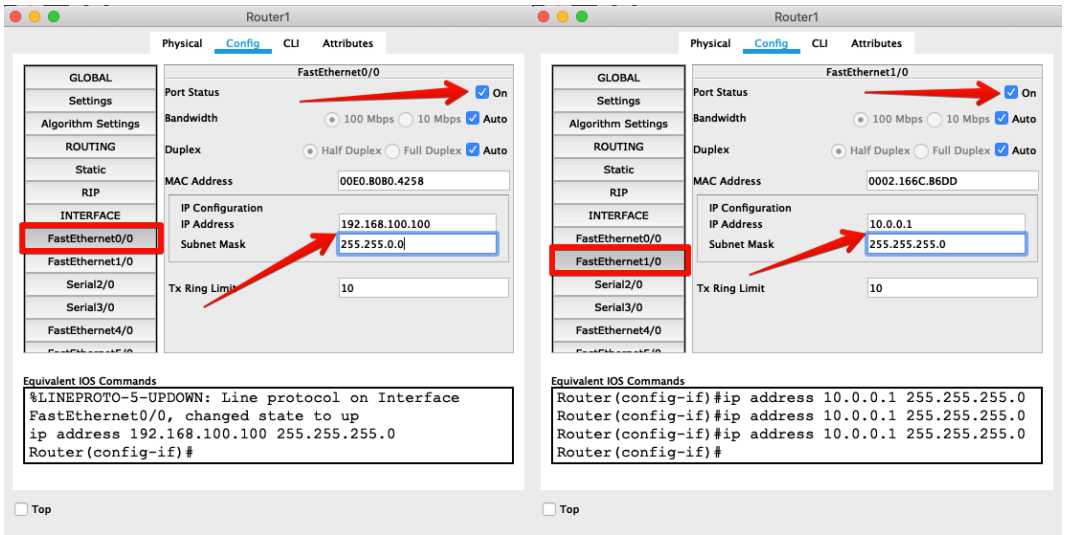
Obrázok 150 Sieť s pridaným smerovačom

Na všetkých zariadeniach sme nastavili bránu. V celej sieti majú všetky zariadenia rovnakú adresu brány. Je to preto, lebo brána je IP adresa smerovača, ktorým sa pripájajú do iných sietí. Smerovač bude mať teda dve IP adresy. Na rozhraní Fa0/0 bude mať adresu 10.0.0.1 (255.255.255.0), na rozhraní Fa0/1 bude mať adresu 192.168.100.100 (255.255.0.0).

Poznámka pre učiteľa:

Ak žiaci zapojili smerovač inak ako je na obrázku, nie je to chyba. Len si musia dať pozor, ktorému rozhraniu nastavujú akú adresu.

Na smerovači nám svieti stav pripojenia načerveno. Rozhranie smerovača musíme zapnúť ručne a spravíme to zároveň s nastavením IP adresy po kliknutí na smerovač na karte *Config*. V ľavom stĺpci vyberieme rozhranie FastEthernet 0/0, zvolíme Port Status „On“, nastavíme IP adresu a masku. Postup zopakujeme aj pre rozhranie FastEthernet 1/0.



Obrázok 151 Nastavenie smerovača

Po úspešnom nastavení smerovača je možné zrealizovať test dostupnosti medzi jednotlivými sieťami.

ÚLOHA – RIEŠTE!

Na serveri otvorte príkazový riadok z spustite príkaz **ping 192.168.1.1**

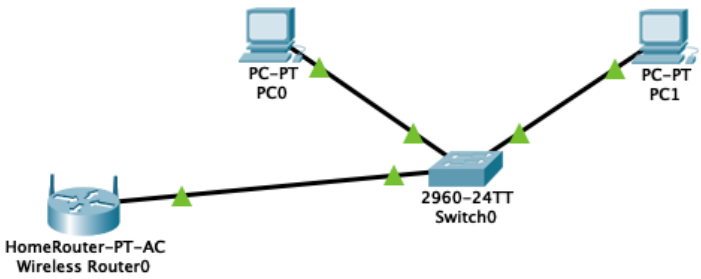
Na počítači otvorte prehliadač a zobrazte obsah webu na serveri: **http://10.0.0.10/**

Na počítači otvorte príkazový riadok a zadajte príkaz **tracert 10.0.0.10**

Príkaz **tracert** nám zobrazí zoznam smerovačov, cez ktoré údaje prešli k cieľu. Tento príkaz je štandarným v systémoch windows aj linux. Môžete skúsiť vykonať ho na počítači v škole alebo doma. Cieľ nemusí byť len IP adresa ale ľubovoľné doménové meno, napr. www.tokyometro.jp

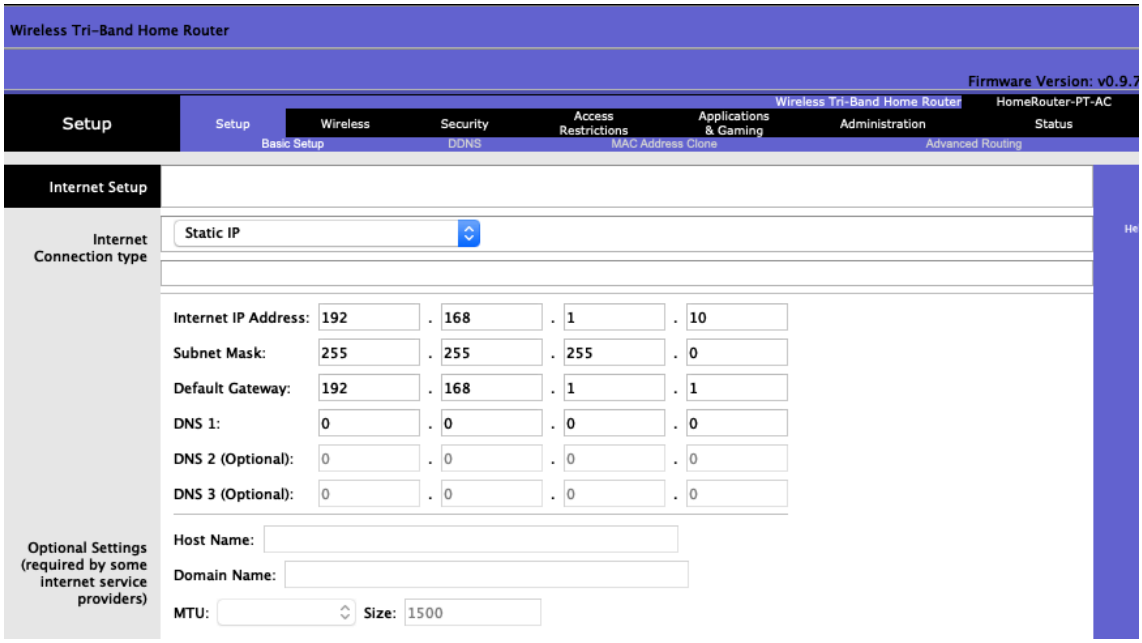
Packet Tracer nám tiež ponúka množstvo bezdrôtových zariadení. Vyskúšajme do našej siete pridať domáci WiFi smerovač. Jeho názov je „Home Router“ a nájdeme ho v skupine „Wireless Devices“. Keďže smerovač má viac rozhraní, nie je dobré nechať pripojenie káblov na automatiku (ikona blesku), ale pripojíme ich ručne.

- Vyberieme typ kábla
- Klikneme na smerovač a vyberieme rozhranie „Internet“
- Klikneme na prepínač a vyberieme ľubovoľné rozhranie „FastEthernet“



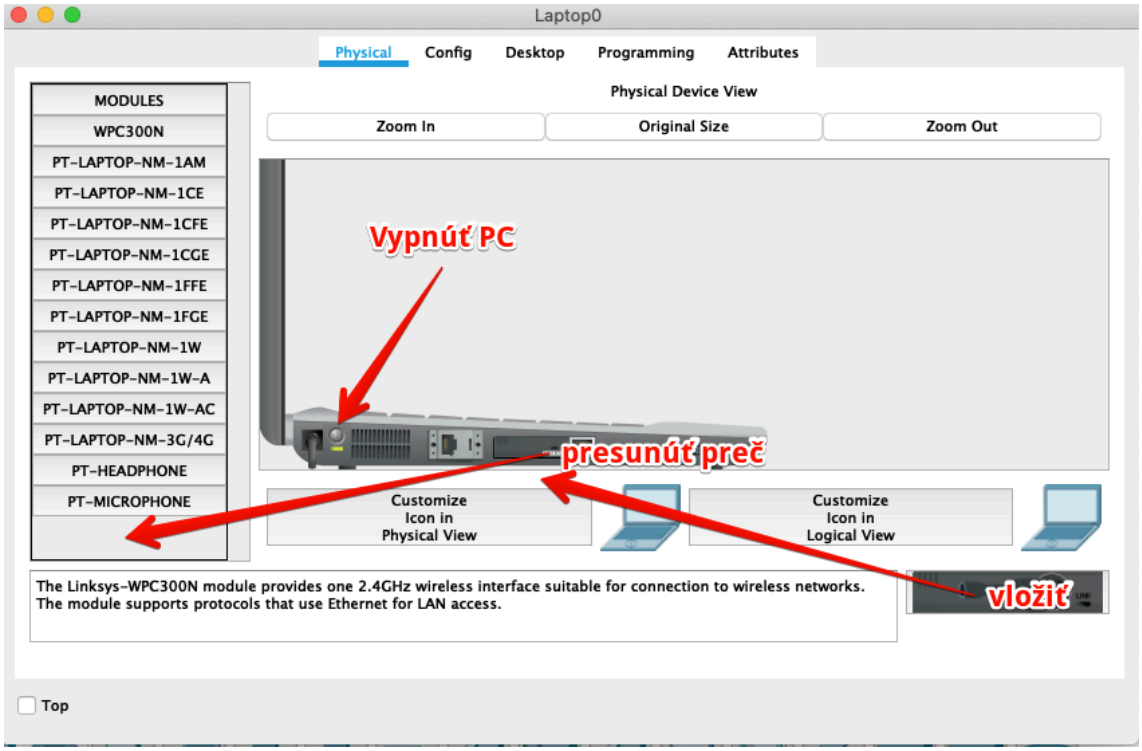
Obrázok 152 Pripojenie domácej WiFi

Po rozkliknutí smerovača je dostupná karta „GUI“ obsahujúce rôzne možnosti nastavenia smerovača. Prvé čo nastavíme je statická IP adresa na internetovom rozhraní (nemáme tu providera, ktorý by nám pridelil adresu).



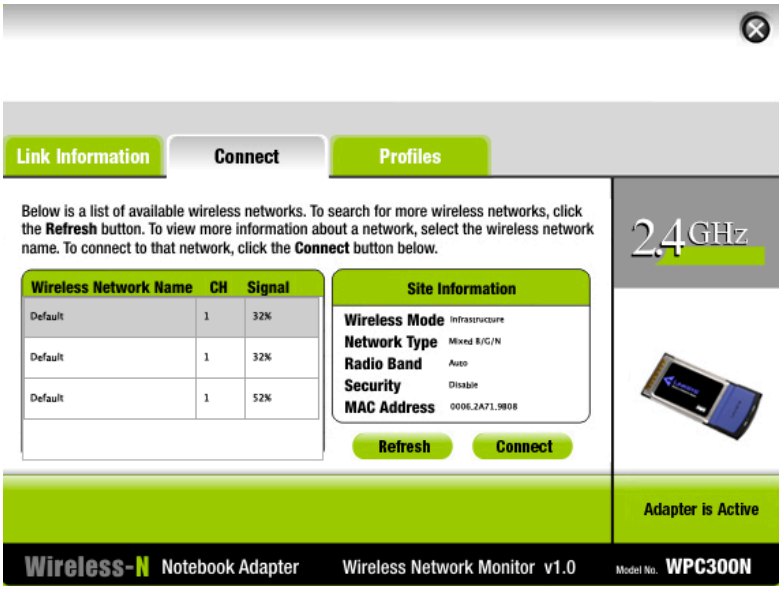
V ďalšom kroku nastavíme WiFi sieť na záložke „Wireless“. Zvolíme si názov siete SSID a na záložke „Wireless Security“ zabezpečenie tejto siete. Typ zabezpečenia zvolíme WPA2 Personal a nastavíme heslo, ktoré si zapamätáme. Na konci obrazovky s nastaveniami sa nachádza tlačidlo „Save Settings“.

Po tomto nastavení môžeme pripojiť bezdrôtového klienta - *Laptop.* Predtým ho však musíme na Wi-Fi pripraviť. Vypneme zariadenie, vyhodíme sieťovú kartu a vložíme Wi-Fi kartu. Nesmieme zabudnúť znova zapnúť zariadenie.



Obrázok 153 Príprava Laptop na pripojenie na Wi-Fi

Teraz máme na karte Desktop nástroj na konfiguráciu Wi-Fi „PC Wireless“. Tá nám vyhľadá dostupné Wi-Fi siete a umožní zadať heslo pre SSID.



Obrázok 154 Nástroj na pripojenie k Wi-Fi

ZHRNUTIE

Čo sme sa naučili

- Základné časti počítačových sietí sú prenosové médiá a aktívne zariadenia.
- Každé zariadenie má celosvetovo jedinečnú MAC adresu.
- Každé zariadenie má pridelenú IP adresu, masku a bránu. Verejné IP adresy sú jedinečné, privátne sú jedinečné len v danej sieti.
- Na pripojenie k Wi-Fi potrebujeme SSID a WPA kľúč.
- Prepojenie sietí umožňuje smerovač. IP adresa smerovača je bránou pre počítače v jeho sieti.

12 SLUŽBY POČÍTAČOVEJ SIETE

CIEĽ

Cieľom tejto kapitoly je spoznať základné služby siete Internet. Jedná sa o služby bez ktorých si dnešný Internet nevieme predstaviť alebo služby, bez ktorých by ani nefungoval. Žiak bude vedieť popísať a vysvetliť ich základné princípy, bude vedieť na vzdialenom systéme identifikovať zoznam poskytovaných služieb.

MOTIVÁCIA

Zamysleli ste sa už nad tým, ako počítač vie aká IP adresa patrí k akému názvu webu? Akým spôsobom sa prenáša email a kam sa uloží kým si ho neprečítame? Akým spôsobom vie hacker, že máme zraniteľný systém? Na tieto otázky nájdeme odpovede v ďalšom texte.

VÝKLAD

12.1 Systém doménových mien

Systém DNS vznikol v roku 1984. DNS je skratka, ktorá vznikla zo slov Domain Name System. Je to služba Internetu, ktorá slúži na preklad doménových mien na IP adresy. Každý IP datagram je v Internete smerovaný na základe cieľovej IP adresy. Keďže IP adresy sa pre človeka ťažko pamätajú, používajú sa tzv. doménové mená. Nie každé sieťové rozhranie však musí mať pridelené svoje doménové meno.

Pokiaľ sieťové rozhranie nemá pridelené doménové meno, môžeme napísať **http://194.160.210.18/** alebo môžeme poslať e-mail na adresu **user@[194.160.210.18]**

Pre pochopenie systému DNS sa musíme oboznámiť s pojmami:

- doména
- doménové meno
- reverzná doména
- zóna
- name server

DOMÉNA

Je skupina mien, ktoré k sebe logicky patria. V rámci domény je možné vytvárať podskupiny, tzv. subdomény. Z jednotlivých mien podskupín je potom vytvorené doménové meno uzla (napr.: prometheus.ukf.sk.). Pri adresovaní počítača v Internete uvádzame jeho doménové meno.

Prideľovanie domén a pravidlá domén musí niekto strážiť. Takou organizáciou je ICANN - The Internet Corporation for Assigned Names and Numbers. Je to technické koordinačné centrum pre Internet, podporuje a zabezpečuje Domain Name System a umožňuje budovať hierarchiu doménových mien.

Domény najvyššej úrovne rozdeľujeme na generické a kódy krajín.

Generické domény

- .com, .net, .org
- .edu, .gov, .int, .mil
- .biz, .info, .name, .pro
- .aero, .coop, .museum
- .asia, .cat, .jobs, .mobi, .tel, .travel, .pro
- .arpa
- .post
- .xxx

Prvých 7 domén existuje od počiatku Internetu. Tie ostatné sa pridali postupom času. Dnes už existujú stovky nových generických domén. Ich zoznam nájdeme na https://newgtlds.icann.org/en/program-status/delegated-strings

Domény krajín

Domény krajín sú zhodné s dvojpísmenovým označením krajiny podľa normy ISO 3166.

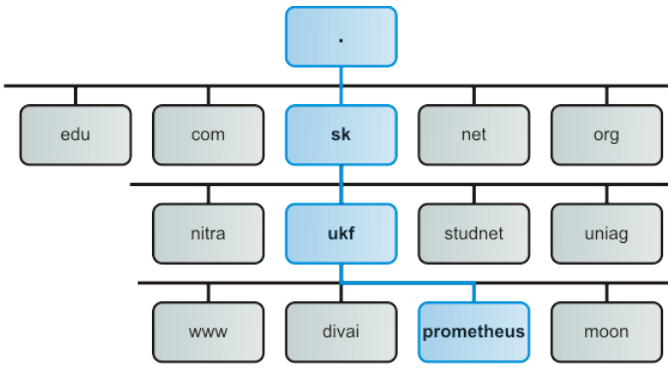
- .ac .ad .ae .af .ag .ai .al .am .an .ao .aq .ar .as .at .au .aw .ax .az .ba .bb .bd .be .bf .bg .bh .bi .bj .bm .bn .bo .br .bs .bt .bv .bw .by .bz .ca .cc .cd .cf .cg .ch .ci .ck .cl .cm .cn .co .cr .cu .cv .cx .cy .cz .de .dj .dk .dm .do .dz .ec .ee .eg .eh .er .es .et .eu .fi .fj .fk .fm .fo .fr .ga .gb .gd .ge .gf .gg .gh .gi .gl .gm .gn .gp .gq .gr .gs .gt .gu .gw .gy .hk .hm .hn .hr .ht .hu .id .ie .il .im .in .io .iq .ir .is .it .je .jm .jo .jp .ke .kg .kh .ki .km .kn .kp .kr .kw .ky .kz .la .lb .lc .li .lk .lr .ls .lt .lu .lv .ly .ma .mc .md .me .mg .mh .mk .ml .mm .mn .mo .mp .mq .mr .ms .mt .mu .mv .mw .mx .my .mz .na .nc .ne .nf .ng .ni .nl .no .np .nr .nu .nz .om .pa .pe .pf .pg .ph .pk .pl .pm .pn .pr .ps .pt .pw .py .qa .re .ro .rs .ru .rw .sa .sb .sc .sd .se .sg .sh .si .sj .sk .sl .sm .sn .so .sr .st .su .sv .sy .sz .tc .td .tf .tg .th .tj .tk .tl .tm .tn .to .tp .tr .tt .tv .tw .tz .ua .ug .uk .um .us .uy .uz .va .vc .ve .vg .vi .vn .vu .wf .ws .ye .yt .yu .za .zm .zw

ÚLOHA 2 – RIEŠTE!

Niektoré krajiny majú doménové meno, ktoré slúži na úplne iný účel. Skúste zistiť, aké domény majú krajiny, ktorým patria nasledovné ostrovy – Ellice Islands vľavo a Caroline Islands vpravo.

DOMÉNOVÉ MENO

Skladá sa z reťazcov vzájomne oddelených bodkou. Meno sa preskúmava smerom sprava doľava. Najvyššou inštanciou v mene je root doména a je označená samostatnou bodkou. V root doméne sú definované domény najvyššej úrovne (TLD, Top Level Domain). Medzi TLD patria domény aero, arpa, biz, com, coop, edu, gov, info, mil, museum, name, net, org, pro a dvojznakové kódy štátov definovaných podľa normy ISO-3166 (sk, cz, at, atď). Pod TLD doménami sú definované ďalšie hierarchicky nižšie domény. Systém domén tvorí strom. Príkladom doménového mena je prometheus.ukf.sk.



Obrázok 155 Stromová štruktúra doménových mien

Doménové meno má svoju presne definovanú syntax: **reťazec.reťazec.reťazec.TLD.**

Každý reťazec môže mať maximálne 63 znakov, pričom celá dĺžka mena nesmie presiahnuť 255 znakov. V doménovom mene sú povolené nasledovné znaky:

- veľké a malé písmena anglickej abecedy [a..z][A..Z],
- číslice [0..9],
- pomlčka [–].

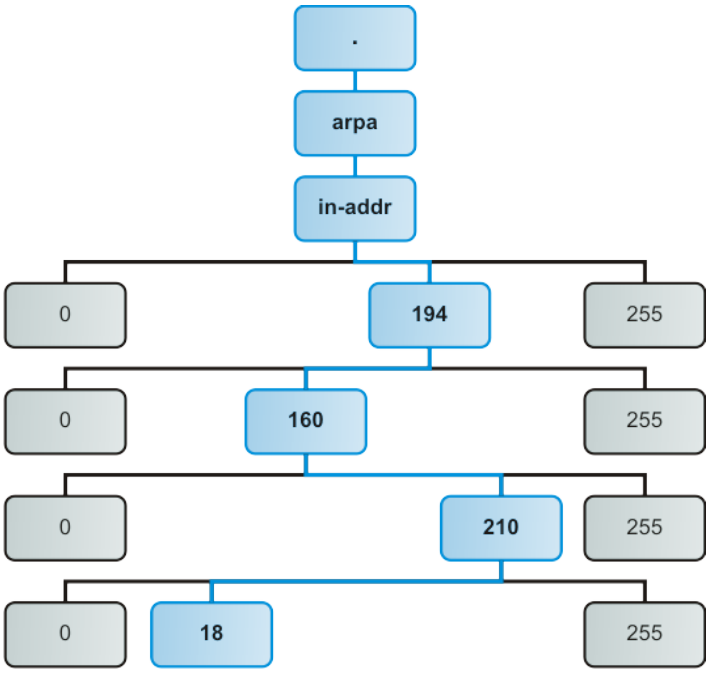
Pomlčka pritom nesmie byť na začiatku ani na konci reťazca.

REVERZNÉ DOMÉNOVÉ MENÁ

Reverzné domény slúžia na vyhľadávanie doménového mena na základe znalosti IP adresy cieľového počítača. Pre reverzné doménové mená sú vyhradené dve mená domén najvyššej úrovne:

- pre IP adresy verzie 4 je to doména in-addr.arpa. (inverse addresses in the arpanet)
- pre IP adresy verzie 6 je to doména ip6.arpa.

Príkladom reverzného doménového mena je 18.210.160.194.in-addr.arpa., alebo a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.6.0.8.1.1.4.1.0.0.2.ip6.arpa.



Obrázok 156 Stromová štruktúra reverzných doménových mien

REZERVOVANÉ DOMÉNOVÉ MENÁ

Pre rôzne účely boli vyhradené nasledovné mená TLD:

- .test – je rezervovaná pre testovacie účely, pre aktuálne alebo nové DNSzáznamy
- .example – je odporúčané používať ju pri písaní dokumentácie
- .invalid – je určená pre doménové mená, ktoré sú na prvý pohľad nesprávne
- .localhost – doménové meno vyhradené pre lokálnu softvérovú slučku

ZÓNA

Údaje o jednej doméne nemusia byť uchovávané na jednom name servery. Databázu doménových mien môžeme rozdeliť na niekoľko name serverov, pričom každý name server bude obsahovať iba časť údajov o doméne. Údaje uložené na jednom name servery nazývame zóna. Name server nespravuje teda doménu, ale zónu, aj keď vo veľa prípadoch je súbor domény a zóny ten istý.

NAME SERVER

Name server prekladá doménové mená na IP adresy a opačne. Name servery rozdeľujeme do štyroch skupín:

Primárny name server

- Na primárnom name servery sú uložené všetky údaje o danej zóne.
- Primárny name server môže byť len jeden.
- Akékoľvek zmeny zóny sa vykonávajú na primárnom name servery.

Sekundárny name server

- Sekundárny name server preberá v určitých časových intervaloch údaje o zóne od primárneho name servera.
- Počas výpadku alebo nedostupnosti primárneho name servera poskytuje autoritatívne odpovede o zóne, ktorú spravuje.
- Počet sekundárnych name serverov nie je obmedzený.

Caching only name server

- Nie je pre žiadnu doménu primárnym, ani sekundárnym name serverov, avšak využíva všeobecné vlastnosti name servera. To znamená, že do svojej pamäte ukladá údaje, ktoré ním prechádzajú. Väčšinou ho nájdeme v domácom smerovači.

Root name server

- Root name server je špeciálny typ name servera. Je základným kameňom celého DNS systému.
- Spravuje údaje o doménach najvyššej úrovne.
- Každý z root serverov je primárnym name serverom, aj keď každý obsahuje rovnakú databázu údajov o zóne.
- Vo svete existuje 13 root name serverov. Ich zoznam nájdeme na adrese http://www.root-servers.org/. V skutočnosti ich nie je 13, avšak pod jedným menom sa skrývajú desiatky fyzických serverov.
- Výpadok týchto name serverov by znamenal výpadok siete Internet na úrovni DNS. Bez znalosti IP adresy cieľa by sme nevedeli adresovať počítač v sieti.

HOSTS SÚBOR

Pozostatkom z čias keď neexistovalo DNS je jednoduchý textový súbor nachádzajúci sa v každom operačnom systéme. V linuxoch ho nájdeme uložený ako **/etc/hosts**, vo windowsoch v **c:\windows\system32\drivers\etc\hosts**

Syntax súboru je jednoduchá, v každom riadku je IP adresa a k nej prislúchajúce doménové meno. Po inštalácii je tam len jeden záznam: 127.0.0.1 localhost. Údaje v hosts súbore majú prednosť pred DNS systémom a preto nesprávne zadané údaje v hosts môžu mať za následok,

že sa nám daný web nezobrazí. Zámerne nesprávne údaje sú aj overeným trikom ako sa vyhnúť zasielaniu údajov na licenčné servery. Nelegálne nainštalovaná aplikácia sa pokúša kontaktovať licenčný server no pre ten je v *hosts* zapísaná neexistujúca adresa.

ÚLOHA – RIEŠTE!

Zobrazte obsah hosts súboru. V prípade, že máte dostatočné oprávnenia dopíšte do súboru riadok

127.0.0.1 www.facebook.com

ZÁKLADNÉ TYPY DOMÉNOVÝCH ZÁZNAMOV

Systém doménových mien umožňuje administrátorom zadať do súboru zóny okrem IP adresy aj množstvo iných údajov. Sú nimi napríklad názvy poštových serverov, pravidlá pre spracovanie SPAMu, verejný kľúč pre šifrovanie a pod.

Najzákladnejším typom záznamu je informácia o IP adrese. Tá môže byť verzie 4, vtedy hovoríme **o zázname typu A**, alebo môže byť verzie 6, a vtedy hovoríme o **AAAA zázname**. Každá doména tiež uvádza informáciu o tom, aký server v Internet je jej poštovým serverom (**MX záznam**). Tiež vymenúvame všetky primárne a sekundárne DNS servery obsahujúce súbor zóny (**NS záznam**). Všetky tieto informácie vieme získať pomocou nástrojov pre prácu s DNS.

NÁSTROJE NA PRÁCU S DNS

nslookup

Nslookup je program, ktorým odosielame DNS dotazy na name server. Nslookup je štandardne implementovaný v systémoch UNIX aj WINDOWS. Program nslookup sa spúšťa príkazom nslookup. Po spustení sa program pripojí na predvolený name server a očakáva príkazy od používateľa.

ÚLOHA – RIEŠTE!

Zistite IP adresu domény www.ukf.sk.

Spustíme program nslookup (start –> run –> nslookup). Príkazom set debug nastavíme podrobnejší výpis informácií. Zadáme názov domény, ktorú hľadáme: www.ukf.sk

```
C:\Users\petko>nslookup www.ukf.sk
Server:  dns.google
Address:  8.8.4.4

Non-authoritative answer:
Name:    fakulty.ukf.sk
Address:  193.87.12.30
Aliases:  www.ukf.sk


C:\Users\petko>
```

Obrázok 157 Výpis z programu nslookup pre doménu www.ukf.sk

whois

Whois je program, ktorým odosielame dotazy na WHOIS servery, spravujúce WHOIS databázu. Táto databáza obsahuje o každej zaregistrovanej doméne údaje o jej vlastníkovi (meno, adresu, kontaktné adresy), registrátorovi (meno, kontakt) a adresy autoritatívnych name serverov pre danú doménu. Whois je štandardne implementovaný iba na systémoch typu UNIX. Používatelia WINDOWS môžu využiť webové rozhranie whois databázy na adrese www.whois.net

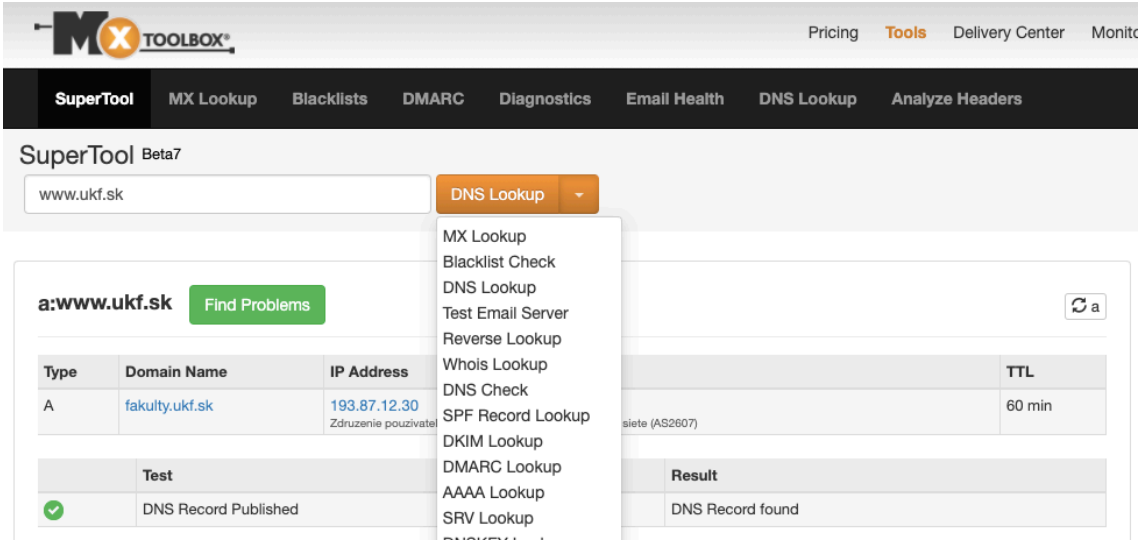
Vo výpise vidíme aj takú informáciu ako je vlastník domény a jeho registrátor, platnosť domény a podobne.

```
Domain:                     ukf.sk
Registrant:                 UNIV-0040
Admin Contact:              UNIV-0040
Tech Contact:               UNIV-0040
Registrar:                  UNIV-0040
Created:                    2003-10-01
Updated:                    2019-04-29
Valid Until:                2027-09-25
Nameserver:                 ns.ukf.sk
Nameserver:                 ns2.ukf.sk
Nameserver:                 ns3.ukf.sk
Nameserver:                 ns1.bbnw.sk
DNSSEC:                     37585 8 1 B3881EA695658A5B4738DB23F4E30C4C26BB9C4C
DNSSEC:                     37585 8 2 C53800B998BDF8F3F324C7AB101C2B0CF7B3F49A7D0F34DAA1F84C
EPP Status:                 ok

Registrar:
Name:                       Univerzita Konštantína Filozofa v Nitre
Organization:               Univerzita Konštantína Filozofa v Nitre
Organization ID:            157716
Phone:                      +421.907670343
Email:                      sk-nic@ukf.sk
Street:                     Trieda Andreja Hlinku 1
City:                       Nitra
Postal Code:                94974
Country Code:               SK
Created:                    2017-09-01
Updated:                    2020-02-05
```

Obrázok 158 WHOIS informácie pre doménu ukf.sk

Alternatívne nástroje na získavanie informácií z DNS

Nie vždy máme po ruke príkazový riadok počítača a radi by sme získali informácie z DNS. Pomôže nám veľa online služieb, ktoré sú však rôzne svojou kvalitou alebo mierou reklamy. Odporúčame použiť web https://mxtoolbox.com/, kde nájdeme množstvo užitočných nástrojov.



Obrázok 159 Online nástroj MXTOOLBOX

Okrem už spomínaného prevodu doménového mena na IP adresu nám tento nástroj umožňuje spraviť prevod z IP adresy na doménové meno = vyhľadať reverzný DNS záznam. Ten je veľmi dôležitý najmä pri odosielaní elektronickej pošty. Vysvetlíme si to na nasledovnom príklade:

E-maily pre doménu studuj.it odosiela server s doménovým menom amos.teacher.sk. Jeho IP adresa je 109.74.145.53. Niekto sa bude snažiť vytvoriť podvodný email (phishing útok) a z nejakého mailového servera (napr. s IP adresou 212.34.56.78) odošle mail, ktorý sa bude tváriť, že je z domény studuj.it a doručí nám to na gmail. Google mailový server si však overí (opýta sa DNS systémy, aký je MX záznam pre doménu), aký mailový server môže posielať maily za domény studuj.it a zistí, že je to amos.teacher.sk

12.2 Architektúra elektronickej pošty

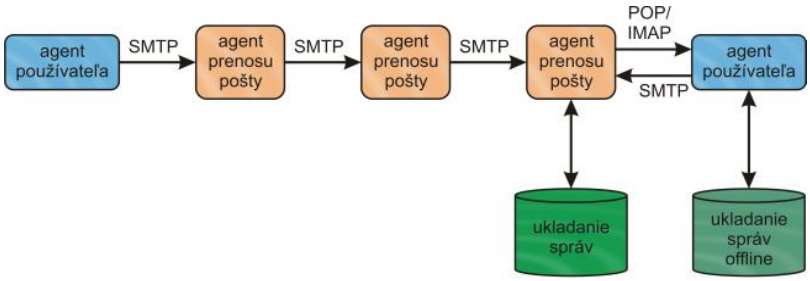
Jednou zo služieb siete Internet je posielanie elektronickej pošty (e-mail). Princípom je posielanie správ medzi poštovými (e-mailovými) servermi (mail exchange server – MX v DNS, ktorých funkciou je správu uložiť a poslať ďalej. Teda správa, ktorú posielame, môže pri svojom „putovaní“ Internetom prejsť viacerými servermi (resp. MTA, povieme si ďalej vtexte).

E-mail vznikol na prelome rokov 1971-72 a jeho tvorcom bol Ray Tomlinson, ktorý zaviedol používanie znaku @. Pred tým sa správy, informácie posielali tak, že sa používatelia prihlásili na spoločný file server a tu ukladali súbory pre ostatných.

Zo širšieho hľadiska sa na e-mailové správy môžeme pozerať dvoma spôsobmi. Ide o e-maily vo vnútri firiem (v rámci intranetu), takéto e-maily by v žiadnom prípade nemali opúšťať poštový server firmy a mali by sa priamo presunúť do inboxu príjemcu. Druhou väčšou skupinou sú e-mailové správy posielané cez Internet, ktoré teda opúšťajú svoj domovský server. Ale štruktúra správ pri oboch typoch je rovnaká.

Dnešné posielanie e-mailov znamená, že si v poštovom klientovi – agent používateľa (mail user agent - MUA) napíšeme správu. Poštových klientov je viacej a závisí od používateľa, pre ktorý sa rozhodne, sú to napr. Mozilla Thunderbird, MS Outlook, a iné. Potom sa táto aplikácia spojí s poštovým serverom (domovský server), ten je buď na našej LAN alebo ide o server u nášho

providera, prípadne môže ísť a webmail server. Spojenie, resp. následné odoslanie správ na server sa realizuje pomocou TCP protokolu a SMTP protokolu. Taktiež nám na domovský server aj pošta prichádza a poštový klient si ju na náš systém stiahne - pomocou protokolu TCP (vytvorí spojenie) a potom použije protokol POP3 alebo IMAP.



Obrázok 160 Princíp elektronickej pošty

Poštový server odosiela e-mailové správy pomocou agenta prenosu pošty (mail transfer agent – MTA) použitím protokolu SMTP na ďalší poštový server. Ako sme si už spomínali, tak správa môže prejsť viacerými servermi a teda viacerými MTA. Keď príde na domovský server príjemcu správy, tam sa vloží do jeho inboxu – e-mailovej schránky.

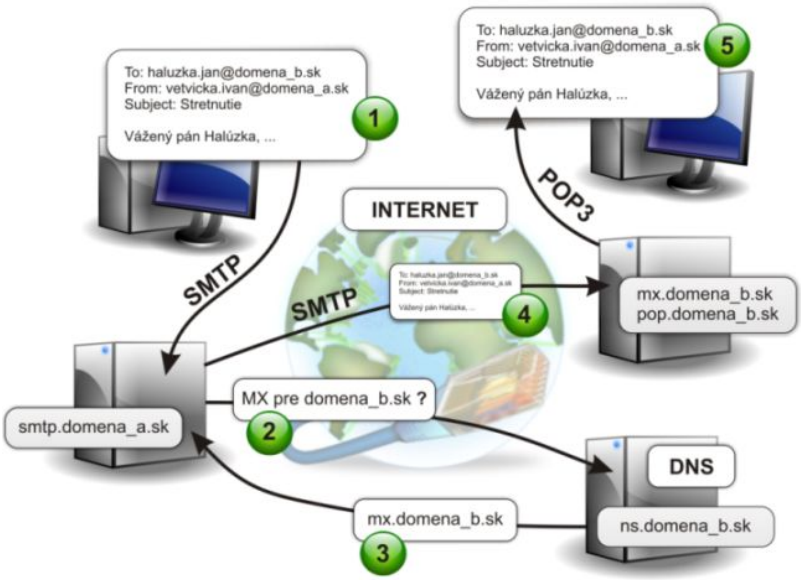
12.2.1 Modely elektronickej pošty

Offline model: klient sa k serveru pripája zriedkavo a vždy načítava všetku poštu. Správy sú pri načítaní zo servera zmazané a ich ďalšie spracovanie vykonáva klient. Najčastejšie sa na načítanie používa protokol POP3 (Post Office Protokol version 3) a pre posielanie SMTP (Simple Mail Transport Protokol).

Online model: vytvorí sa spojenie so serverom. Prácu s poštou riadi klient a zmeny sa vykonávajú priamo na serveri. Používa sa protokol IMAP (Internet Message Access Protocol) a SMTP. Takýto model používajú webmaily ako napríklad GMAIL.

12.2.2 Proces posielania e-mailu

Predtým, ako si povieme viacej o štruktúre správy a spomínaných protokoloch, si ešte ukážme ako sa vlastne pošle naša správa. Proces posielania nám znázorňuje nasledujúci obrázok:



Obrázok 161 Princíp doručenia e-mailu

1) Používateľ Ivan si otvorí svojho e-mailového klienta (MUA) a napíše správu. Potom stlačí tlačidlo „odoslať“ (send), klient automaticky sformátuje správu na správny formát a pošle ju protokolom SMTP domovskému agentovi prenosu pošty (MTA) – smtp.domena_a.sk (je to mail exchange server providera domény).
2) MTA podľa adresy určenej protokolom SMTP požiada systém DNS o adresu poštového servera (mail exchange server) pre domena_b.sk.
3) DNS pre doménu je ns.domena_b.sk, ten nám vráti MX záznam a v našom príklade je exchange server mx.domena_b.sk.
4) MTA smtp.domena_a.sk potom protokolom SMTP pošle správu na ďalší MTA a to mx.domena_b.sk, na ktorom sa správa vloží do inboxu Jána.
5) Používateľ Ján potom vo svojom klientovi stlačením tlačidla „prijať poštu“ (get mail) inicializuje kontrolu novej pošty, jeho MUA prostredníctvom protokolu POP3 stiahne správu, ktorú mu poslal Ivan.

12.2.3 Základné hlavičky správy

E-mail sa skladá z hlavičky a tela, pričom telo môže obsahovať prílohy. Ako celok tvorí jeden textový súbor. Príloha je z binárneho tvaru prekódovaná do textu pomocou BASE-64 kódovania.

V hodnotách hlavičiek môžeme použiť aj špeciálny zápis:

- môžeme použiť čiarku na oddelenie (napr. meno1@domena1, meno2@domena1, meno3@domena2)
- okrúhle zátvorky ( ) uzatvárajú komentár
- použitím znakov < > sa adresa nachádza medzi nimi a reťazec mimo nich sa ignoruje, toto sa často objavuje, keď odpovedáme na e-mail alebo keď zadáme adresu zo zoznamu kontaktov (napr. “Hraško Ján“<jhrasko@domena_a.sk>)
- hranaté zátvorky [ ] určujú IP adresu počítača (mail servera) a preto takýto zápis indikuje, že „názov“ sa nemá prekladať v DNS (napr. meno@[85.41.11.2])

Na nasledovnom obrázku sú zobrazené hlavičky e-mailovej správy.

```
From - Mon Sep 28 21:16:21 2020
X-Account-Key: account1
X-UIDL: 00015de0485d7b1f
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
X-Mozilla-Keys:
Return-Path: <petko1981@gmail.com>
X-Original-To: psvec@ukf.sk
Delivered-To: psvec@ukf.sk
Received: from localhost (localhost [127.0.0.1])
    by posta.ukf.sk (Postfix) with ESMTP id C7E07420247
    for <psvec@ukf.sk>; Mon, 28 Sep 2020 21:16:04 +0200 (CEST)
X-Virus-Scanned: postmaster at posta.ukf.sk
Received: from posta.ukf.sk ([127.0.0.1])
    by localhost (posta1.ukf.sk [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTP id sAey0HRWdMEU for <psvec@ukf.sk>;
    Mon, 28 Sep 2020 21:15:59 +0200 (CEST)
Received: from inmail2.ukf.sk (inmail2.ukf.sk [193.87.12.27])
    (using TLSv1 with cipher ADH-AES256-SHA (256/256 bits))
    (No client certificate requested)
    by posta.ukf.sk (Postfix) with ESMTPS id 52AAE42012C
    for <psvec@ukf.sk>; Mon, 28 Sep 2020 21:15:59 +0200 (CEST)
Received: from localhost (localhost [127.0.0.1])
    by inmail2.ukf.sk (Postfix) with ESMTP id 87B17FC866
    for <psvec@ukf.sk>; Mon, 28 Sep 2020 21:15:58 +0200 (CEST)
X-Virus-Scanned: postmaster at inmail2.ukf.sk
Received: from inmail2.ukf.sk ([127.0.0.1])
    by localhost (inmail2.ukf.sk [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTP id 54UVrICueCmp for <psvec@ukf.sk>;
    Mon, 28 Sep 2020 21:15:47 +0200 (CEST)
Received: from mail-ej1-f46.google.com (mail-ej1-f46.google.com [209.85.218.46])
    by inmail2.ukf.sk (Postfix) with ESMTPS id 06463FC86F
    for <psvec@ukf.sk>; Mon, 28 Sep 2020 21:15:04 +0200 (CEST)
Received: by mail-ej1-f46.google.com with SMTP id o8so10392869ejb.10
        for <psvec@ukf.sk>; Mon, 28 Sep 2020 12:15:04 -0700 (PDT)
```

Obrázok 162 Úplná hlavička e-mailovej správy

Recieved – hlavičiek recieved môže byť v záhlaví niekoľko. Vždy ju na začiatok pripisuje poštový server, ktorým správa prechádza. To znamená, že najvyššie je zobrazený je posledný server a teda zhora nadol vidíme spätnú cestu k odosielateľovi. V hodnote hlavičky sa vyskytujú riadky, ktoré začínajú slovami **FROM** (odkiaľ správa prišla), **BY** (počítač, ktorý správu prijal), **VIA** (určuje fyzickú adresu), **WITH** (určenie sieťového alebo poštového protokolu), **ID** (identifikácia správy) a **FOR** (pre koho je správa určená).

From – hlavička určujúca od koho bola správa poslaná.

Sender – ak sa v hlavičke from nachádza viacej adries (oddelených čiarkou), tak potom táto hlavička obsahuje jednu adresu a určuje odosielateľa.

Date – určuje dátum odoslania (deň, dátum, čas a časová zóna).

Reply-To – určuje, na akú adresu sa má odoslať odpoveď.

In-Reply-To – odpoveď na správu: identifikácia správy

To – adresa príjemcu správy (adresy príjemcov správy, oddelené čiarkami).

Cc – určenie adries pre zaslanie kópie správy (skratka Cc je z Carbon copy).

Bcc – podobné ako Cc, ale ostatní príjemcovia tieto adresy nevidia (Blind carbon copy).

Message-Id – identifikácia správy.

Keywords – popis, kľúčové slová charakterizujúce obsah. Takýchto hlavičiek môže byť viacej.

References – odkaz na správu, obsahuje identifikáciu správy.

Subject – predmet správy, krátky popis.

Comments – komentáre, môže byť viackrát.

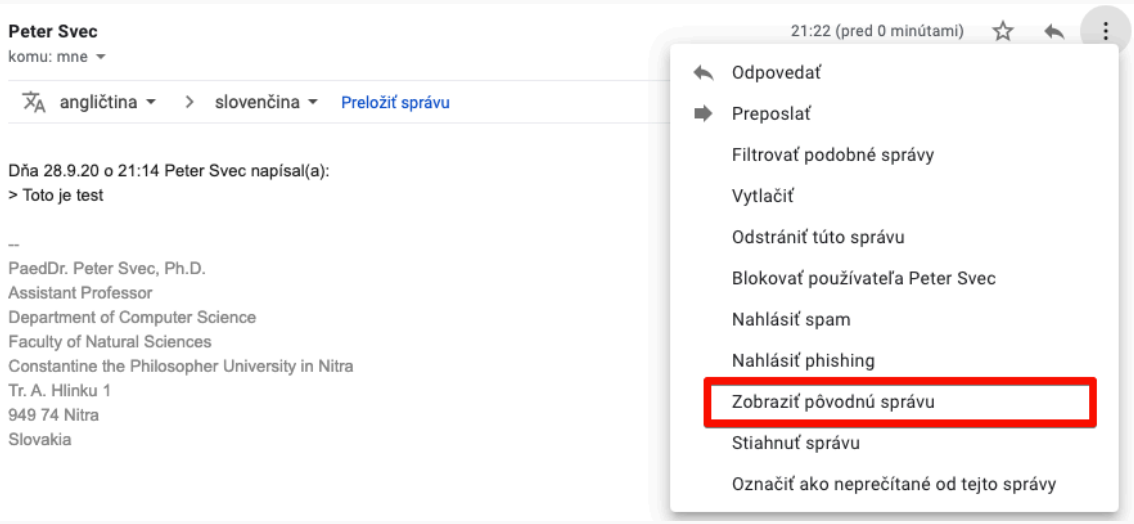
X- (každá používateľsky definovaná hlavička musí mať predponu X-, najčastejšie sa stretávame s hlavičkou X-Mailer, ktorá určuje verziu a typ aplikácie na odosielanie pošty)

Resent- (používa sa pri preposielaní správy, pred pôvodne hlavičky sa pridá Resent-)

ÚLOHA – RIEŠTE!

Zo svojho e-mailu zašlite správu spolužiakovi, preskúmajte a analyzujte plnú hlavičku správ. Zamerajte sa na cestu, ktorú email prešiel.

Ak používate gmail, k úplným hlavičkám sa dostanete nasledovne:



Diskutujte o tom čo je SPAM a Phishing.

ZHRNUTIE

Čo sme sa naučili

- Systém DNS je základnou a neoddeliteľnou súčasťou služieb Internetu a bez neho by sme nevedeli využívať služby Internetu.
- Doménové mená delíme na generické a kódy krajín.
- Údaje o vlastníkoch domén si vieme pozrieť v whois databáze.
- Elektronická pošta sa prenáša ako textový súbor cez smtp protokol.
- Používatelia pristupujú k pošte cez protokoly pop3 alebo imap, ktorý používa aj napríklad webmail (gmail a pod.).
- V zobrazení pôvodnej správy môžeme vidieť presnú trasu, ktorú email prešiel.

13 PROTOKOL IPV6

CIEĽ

Cieľom tejto kapitoly je získať základné informácie o „novom“ protokole v Internete. Je ním protokol IPv6 a hoci je tu už niekoľko desiatok rokov, jeho používanie je stále na nízkej úrovni. Naučíte sa zapísať IPv6 adresu, poznať pravidlá delenia IPv6 adresného priestoru a nakonfigurovať lokálnu sieť s IPv6 adresáciou.

VÝKLAD

Počiatky protokolu IPv6 siahajú do začiatku deväťdesiatych rokov, kedy sa Internet začal rozširovať v oblasti obchodu a priemyslu. Už v tom období však bolo zrejmé, že adresný priestor protokolu IPv4 nevydrží budúcu expanziu internetu, ktorého štúdie na najbližšie obdobie naznačovali, že k jeho výraznému nedostatku dôjde do najbližších desiatich rokov. V roku 1992 bol globálny nedostatok IPv4 adries uznaný ako vážny obmedzujúci faktor budúceho rozvoja internetu. Adresný priestor IPv4 predstavuje 4 miliardy adries, no veľká časť z nich nie je použiteľná pre koncové zariadenia. Ako reakciu na tento problém začala v roku 1994 inžinierska pracovná skupina IETF návrh a vývoj sady princípov a noriem nového protokolu, ktorý by tento problém vyriešil. Zároveň však mali výskumníci dostatok času a priestoru na implementáciu nových vlastností, ktoré by do nového protokolu začlenili. V decembri 1995 už bolo špecifikované jadro IPv6, priemysel však nebol pripravený prijať nový protokol, pretože jednak nebol kompatibilný s aktuálne používaným protokolom IPv4 a zároveň by prechod na nový protokol nezaručil zisky, ktoré boli v prípade protokolu IPv4 v tom období zaistené.

Spomaliť vyčerpanie adresného priestoru IPv4 sa podarilo vyčlenením privátnych IP adries a spustením technológie NAT (Network Address Translation). Tento mechanizmus skrýva celú lokálnu sieť s privátnymi adresami pod jednu verejnú adresu. Je to rovnaký mechanizmus ako máme doma. Domáci smerovač má od poskytovateľa pridelenú jednu verejnú adresu a doma máme mnoho zariadení, z ktorých ma každé svoji vnútornú adresu, no navonok vystupujú len s adresou smerovača. Týmto spôsobom šetríme verejné adresy. Technológia NAT má mnoho obmedzení, no pre väčšinu malých spoločností a domácnosti nie sú nijak zásadné.

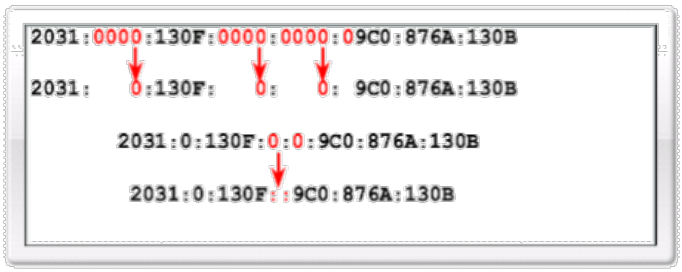
Postupne sa vývoj IPv6 posúva tak, že v roku 2015 sa stáva súčasťou Linuxov a v roku 2016 súčasťou Windowsov. Dnešné operačné systémy majú predvolenú podporu pre IPv6, systém DNS ho podporuje rovnako a veľký hráči na trhu (google, amazon, facebook) sú podporovateľmi nového protokolu.

Nový protokol však nie je len o veľkom adresnom priestore, no tvorcovia sa pokúsili odstrániť mnohé nedostatky IPv4 protokolu.

13.1 IPv6 adresa

Adresy IPv6 sú dlhé 128 bitov, čo nám dáva približne 3,4 x 10^{38} unikátnych adries (2^{128}). Oproti protokolu IPv4 sa zapisujú v hexadecimálnom formáte a ako oddeľovač sa namiesto bodky

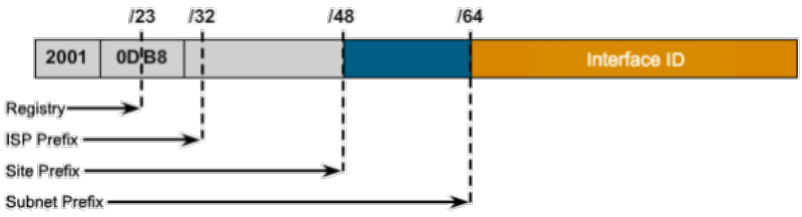
používa dvojbodka. Adresa je rozdelená na osem blokov po štyroch číslicach. V špeciálnych prípadoch si však zápis môžeme zjednodušiť (skrátiť).



Obrázok 163 Zápis IPv6 adresy

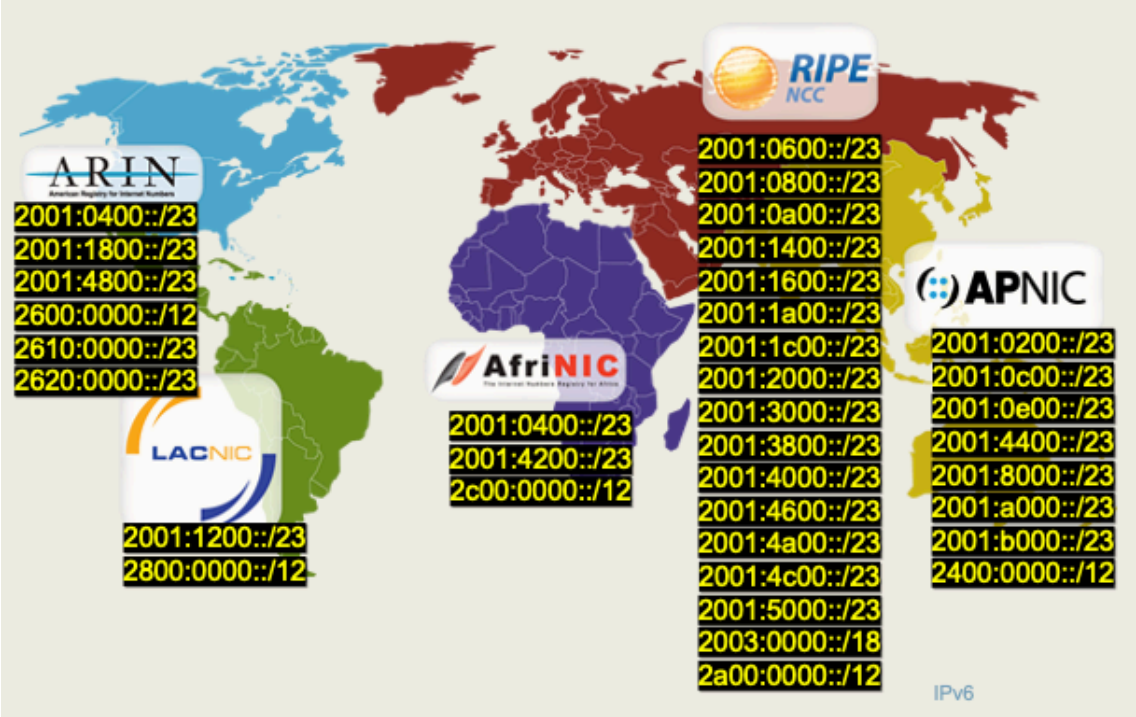
- V každom bloku môžeme vynechať počiatočné nuly.
- Ak blok obsahuje len nuly (0000), môžeme ich nahradiť jednou nulou (0).
- Ak sú vedľa seba bloky s nulami (0000:0000:0000), môžeme ich úplne vynechať (::), pričom takého vynechanie je možné spraviť len raz a vynechávame najväčšiu sekvenciu núl.

Istým negatívom IPv4 adresného priestoru je jeho roztrúsenosť. Podľa IPv4 adresy len veľmi ťažko určíme, do ktorej časti sveta adresa patrí. Zaradenie do regiónu je pritom dôležité pri optimalizácii trasy, ktorou prechádzajú dáta v Internete. Tvorcovia IPv6 si však na poriadku v adresách dali záležať, a každá časť IPv6 adresy má svoj hierarchický význam. Časť IPv6 adresy vyčleňujeme **prefixom** a zapisujeme lomkou s číslom. Prefix určuje počet bitov od začiatku adresy.



Obrázok 164 Hierarchia v IPv6 adrese (http://network.jecool.net/ipv6-address-types/)

- Prvých 23 bitov IPv6 adresy označuje regionálny register. Svet je rozdelený rovnako ako pri IPv4 do piatich regiónov.
- Časť adresy od /23 do /32 označuje poskytovateľa služieb internetu (ISP).
- Časť od /32 do /48 označuje už samotnú sieť koncového zákazníka (domácnosť, firma).
- Koncový zákazník si ešte môže u seba vytvoriť menšie navzájom oddelené siete. Ich počet je od prefixu /48 po /64, čiže 2^{16} = 65536.

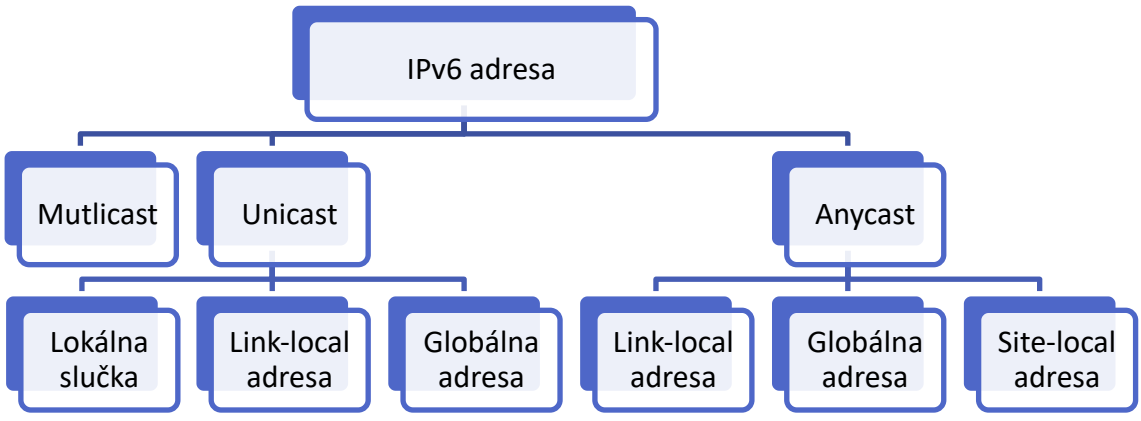


Obrázok 165 Rozdelenie IPv6 adries po regiónoch

Pri konfigurácii IPv6 nie je nutné nastavovať aj IPv4 protokol. Ak by sme však nastavili aj IPv4 adresy, nič sa nedeje. Oba protokoly vedia vedľa seba existovať.

13.2 Druhy IPv6 adries

Podobne ako v IPv4 aj v IPv6 rozdeľujme adresy do niekoľkých skupín v závislosti od spôsobu komunikácie.

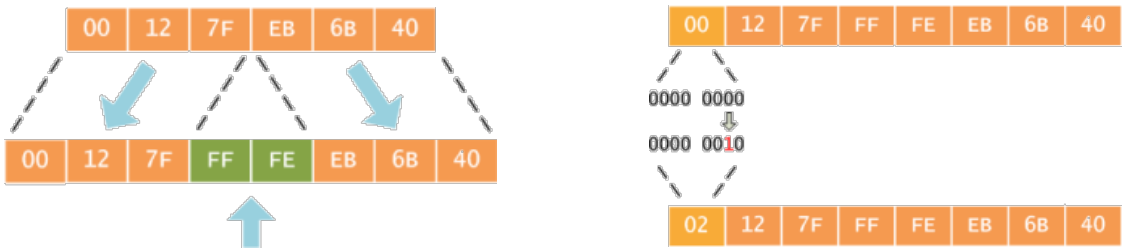


Obrázok 166 Skupiny adries v IPv6

Tak ako v IPv4 existuje špeciálna adresa pre localhost, existuje aj v IPv6. Jej adresa je **::1.** Unicast aj Anycast adresy majú globálnu aj lokálnu verziu. Lokálna slúži len na komunikáciu v jednej lokálnej sieti. Lokálna adresa začína FE80.

13.3 Lokálna adresa (link-local)

Zariadenia v IPv6 si dokážu vytvoriť lokálnu adresu automaticky. Vygenerujú si ju zo svojej MAC adresy vytvorením EUI-64 identifikátora.



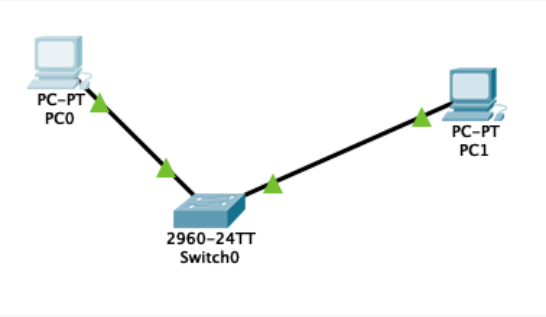
Obrázok 167 Vytvorenie EUI-64 adresy

Postup pre vytvorenie EUI-64 identifikátora je nasledovný:

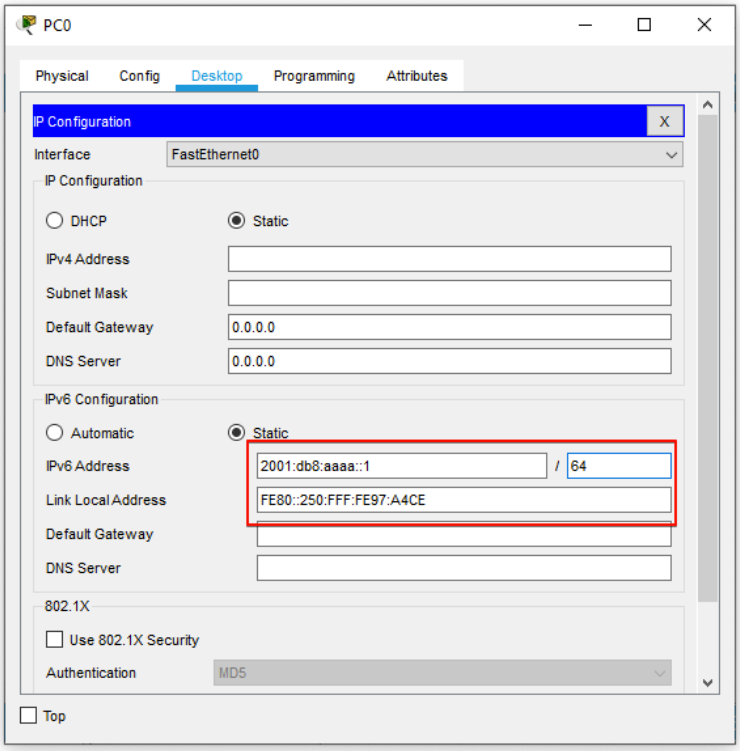
- MAC adresa zariadenia sa rozdelí na polovicu a do stredu sa vloží FFFE
- Invertujeme 7. bit zľava

ÚLOHA – RIEŠTE!

Vytvorte sieť pozostávajúcu z dvoch počítačov a jedného prepínača. Počítačom prideľte nasledovné adresy *(link-local adresa je predvyplnená a nemeníme ju).*



- PC0 – 2001:db8:aaaa::3/64
- PC1 – 2001:db8:aaaa::2/64

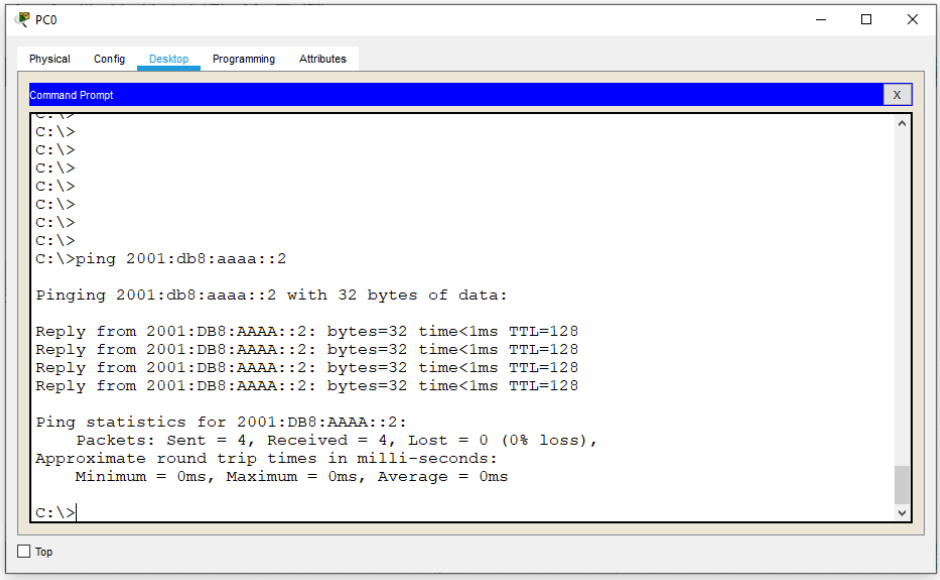


Obrázok 168 Konfigurácia IPv6 adresy

Poznámka pre učiteľa:

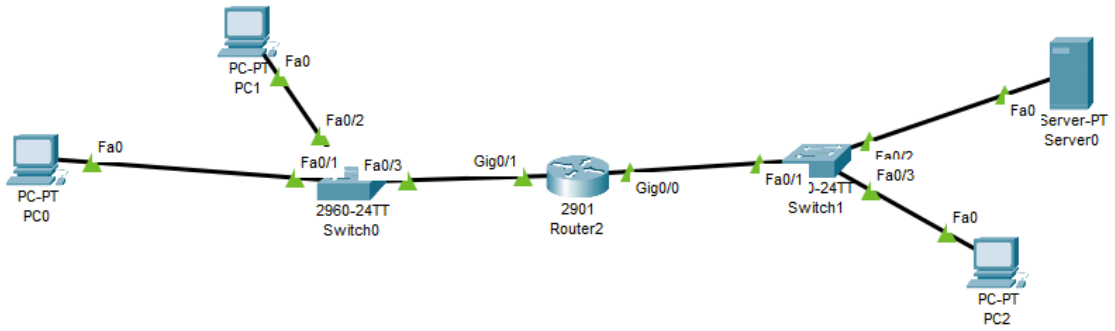
V protokole IPv6 má každé zariadenie vždy aj link-local adresu začínajúcu na FE80. Pomocou príkazu **ipconfig /all** zobrazte MAC adresu zariadenia a overte, že link-local adresa je odvodená od MAC adresy.

Správnosť konfigurácie si overíme príkazom **ipconfig**. Spojenie medzi počítačmi overíme pomocou príkazu **ping**.



Obrázok 169 Overenie dostupnosti príkazom ping

Výhodnou vlastnosťou IPv6 je schopnosť autokonfigurácie. Znamená to, že zariadenia si vedia samé získať množstvo údajov, ako napríklad adresu smerovača (ktorý oznamuje do siete svoju existenciu). Táto konfigurácie je náročnejšia, avšak zvládnuteľná. Topológia siete bude nasledovná, no do siete treba pridať smerovač 2901 (ten má aj podporu IPv6).



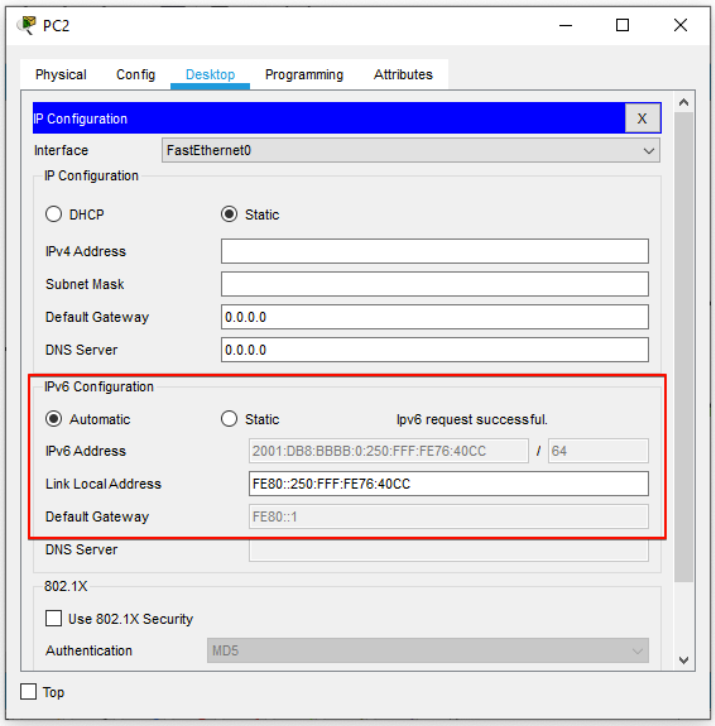
Obrázok 170 Sieť s pridaným smerovačom 2901

IPv6 adresácia novo pridanej časti siete bude nasledovná:

- Smerovač na rozhraní Gig0/0 bude mať adresu 2001:db8:bbbb::1/64
- Smerovač na rozhraní Gig0/0 bude mať link-local adresu fe80::1
- Smerovač na rozhraní Gig0/1 bude mať adresu 2001:db8:aaaa::1/64
- Smerovač na rozhraní Gig0/1 bude mať link-local adresu fe80::1
- Server bude mať adresu ručne zadanú 2001:db8:bbbb::9999/64
- PC0, PC1 a PC2 budú mať IPv6 automaticky nakonfigurovanú

Konfigurácia smerovača pre IPv6 je možná len v príkazovom riadku. Po kliknutí na smerovač prejdeme na záložku CLI. Cisco smerovač rozlišujú dva režimy práce – neprivilegovaný označený v prompte ako **>** a privilegovaný označený ako **#**. Do konfigurácie sa dostaneme stlačením klávesy ENTER (RETURN) a zadaní **enable** (privilegovaný režim)

- enable
- configure terminal
- ipv6 unicast-routing
- interface gig0/0
- ipv6 address 2001:db8:bbbb::1/64
- ipv6 address fe80::1 link-local
- no shutdown
- interface gig0/1
- ipv6 address 2001:db8:aaaa::1/64
- ipv6 address fe80::1 link-local
- no shutdown



Obrázok 171 Automatická konfigurácia IPv6 na PC2

Poznámka pre učiteľa:

Niekedy sa stane, že sa smerovač spýta na inicializačný konfiguračný dialóg. Na túto otázku odpovedáme **NO**. Ak by sa žiak nachádza v nejakej inej časti konfigurácie ako úvodnej, po stlačení kombinácie CTRL+Z sa vráti na začiatok.

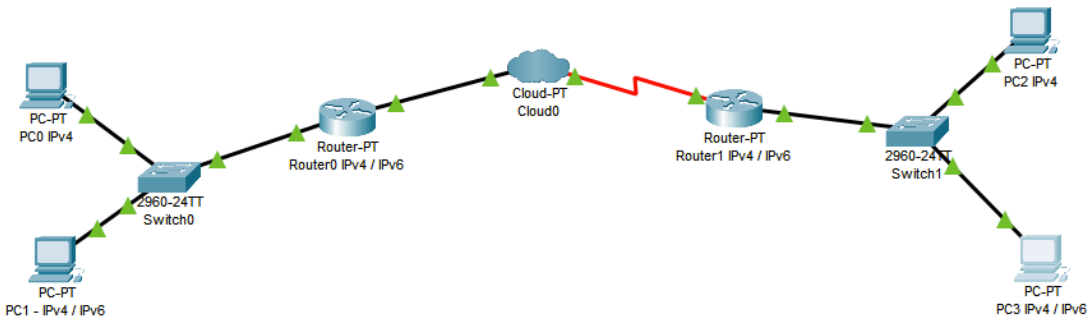
13.4 Prechod medzi IPv4 a IPv6

Prechod medzi dnešným IPv4 svetom a IPv6 svetom bude dlhodobý proces. Mnohí poskytovatelia služieb internetu neponúkajú nový protokol, používatelia ho od nich nevyžadujú. Zmenu možno prinesie čoraz väčšie používanie zariadení internetu vecí a nástup 5G mobilnej siete.

Oba IP svety môžeme prepojiť rôznymi spôsobmi.

- Dual Stack
- Tunelovanie

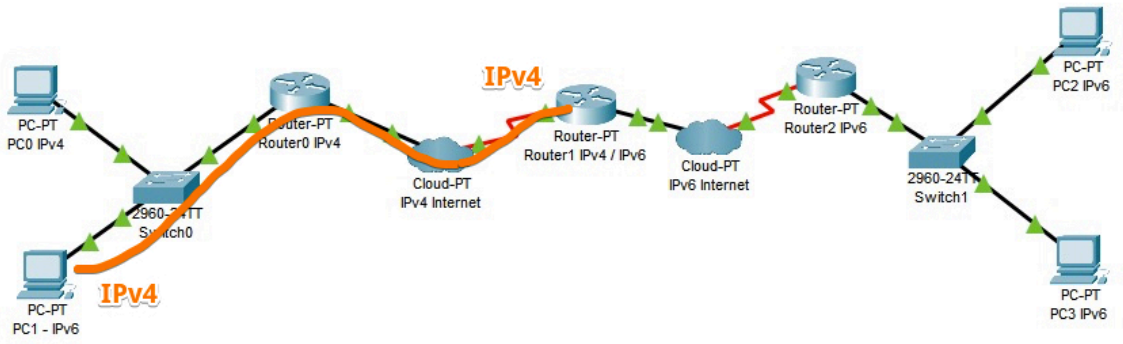
Pri Dual Stack máme k dispozícii smerovač, ktorý rozumie IPv4 aj IPv6 a má tak prístup do oboch častí siete. Z pohľadu koncového používateľa je tento spôsob najideálnejší, stačí ak aj jeho operačný systém podporuje IPv6 a má pridelenú od poskytovateľa aj IPv6 adresu, no zároveň má aj IPv4 adresu.



Obrázok 172 Princíp Dual Stack

V prípade, že náš poskytovateľ internetu nový protokol nepodporuje a my do IPv6 siete prístup chceme, musíme nájsť v Internete vstupný bod, cez ktorý sa tam dostaneme. Spojenie medzi naším počítačom a týmto vstupným bodom nazývame **tunel**.

Základná myšlienka tunelovania je zabalenie jedného protokolu do druhého. To nám umožňuje preniesť dáta cez sieť, ktorá daný protokol nepodporuje.



Obrázok 173 Princíp tunelovania

Každý tunel má 2 konce, ktoré majú svoju IPv4 adresu. IPv6 údaje sú vložené do IPv4 odoslané tunelom. Keď dorazia na druhý koniec tunelu zariadenie vybalí pôvodné IPv6 dáta a odošle ho do cieľovej adresy. Výhodou aj nevýhodou zároveň je, že údaje vnútri v tuneli sú skryté pred správcom siete.

13.4.1 Tunnel Broker a Tunnel Server

Aby sme mohli používať IPv6 tunel, potrebujeme poskytovateľa tunelu (Tunnel Broker) a potrebujeme tiež verejnú IP adresu. V súčasnosti je spoločností poskytujúcich tunel len veľmi málo. Jednou takou je Hurricane Electirc (https://tunnelbroker.net) alebo NetAssist (http://tb.netassist.ua)

Tunnel broker je miesto kde sa používateľ registruje a aktivuje tunel. Tunel broker poskytuje vytvorenie tunelu, modifikácie a zmazanie na želanie používateľa.

Tunnel server je dual-stack (IPv4 a IPv6) smerovač. Tunnel server vytvára a konfiguruje požiadavky od tunel brokera. Taktiež si udržiava štatistiky o všetkých aktivovaných tuneloch. Aktivovaný tunel spotrebúva pamäť serveru a spracovateľský čas.

ÚLOHA – RIEŠTE!

Vyberte si tunnel brokera a pokúste sa vytvoriť doma tunel k sieti IPv6. V škole vám to takmer s istotou nepôjde. Potrebujete verejnú IP adresu.

ZHRNUTIE

Čo sme sa naučili

- V IPv6 používame 128 bitovú adresu. Jej štruktúra je prísne hierarchická.
- Protokol predstavuje návrat k end-to-end princípu.
- Každé zariadenie má link local adresu začínajúcu na FE80.
- V IPv6 neexistuje broadcast, je nahradený multicastom.

14 ANONYMITA NA WEBE A NEVIDITEĽNÝ WEB

CIEĽ

Cieľom tejto kapitoly je upozorniť na riziká komunikácie v digitálnom priestore. Každá činnosť, ktorú vykonávame zanecháva po nás stopu. V tejto kapitole si ukážeme, čo všetko o sebe prezrádzame a ako sa môžeme stať na Internete neviditeľnými. V žiadnom prípade, táto kapitola nie je návod na „zlé“ veci, no musíme ich poznať aby sme im nepodľahli.

VÝKLAD

Premýšľali ste niekedy nad tým, čo sa stane keď klikneme na webový odkaz? Operačný systém pripraví údaje na odoslanie a zašle ich na

- domáci smerovač
- smerovač poskytovateľa internetu
- peeringové centrum (miesto kde sa prepájajú poskytovatelia)
- chrbticová sieť Internetu
- peeringové centrum (miesto kde sa prepájajú poskytovatelia)
- smerovač spoločnosti, kde je uložený webový server
- webový server.

Keď server pripraví odpoveď, údaje sa prenesú rovnakým spôsobom naspať. Cestu údajov si môžeme overiť pre nás už známym príkazom **tracert**.

traceroute www.theguardian.com

traceroute to guardian.map.fastly.net (151.101.37.111), 64 hops max, 52 byte packets

1 gw.ukf.sk (194.160.208.1) 1.255 ms 0.368 ms 0.407 ms

2 wgw-fo-c6k5.nr.sanet.sk (193.87.99.1) 0.716 ms 0.670 ms 0.548 ms

3 cvt-bratislava.sanet2.sk (194.160.8.1) 1.967 ms 2.077 ms 3.226 ms

4 sanet-ias-geant-gw.bra.sk.geant.net (83.97.88.237) 8.686 ms 2.272 ms 2.081 ms

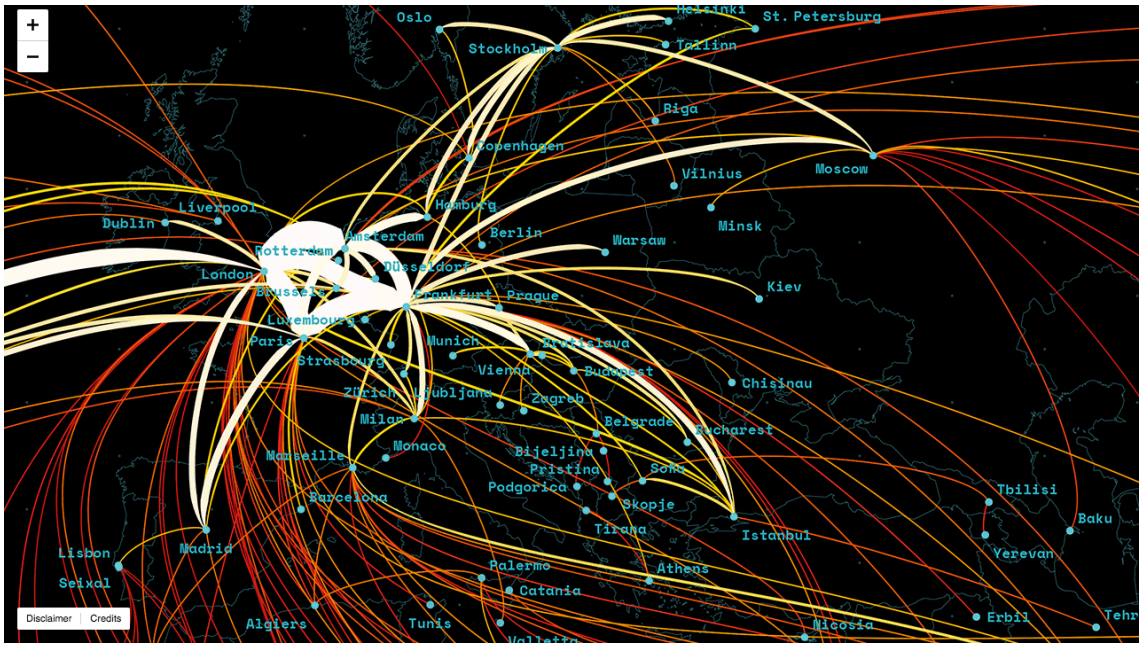
*5 * * **

6 ae6.2.mx1.fra.de.geant.net (62.40.98.96) 25.156 ms 22.158 ms 15.494 ms

7 ae1.mx1.ams.nl.geant.net (62.40.98.129) 21.901 ms 22.655 ms 21.929 ms

Obrázok 174 Graficky zobrazená trasa údajov cez Internet

Ak by sme spravili príkaz **tracert** na každý počítač v Internete, získali by sme veľmi presnú mapu Internetu. Ak keď vzhľadom na veľkosť adresného priestoru IPv4 (4 miliardy adries) znie táto úloha nereálne, mnoho jednotlivcov aj organizácií generuje rôzne mapy Internetu.



Obrázok 175 Mapa Internetu (prepojenie medzi peeringovými centrami)

Niekoľko máp môžeme nájsť aj na nasledovných odkazoch:

- http://som.csudh.edu/fac/lpress/history/arpamaps/
- https://www.submarinecablemap.com
- https://www.internetexchangemap.com
- http://global-internet-map-2017.telegeography.com
- http://submarine-cable-map-2018.telegeography.com

Pri prenose údajov medzi dvoma zariadeniami je nevyhnutnou súčasťou prenášaných údajov IP adresa odosielateľa a IP adresa príjemcu. Adresa odosielateľa môže mať pre príjemcu aj väčší úžitok ako len poznačiť si, kam má údaje naspäť poslať.

Popri databázach IP adries, ktoré sme si už spomínali a boli v nich uložené údaje o vlastníkoch, existuje aj mnoho iných. Sú nimi napríklad čierne listiny odosielateľov spamu (RBL – Realtime Black List) alebo geolokačné databázy, ktoré nám zobrazia napríklad mesto, krajinu alebo poskytovateľ služieb Internetu.

Obrázok 176 Geolokačné údaje z IP adresy

ÚLOHA

Pomocou stránok https://whatismyipaddress.com/ alebo https://www.whatismyip.com zistite informácie o svojej IP adrese.

Programátor webovej stránky tak presne vie, odkiaľ sa na jeho web pripájate a podľa toho vám môže zobrazovať napríklad reklamu, alebo sa vás môže snažiť oklamať alebo nanútiť vám produkt alebo iný web. Typické sú hlásenia typu *„Pozor akcia: Ak si z okolia Nitry KLIKNI SEM a môžeš vyhrať nový iPhone. Akcia je len pre používateľov z okolia Nitry.“*. Väčšinou nasleduje registrácia na pochybnom portáli.

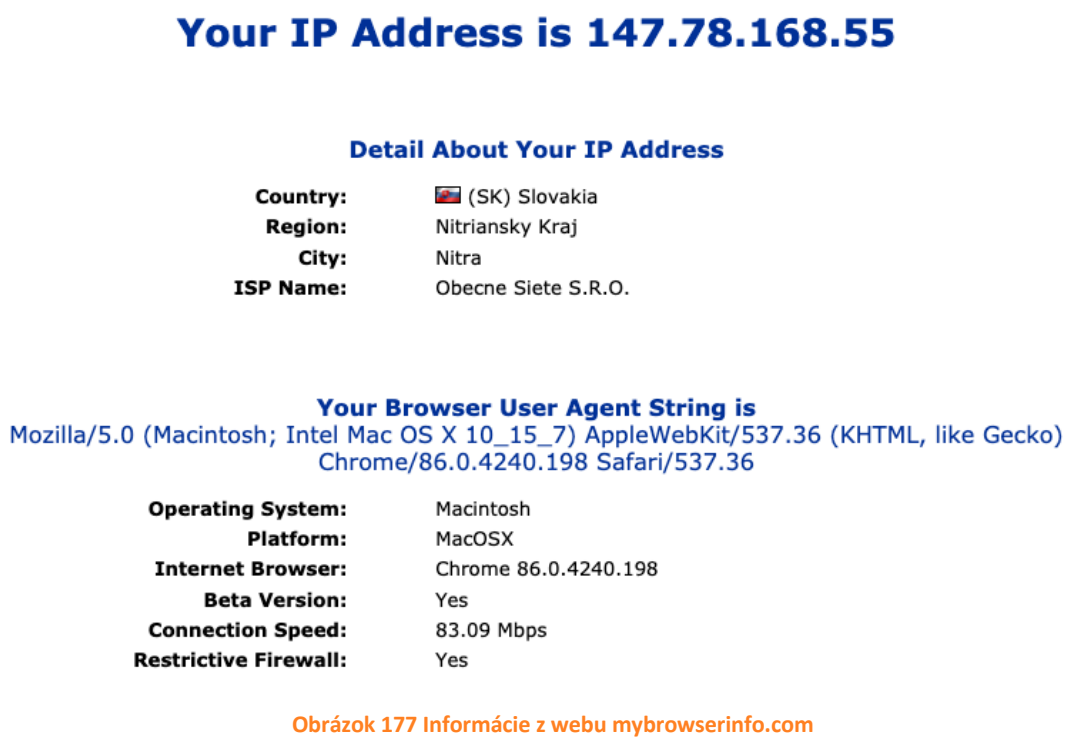
Informácia o meste alebo krajine sa používa aj pri filtrovaní obsahu návštevníkom. Ľudom z jednej krajiny obsah zobrazíme, iným nie. Typickým príkladom sú športové prenosy, kedy licenčné podmienky neumožňujú aby obsah sledovali ľudia mimo krajiny, na ktorú bola vydaná licencia.

Okrem IP adresy prezrádzame návštevou webu omnoho viac.

ÚLOHA

Otvorte web http://mybrowserinfo.com/ a prezrite si zobrazené informácie. Kliknite aj na odkaz **See Detailed Location and Browser Information**

Prehliadač posiela alebo webová stránka si môže vyžiadať informácie o verzii prehliadača, operačnom systéme, rozlíšení a mnoho ďalšieho.



Obrázok 177 Informácie z webu mybrowserinfo.com

Primárny účel týchto informácií je pre vývojára, aby vedel nastaviť zobrazenie webu presne na naše zariadenie (podľa rozlíšenia, typu zariadenia), avšak vynaliezaví vedia tieto informácie zneužiť. Zozbieraním týchto údajov môžeme ľudí „vydierať“, že vieme odkiaľ sa pripájajú, aký majú počítač, akú obrazovku a keď nepošlú bitcoiny tak im zablokujeme počítač.

Už sme si spomenuli, že aj poslaním emailu prezrádzame svoju polohu. Pri zobrazení pôvodnej správy vidíme hlavičky Received a príjemca presne vie, z akého počítača alebo siete bol email poslaný. Vymyslieť si falošnú adresu a poslať email o bombe v škole kvôli písomke a spraviť to z domu nie je práve najlepší nápad. Stačí sa spýtať poskytovateľa, ktorý zákazník mal v čase odoslania pridelenú danú IP adresu.

- **Received: from MacBook-Pro.local (unknown [213.215.118.197])** (using TLSv1 with cipher DHE-RSA-AES128-SHA (128/128 bits)) (No client certificate requested) by posta.ukf.sk (Postfix) with ESMTP id 841D71E7508 for <psvec@teacher.sk>; Sun, 27 May 2020 21:17:29 +0200 (CEST)
- User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:52.0) Gecko/20100101 Thunderbird/52.8.0
- Message-ID: 8ea6d072-ba81-9d72-a24c-038751ab67c4@ukf.sk

Pozor by sme si mali dať aj na to, aké fotky publikujeme na webe. Fotoaparáty aj telefóny vkladajú do jpg súboru informácie o nastavení objektívu, clone, blesku, rozlíšení čipu, GPS koordináty (EXIF formát).

ÚLOHA

Zistite, kde bola vytvorená nasledovná fotografia.

- Dátum a čas (16. 6. 2014, 16:09)
- Nastavenie fotoaparátu (Sony DSC-F828, 3263x2176, F4, e 1/60)
- GPS súradnice (-16.173851, 145.441734)

Publikovaním fotky s GPS prezrádzame kde bývame, či kde sa nachádzame. Pre publikovaním je viac ako vhodné EXIF hlavičku z jpg súboru odstrániť službami ako napríklad https://www.verexif.com/en/ alebo https://www.exifremove.com/ alebo http://www.exifpurge.com/ . Nie je vhodné spoliehať sa na to, že sociálna sieť tieto info odstráni. Možno takú funkciu aj ponúka, no predtým si ich dobre odloží.

14.1 Bezpečná registrácia na „pochybnom“ webe

Každý sa už dostal do situácie, keď potreboval vyplniť registračný formulár niekde na webe, no nechcel tam vyplniť svoju mailovú adresu alebo telefónne číslo. Jedná sa o rôzne „diskusné fóra“, súťaže, ankety, hlasovania a podobne. Pri registrácii si musíme vymyslieť prihlasovacie meno a heslo, zadať osobné údaje (meno, priezvisko, adresu, ...), telefónne číslo a funkčný e-mail (opt-in). Osobné údaje si vymyslíme ľahko, no vytvárať si nový email sa nie každému chce. Pomoc poskytuje 10 minútový email. Ten je určený len na doručovanie správ, dá sa zriadiť bez registrácie a je platný len niekoľko minút.

- https://10minutemail.com
- https://temp-mail.org/
- https://www.fakemail.net/

ÚLOHA

Vytvorte si dočasný email a pošlite si tam správu z vášho súkromného gmailu.

Na niektorých portáloch je nutné zadať aj telefónne číslo, na ktoré príde autorizačná SMS správa. Zadať svoje číslo je už väčší zásah do súkromia ako email. Podobne ako dočasné emaily existujú aj služby s dočasnými telefónnymi číslami. Prezrite si nasledovné weby a ak chcete, môžete na niektoré z tých čísiel poslať SMS správu.

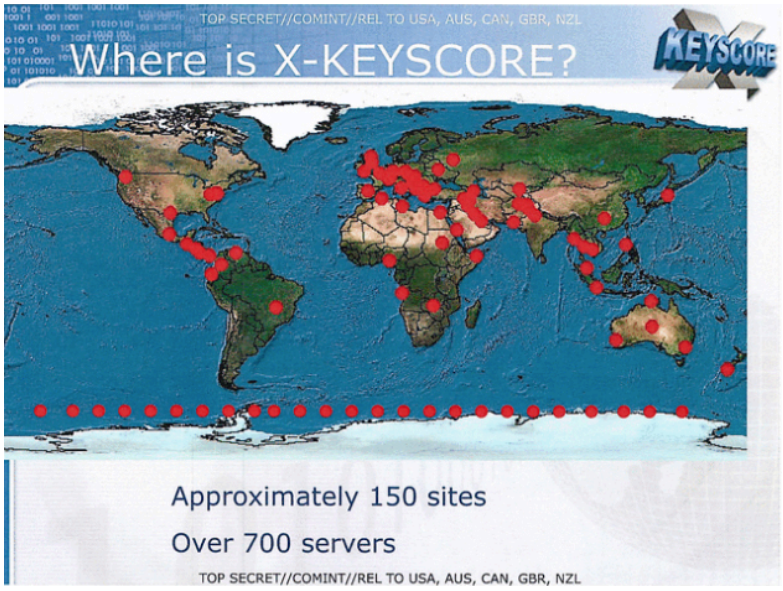
- https://receive-sms-online.com/
- https://smsreceivefree.com/

14.2 Odpočúvanie

Každé zariadenie, cez ktoré prechádzajú údaje je potencionálny zdroj odpočúvania komunikácie. To čo robíte na Internete vidí v prvom rade váš poskytovateľ ako aj prevádzkovateľ peeringového centra. Odpočúvanie je možné len za prísnych zákonom stanovených podmienok.

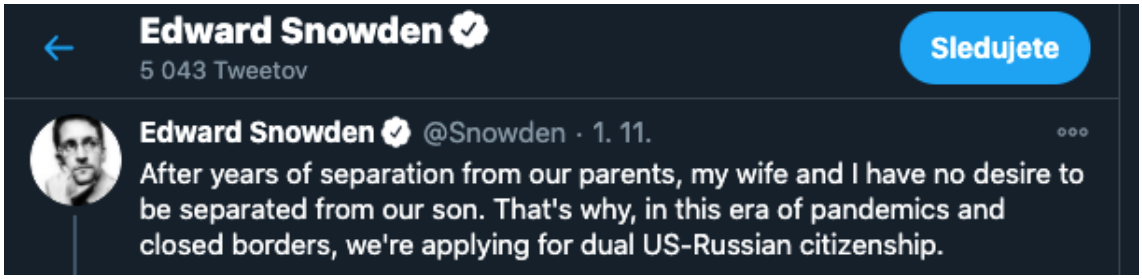
Bohužiaľ, mnohé aj vládne organizácie to robia aj bez zákonných dôvodov. O praktikách z odpočúvania prehovorili informátori – Whistle Blowers. Medzi známe mená patrí Julian Assange, ktorý v roku 2006 vytvoril pre novinárov platformu Wikileaks na nezávislé publikovanie bez zásahu redakcie alebo štátnych orgánov. V roku 2010 platformu Wikileaks využila americká vojačka Chelsea Elizabeth Manning (narodená ako Bradley Edward Manning - zmena pohlavia vo väzení) aby publikovala tajné diplomatické správy diplomatov USA. Dňa 25. mája 2010 bola zadržaná a 21. augusta 2013 odsúdená za špionáž, krádež vládneho majetku a jeho publikovanie na wikileaks na 35 rokov vo väzení. Obama jej znížil trest na 7 rokov, a prepustená bola 17. mája 2017.

Ešte známejší je americký Edward Snowden, ktorý v roku 2013 cez noviny The Guardian publikoval informácie o tajnom odpočúvaní agentúrami NSA a Britain's GCHQ. Tajné programy FISA, PRISM, Tempora, XKeyscore dokázali napichnúť optické káble po celom svete a sledovať tak svetových lídrov a cudzie krajiny.



Obrázok 178 Umiestnenie odpočúvacích bodov programu X-Keyscore

Snowden ušiel v roku 2013 do Ruska a 1.11.2020 požiadal o ruské občianstvo.



Obrázok 179 Snowdenow tweet z 1.11.2020

Napichnutie optického kábla je veľmi ťažké odhaliť. Na kopírovanie sieťovej komunikácie stačí narušiť vonkajšiu izoláciu aby stačí aby presvitalo len 1% svetla. Ak útočník pozná použitú technológiu, jednoduchšie trafí amplitúdu, fázu a polarizáciu.

ÚLOHA

Preštudujte si nasledovné odkazy:

- https://www.nytimes.com/2005/02/20/politics/new-nuclear-sub-is-said-to-have-special-eavesdropping-ability.html
- https://www.theatlantic.com/international/archive/2013/07/the-creepy-long-standing-practice-of-undersea-cable-tapping/277855/
- http://www.theregister.co.uk/2014/06/03/revealed_beyond_top_secret_british_intelligence_middleeast_internet_spy_base/
- http://www.thefoa.org/tech/ref/appln/tap-fiber.html
- https://www.joshruppe.com/fiber-optic-tapping-tapping-setup/
- https://youtu.be/X-hAtc-ku-Y

Spoločnosť FinFisher vyvinula zariadenie, ktoré je schopné vložiť sledovací malvér do bežnej aplikácie (Whatsapp, Skype, VLC, ...) a pri aktualizácii si ho sledovaný cieľ nainštaluje. Podľa wikileak dané zariadenie má aj Slovensko (https://wikileaks.org/spyfiles4/customers.html#customer_15). Špehovanie odhalila v roku 2017 spoločnosť ESET (https://androidportal.zoznam.sk/2017/09/eset-odhalil-spehovaciu-kampan/).

14.3 Ako sa skryť

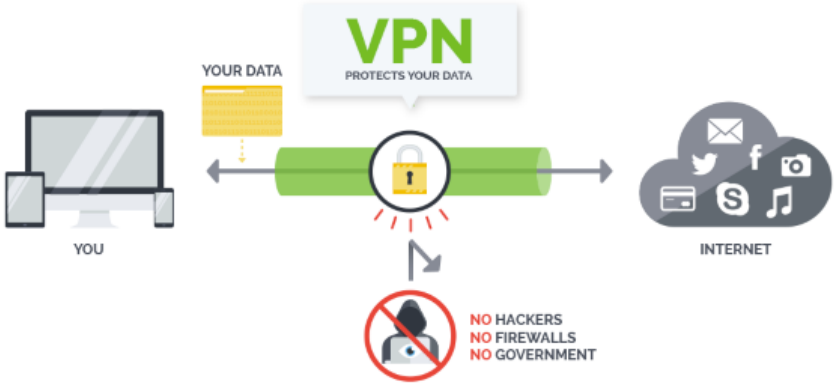
Na problém s odpočúvaním zareagoval Google tak, že pri vyhľadávaní začal dávať vyššiu priečku webov, ktoré šifrovali komunikáciu cez protokol SSL. V roku 2013 totiž šifrovane boli prenášané len prihlasovacie údaje a zvyšok stránky nie. Šifrovanie celého obsahu predstavuje značné nároky na výpočtový výkon servera aj klienta a certifikáty na šifrovanie niečo stoja. V roku 2016 vznikla certifikačná autorita Let's Encrypt, ktorá zdarma vydáva certifikáty platné 90 dní. Tento krok umožnil, aby dnes drvivá väčšina webov podporovala šifrovaný prenos medzi koncovým počítačom a serverom. Spoločnosť, ktorá odpočúva síce získa údaje, avšak nevie ich prečítať.

Podobe zareagovali aj poskytovatelia emailov a komunikácie medzi poštovými servermi a klientami je povinne šifrovaná. Šifrovanú máme aj komunikáciu vo Viber, Whatsapp a pod.

Keď používame šifrovanie nikto síce nevidí obsah našej komunikácie, avšak vidí s kým komunikujeme (zdrojová a cieľová IP adresa). Ak chcem skryť aj túto informáciu, musíme použiť virtuálnu privátnu sieť.

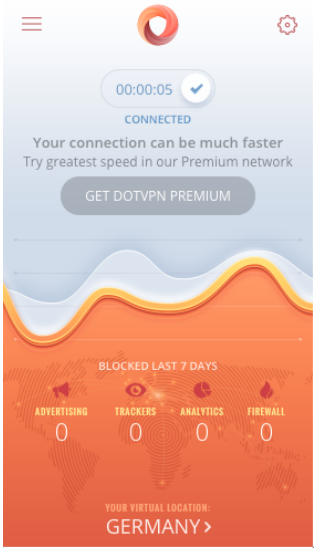
14.3.1 Virtuálna privátna sieť (VPN)

Koncept VPN je založený na tuneli podobnom ako sme si spomínali pri IPv6. Na počítači si nainštalujeme VPN klienta, ktorý sa pripojí na prevádzkovateľa VPN servera, vytvorí sa tak šifrovaný tunel, ktorým prechádza celý naša komunikácia. Náš poskytovateľ vidí len vytvorený tunel a nevidí čo je v ňom. Komunikáciu nám môže čítať len prevádzkovateľ VPN servera, no ak by to spravil, stratí dôveru. Väčšina VPN serverov je platená. Kompletný zoznam nájdeme na https://topvpnsoftware.com.



Obrázok 180 Princíp VPN

Ak chceme „tajne“ prezerať len web, postačí nám aj VPN rozšírenie prehliadača. Príkladmi takých rozšírení sú SETUP VPN, DOT VPN, HOLA.



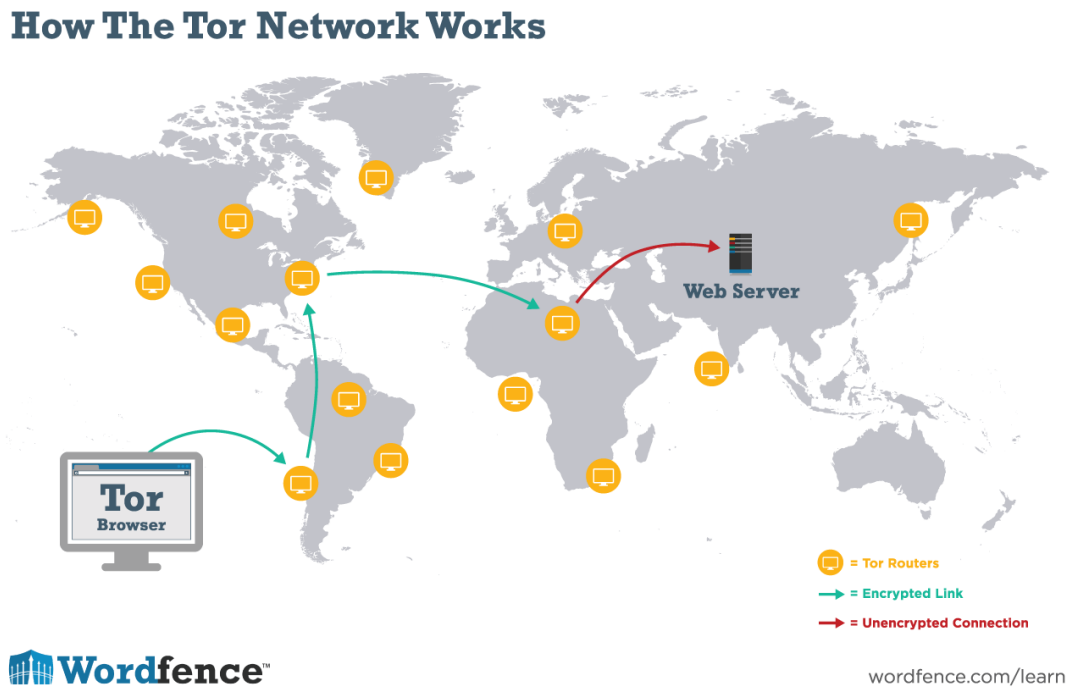
Obrázok 181 Rozšírenie prehliadača DOTVPN

ÚLOHA

Do prehliadača si nainštalujte VPN rozšírenie. Po úspešnom pripojení zobrazte svoju IP adresu na webe www.moja-ip.sk alebo podobnom.

14.4 Projekt TOR

Úplnú anonymitu poskytuje projekt TOR. Je to špeciálna počítačová sieť zložená z mnohých navzájom prepojených uzlov so špeciálne upraveným internetovým prehliadačom. Komunikácia medzi jednotlivými uzlami siete je šifrovaná vždy z iným kľúčom a výstupný bod zo siete je zvolený náhodne. Nie je možné spätne vystopovať cestu k odosielateľovi informácie.



Obrázok 182 Princíp TOR siete (http://www.wordfence.com/wp-content/uploads/2015/12/HowTorWorks_1340px.png)

Používaním Tor prehliadača majú všetci rovnakú verziu, rovnaké rozmery okna, rovnaké vlastnosti (spomeňte si na úvod, čo všetko prezrádza prehliadač). Tor sieť slúži na dobré aj zlé veci. V krajinách, kde neexistuje sloboda slova a prejavu a novinári a aktivisti sú prenasledovaní je Tor jednou z možností ako sa pred vládou skryť.

Okrem bežného Internetu Tor umožňuje prístup aj k jeho neviditeľnej časti – dark webu.

14.5 Deep web, dark web

Podľa dostupnosti informácií na webe, delíme web na

- Surface web
- Deep web
- Dark web

Surface web predstavuje stránky a obsah dostupný cez vyhľadávače (Google, Bing, ...). Táto časť webu predstavuje len 4% všetkého obsahu na webe. Základnou vlastnosťou je, že sa jedná o verejne dostupné informácie.

Deep web je pred vyhľadávačmi skrytý. Jedná sa o weby, ktoré majú prístup chránený heslom, nutnosťou registrácie. Sú to podnikové databázy, firemný intranet, vedecké databázy, knižnice. Nejedná sa o žiadny nelegálny obsah. Len strážený, kvôli autorským právam, firemným tajomstvám a pod.

Dark web je dostupný cez Tor sieť. Pre bežných používateľov je neviditeľný. Doménové mená v dark webe majú koncovu .onion a doménové názvy sú generované znaky. Na dark webe žije čierny trh, drogy, pornografia, zbrane, vrahovia, falšovanie, hackovanie.

Svetlou stránkou dark webu je prístup a zdieľanie informácií z krajín s cenzúrou webu (Bahrain, Belarus, China, Cuba, Ethiopia, India, Iran, North Korea, Pakistan, Russia, Saudi Arabia, Sudan, Syria, Turkmenistan, United Arab Emirates, United Kingdom, United States, Uzbekistan, Vietnam), novinári môžu získať tajné a dôverné informácie prípadne inkriminujúce informácie.

Poznámka pre učiteľa:

Vysvetlite žiakom, že Tor vznikol aby podporoval slobodu slova a prejavu pre krajiny kde to nie je samozrejmé. Čierny trh a nelegálny obsah je síce jeho súčasťou, no je potrebné poznať riziká a právne následky. Cez Tor si môžem objednať drogy, no ako mi ich doručia? Ak skutočne mi ich aj pošlú? Alebo budem len ďalšou obeťou podvodníkov. Na Tor sieti sa platí Bitcoinom.

Vyhľadávačom v sieti Tor je projekt DuckDuckGo. Je dostupný na adrese https://3g2upl4pq6kufc4m.onion (https://duckduckgo.com)

Zoznam významných webov dostupných na Tor sieti sa nachádza na HiddenWiki (http://zqktlwiuavvvqqt4ybvgvi7tyo4hjl5xgfuvpdf6otjiycgwqbym2qad.onion/wiki/index.php/Main_Page).

ÚLOHA

Z webu https://www.torproject.org/ si nainštalujte Tor Browser a prezrite si obsah Dark Webu.

ZHRNUTIE

Čo sme sa naučili

- Každá naša aktivita zanecháva v digitálnom svete stopu.
- Zdrojová IP adresa sa nachádza v odoslanom e-maily.
- Prehliadač poskytuje webovej aplikácii detailné informácie o operačnom systéme.
- Sieťová komunikácia môže byť monitorovaná poskytovateľom služieb internetu.
- Pre skrytie komunikácie používame šifrovaný prenos a VPN siete.
- Neviditeľná časť Internetu sa nazýva Deep Web a Dark web.
- Dark web predstavuje prevažne nelegálny obsah a je dostupný len cez sieť Tor.

15 ŠIFROVANIE A DÔVERYHODNOSŤ KOMUNIKÁCIE

CIEĽ

Cieľom tejto kapitoly je pochopiť základné princípy šifrovania a útokov slúžiacich na získanie hesla alebo šifrovacieho kľúča. Praktickou aplikáciou šifrovania je elektronický podpis, ktorý zabezpečuje pravosť dokumentu a istotu o tom, kto ho posiela. Pomocou občianskeho preukazu sa naučíte podpísať dokument a overiť jeho platnosť.

VÝKLAD

15.1 Šifrovanie

Pri posielaní dokumentov Internetom, ich vystavujeme dvom rizikám, že ich niekto môže prečítať alebo ich pozmeniť. Už v minulosti vznikali rôzne metódy, ktoré pozmeňovali pôvodný text na „nečitateľnú formu“. Takéto metódy nazývame **šifrovacie algoritmy**. Ide o postup, podľa ktorého sa pôvodný – vstupný (otvorený) text upraví, čím dostaneme výstupný – zašifrovaný text. Pre zašifrovanie sa používa **šifrovací kľúč**, ktorý by mali poznať iba odosielateľ a príjemca (prípadne sa kľúče používajú dva, vysvetlíme si ďalej v texte). Potom takýto text, správu môžeme poslať príjemcovi, ktorý musí vedieť ako daný text prečítať – dešifrovať (previesť do pôvodnej podoby). V minulosti používanými a najznámejšími algoritmami boli: **Caesarova šifra** (posun znakov), **Vigenerová šifra** (šifrovanie pomocou tabuľky), **Vermanova šifra** (používala sa na šifrovanie telegrafických správ) a iné.

V oblasti šifrovania existujú špecializované vedné disciplíny, ktoré sa ním zaoberajú. Hlavnou je **kryptológia**, ktorá sa zaoberá šifrovaním zo všetkých uhlov pohľadu. Kryptológia sa delí na kryptografiu a kryptoanalýzu. **Kryptografia** študuje šifrovacie algoritmy, kryptografické nástroje, hardvérové implementácie šifrovacích algoritmov, kryptografické protokoly a pod. **Kryptoanalýza** sa zaoberá lúštením šifier a v poslednej dobe jej význam získava stále viac na váhe, vďaka odhaľovaniu teoretických slabín bežne používaných šifier.

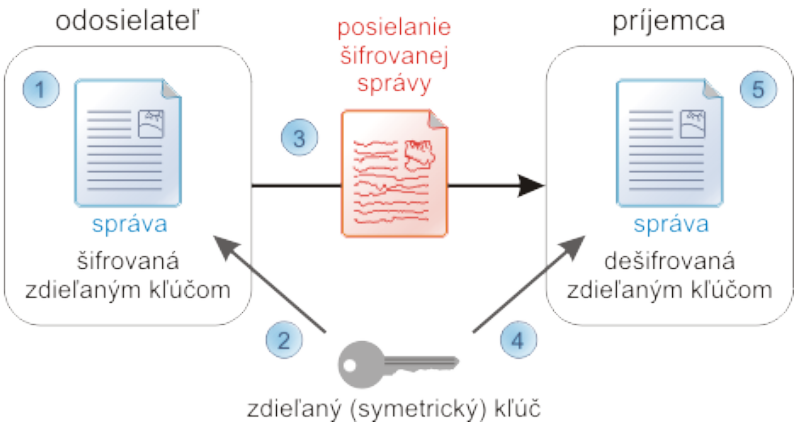
V informatike a v počítačových systémoch sa stretávame aj s pojmom kódovanie, resp. **kódovací algoritmus**. Jeho funkcia je rovnaká ako pri šifrovacom algoritme, ale tu sa nesnažíme informáciu ukryť, len ju zapíšeme iným spôsobom (ASCII, UNICODE, UTF – 8 a ďalšie). O kódovaní sme si už hovorili pri emailoch. Používali sme BASE64 kódovanie.

15.2 Metódy šifrovania

Existujú tri typy kryptografie (šifrovania): symetrická, asymetrická a hybridná, ktorá predstavuje spojenie symetrickej s asymetrickou.

15.2.1 Symetrická kryptografia

Pri šifrovaní sa používa tajný kľúč. V tomto prípade je jeden symetrický – zdieľaný kľúč pre obe strany komunikácie. Komunikujúci si vopred dohodnú spoločný tajný kľúč, ktorý budú používať. Pokiaľ nechceme, aby niekto iný okrem príjemcu vedel, čo posielame, tak musíme tento kľúč držať v tajnosti. Podobne nesmie tento kľúč vyzradiť ani príjemca. To znamená, že pokiaľ komunikujeme s viacerými ľuďmi a pri šifrovaní nechceme aby existovala možnosť, že by niekto nepovolaný prečítal našu správu, tak musíme používať zakaždým iný kľúč. Toto je hlavnou nevýhodou pri tomto type šifrovania (nutnosť väčšieho počtu kľúčov). Výhodou je rýchle šifrovanie a dešifrovanie.



Obrázok 183 Princíp symetrického šifrovania

Postup pri posielaní je nasledovný:

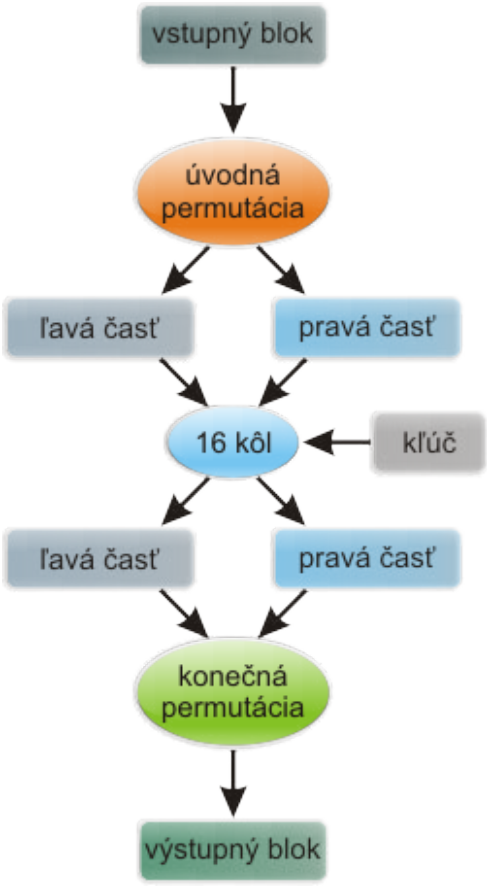
1) odosielateľ napíše text – správu
2) správa je pomocou šifrovacieho algoritmu a zdieľaného kľúča zašifrovaná
3) zašifrovaná správa sa pošle príjemcovi
4) príjemca správu dešifruje pomocou zdieľaného kľúča
5) príjemca si prečíta dešifrovanú správu

Symetrických algoritmov je viacej a najrozšírenejším je algoritmus DES (Data Encryption Standard), ktorý používa šifrovací kľúč s veľkosťou 64 bitov. Z neho vznikol algoritmus 3DES (používa 122 alebo 168 bitový kľúč). Ďalšími algoritmami sú napr. RC4 (Rivest Cipher 4 – používa 128 bitový kľúč) a v dnešnej dobe odporúčaný AES (Advanced Encryption Standard – 128, 192 alebo 256 bitový kľúč).

DES

Algoritmus DES (Data Encryption Standard) bol používaný americkými úradmi pre šifrovanie citlivých údajov štátnych orgánov (v dnešnej dobe je nahradený algoritmom AES). DES je bloková šifra, ktorá šifruje otvorený text po blokoch s dĺžkou 64 bitov. Rovnakú dĺžku má šifrovací kľúč,

každý ôsmy bit kľúča je však paritný - efektívna dĺžka kľúča je teda 56 bitov. **Je kombináciou substitučnej** (niečo sa nahrádza) **a transpozičnej** (niečo sa vymienia medzi sebou) **šifry.**



Obrázok 184 Princíp DES šifry

Na blok otvoreného textu je najprv aplikovaná jednoduchá **substitúcia** a následne je uskutočnená **úvodná transpozícia (permutácia)**. Ďalej sa 64 bitový blok delí na dve 32 bitové časti a nasleduje 16 kôl (cyklov - rovnakých operácií), v ktorých sú údaje kombinované s hodnotou kľúča. Nakoniec sa polovice spoja a uskutoční sa **konečná permutácia** – inverzia k úvodnej permutácii. Výsledkom je **šifrovaný text**.

Sekvencia činností počas jedného kola:

1) substitúcia, pomocou tzv. S-boxov - pevne stanovené substitučné tabuľky
2) permutácia zrealizovaná v P-boxe - pevne stanovené tabuľky, ktoré určujú zmenu poradia jednotlivých bitov práve šifrovaného bloku
3) výsledná hodnota je potom XOR-ovaná s istou časťou kľúča (kľúč sa pre každé kolo mení)

Týmito činnosťami prechádza iba pravá strana bloku a pri nasledovnom kole sú strany prehodené – ľavá strana je teraz pravou, doterajšia pravá časť bloku si odpočinie od permutácií a substitúcií na pravej strane.

Vylepšenou verziou je **3DES (TripleDES)**, ktorý zopakuje DES tri krát, s tromi rôznymi kľúčmi. Šifra DES dnes nie je považovaná za bezpečnú, dokonca sa neodporúča používať ani šifra 3DES. Náhradou je používanie šifry AES.

AES

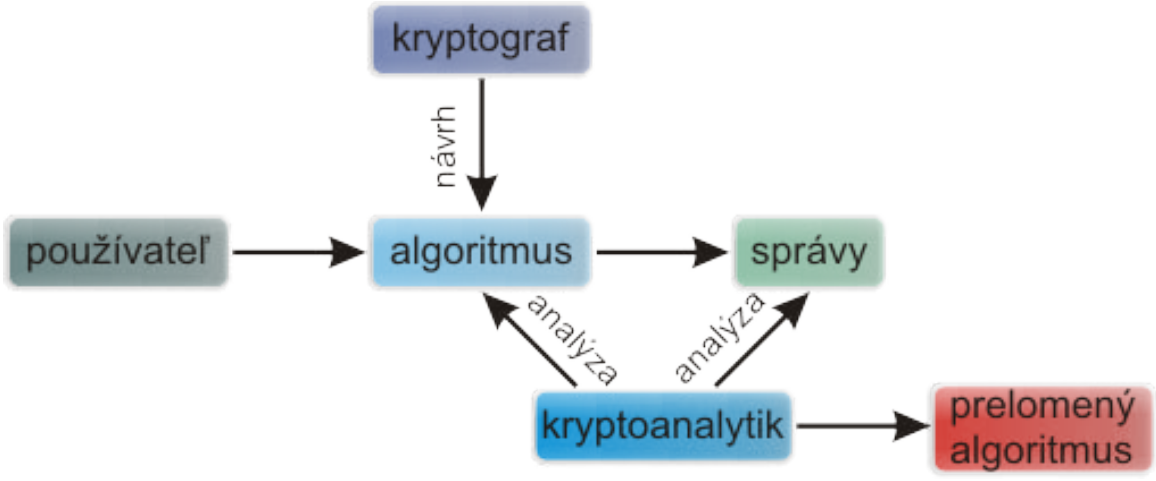
Algoritmus **Adanced Encryption Standard (AES)** je bloková šifry, ktorá šifruje otvorený text po blokoch. AES používajú okrem iných aj americké úrady, kde nahradil algoritmus DES a je v dnešnej dobe najpopulárnejším symetrickým šifrovacím algoritmom. Podobne ako DES aj AES je **kombináciou substitučnej a transpozičnej šifry**. Veľkosť šifrovacieho kľúča je 128, 192 alebo 256 bitov. Podobne ako pri DES aj tu sa používajú kolá, v ktorých sa na bloku vykonávajú 4 operácie. Pred začatím kôl je blok 128 bitov rozložený do matice 4x4 a prvkami matice sú bajty.

ÚLOHA

Na stránke https://www.tools4noobs.com/online_tools/encrypt/ si vyskúšajte šifrovanie a na stránke https://www.tools4noobs.com/online_tools/decrypt/ dešifrovanie pomocou rôznych algoritmov.

15.3 Formy útokov na kryptografické algoritmy

Vylúštenie šifry sa nazýva prelomenie šifrovacieho algoritmu a potom je možné vďaka prelomeniu čítať šifrovaný text, bez znalosti šifrovacieho, resp. dešifrovacieho kľúča.



Obrázok 185 Postup prelomenia šifry

Existuje viacej typov útokov:

- útok hrubou silou (brute force attack), pri ktorom skúšame všetky možné kombinácie šifrovacieho kľúča (počet možných kľúčov je 2n, kde n je dĺžka kľúča)

- lúštenie so znalosťou šifrovaného textu (cipher only text) - k dispozícii máme niekoľko textov zašifrovaných rovnakým kľúčom a algoritmom
- lúštenie so znalosťou otvoreného textu (known – plaintext attack), pričom sa snažíme získať kľúč
- lúštenie so znalosťou vybraných otvorených textov (chosen plaintext attack)
- lúštenie so znalosťou vybraných šifrovaných textov (chosen ciphertext attack)
- lúštenie pomocou kompromitácie používateľa (purchase key attack)
- priame získanie kľúča od vlastníka rôznymi metódami (podplatenie, mučenie)

15.4 Asymetrická kryptografia

Používa iné matematické metódy ako symetrická kryptografia a rozdiel je aj v počte potrebných kľúčov. Tu sa definujú dva typy kľúčov (používajú sa ako pár): verejný a súkromný. **Verejný kľúč** sa používa na zašifrovanie textu (správy) a **súkromný kľúč** sa používa na dešifrovanie textu (správy). Vďaka tomu nie je nutné, aby odosielateľ správy chránil verejný kľúč príjemcu, keďže ním text nerozšifrujeme. Ďalej sa verejný kľúč dá použiť pri rôznych odosielateľoch (nie je nutné mať s každým vlastný kľúč ako pri symetrickom šifrovaní), čo nám uľahčuje jeho distribúciu. Jednoducho si náš verejný kľúč umiestnime na web. Potenciálny odosielateľ si ho tam nájde a môže nám poslať šifrovaný text.

Pre dešifrovanie textu príjemca použije svoj súkromný kľúč, ktorý má utajený, čo zaručuje, že nikto okrem neho nerozšifruje správu. Toto sa berie ako výhoda asymetrickej kryptografie - každý používateľ sa stará o bezpečné uloženie jediného kľúča. Nevýhodou je veľká časová náročnosť, s ktorou prebieha šifrovanie a dešifrovanie.

Uveďme si postup pri poslaní šifrovanej správy a odpovede na ňu:

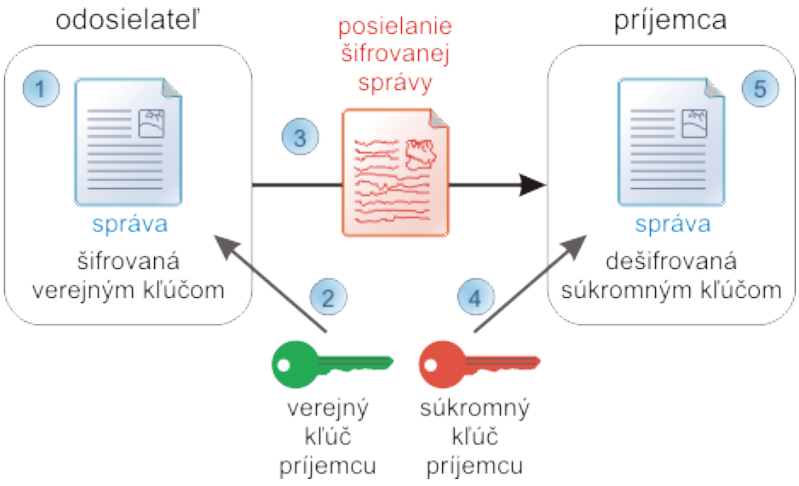
1) odosielateľ napíše text – správu

2) správa je pomocou šifrovacieho algoritmu a verejného kľúča príjemcu zašifrovaná (nikto okrem toho čo vlastní súkromný kľúč ju nevie otvoriť)

3) zašifrovaná správa sa pošle príjemcovi

4) príjemca správu dešifruje pomocou svojho súkromného kľúča

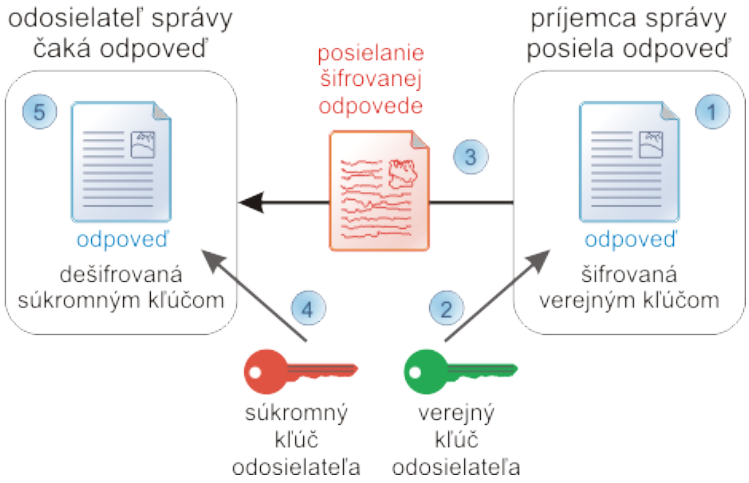
5) príjemca si prečíta dešifrovanú správu



Obrázok 186 Odoslanie správy

Príjemca sa rozhodol poslať odpoveď, na web stránke odosielateľa (prípadne na inom zdroji, napr. odosielateľ svoj verejný kľúč uviedol v správe,...) si našiel jeho verejný kľúč. Po napísaní odpovede použije na zašifrovanie pre algoritmus tento verejný kľúč a dostane zašifrovanú odpoveď.

Postup je obdobný len s iným párom kľúčov:



Obrázok 187 Odoslanie šifrovanej odpovede

Podobne aj tu existuje viacej algoritmov a najviac používaným je algoritmus RSA (Rivest Shamir Aleman). Pre RSA sa odporúča používať šifrovací kľúč veľkosti aspoň 1024 bitov a často sa používajú kľúče s veľkosťou 2048 alebo 4096 bitov. Ďalším asymetrickým algoritmom je ECC (Elliptic Curve Cryptography). Pre porovnanie ku 1024 bitovému RSA kľúču odpovedá 160 bitový ECC kľuč, pri zachovaní podobnej úrovne bezpečnosti.

Pri asymetrickej kryptografii je dôležitý Diffie-Hellmanov algoritmus (DH algoritmus). Používa sa na vytvorenie dvojíc kľúčov, ale aj pre vytvorenie symetrického kľúča.

15.5 Hybridná kryptografia

Výhodou symetrických šifier je ich rýchlosť výpočtu, nevýhodou zdieľaný kľúč. Výhodou asymetrických je pár kľučov, nevýhodou výpočtová zložitosť. Hybridná kryptografia spája tieto

dva prístupy tak, že na začiatku sa vygeneruje unikátny symetrický kľuč, ktorý sa medzi komunikujúcimi stranami vymení cez asymetrickú šifru. Prenos kľúča je bezpečný a zvyšok komunikácie sa šifruje symetricky.

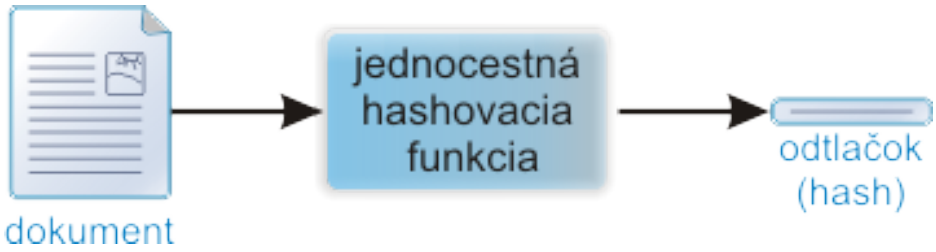
15.6 Digitálny podpis

Digitálny podpis je mechanizmus, ktorý zaručuje dve základné požiadavky pre posielaný (resp. prijatý dokument). Ide o **integritu dokumentu** (kde je zaručené, že dokument od odosielateľa nebol nijako pozmenený) a **nepopierateľnosť dokumentu** (pravosť dokumentu – že ho poslal skutočný odosielateľ).

15.6.1 Integrita dokumentu

Pre zaistenie integrity posielaného dokumentu sa používa jednocestná matematická funkcia odtlačku. Odtlačok (hash) je krátky textový reťazec vytvorený pomocou hashovacej funkcie z ľubovoľne veľkého textu (dokumentu). Typická veľkosť výsledného textu je 16 bajtov (algoritmus MD-5 [Message-Digest]) alebo 20 bajtov (algoritmus SHA-1 [Secure Hash Algorithm]).Ďalšími algoritmami sú SHA-224 (28 B), SHA-256 (32 B), SHA-384 (48 B) a SHA-512 (64 B).

Odtlačok sa aj pri malej zmene dokumentu tiež zmení. To zaručuje že dva rôzne dokumenty (texty) nemôžu mať rovnaký odtlačok, a teda tým je zaručená integrita dokumentu. Zároveň tiež platí, že z odtlačku neviem spätne poskladať pôvodný text.

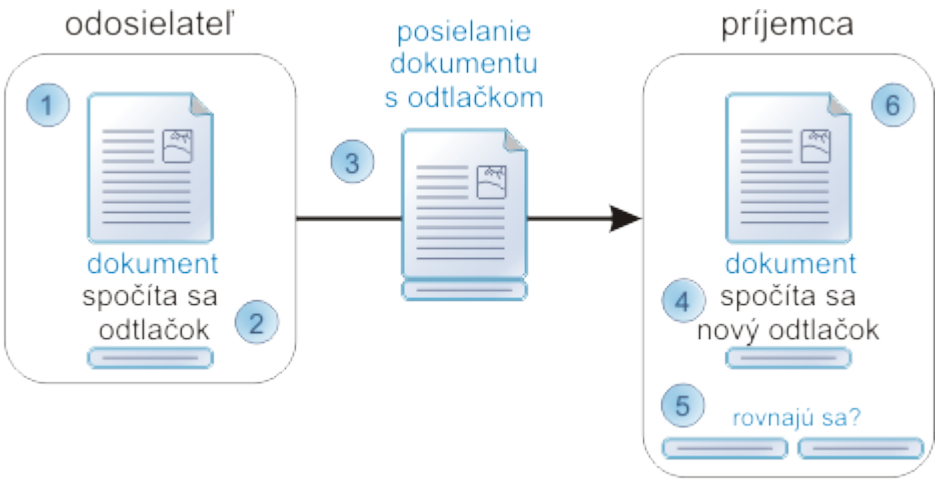


Obrázok 188 Princíp hashovacej funkcie

ÚLOHA

Na stránke https://passwordsgenerator.net/sha1-hash-generator/ si vyskúšajte generovanie hash odtlačkov pomocou rôznych funckií.

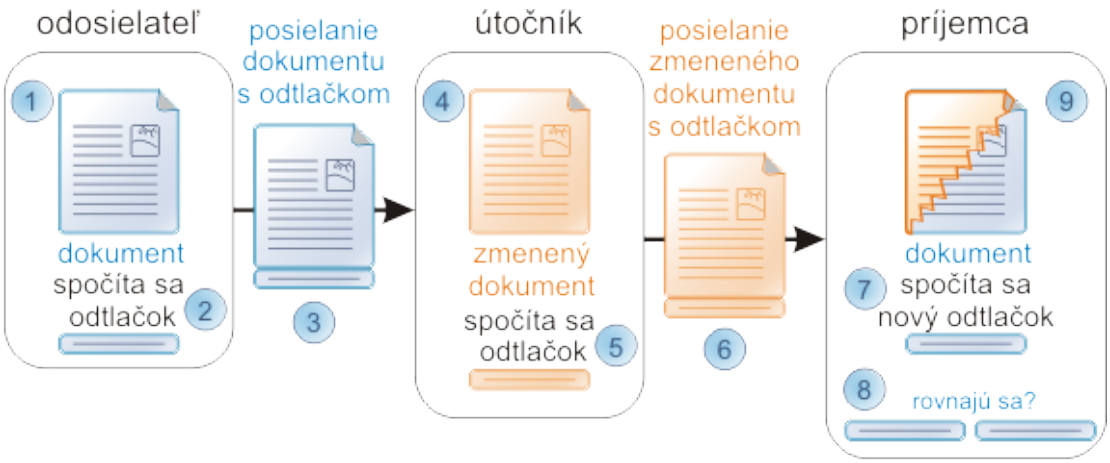
Potom pri posielaní dokumentu sa k nemu priloží odtlačok. Po prijatí správy príjemca vypočíta odtlačok z prijatého dokumentu a porovná svoj výsledok s tým, ktorý bol pripojený k správe. Ak je zhodný, správa sa po ceste nezmenila.



Obrázok 189 Overenie odtlačku pripojeného k správe

1) odosielateľ napíše dokument
2) vypočíta odtlačok z dokumentu
3) pošle dokument aj s odtlačkom
4) príjemca vypočíta odtlačok z prijatého dokumentu
5) porovná prijatý odtlačok s novým
6) príjemca číta dokument

Takýmto spôsobom možno detekovať chyby v dokumente vzniknuté pri prenose, ale aj ak ho niekto pozmenil. Keďže hashovacie algoritmy sú všeobecne známe, tak samotná integrita nám nezaručuje pravosť dokumentu (útočník má možnosť upraviť správu a nanovo spočítať odtlačok).



Obrázok 190 Útočníkom modifikovaný odtlačok

1) odosielateľ napíše dokument

2) vypočíta odtlačok z dokumentu

3) pošle dokument aj s odtlačkom

4) útočník zachytí dokument a pozmení ho

5) útočník spočíta nový odtlačok z pozmeneného dokumentu

6) útočník pošle zmenený dokument s novým odtlačkom príjemcovi

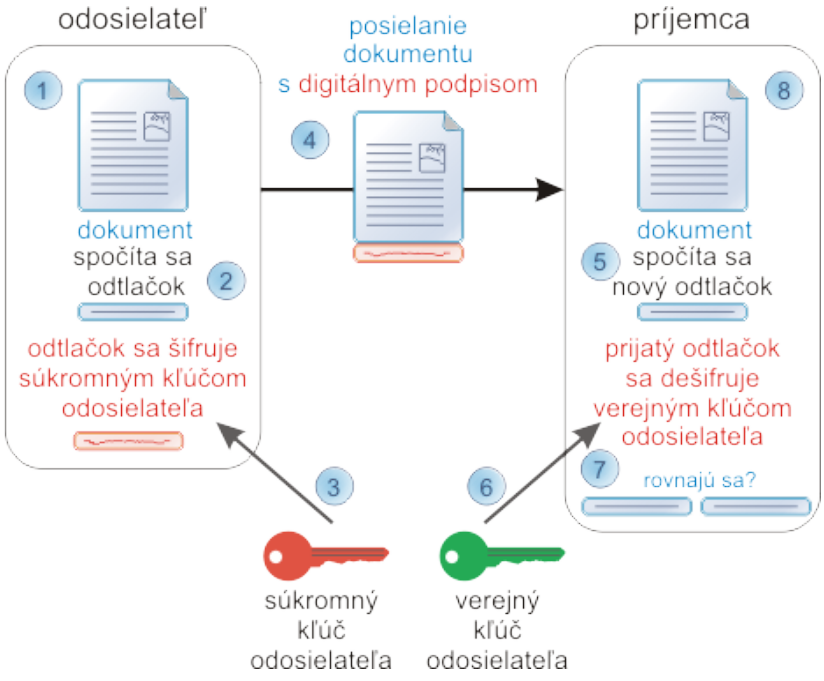
7) príjemca vypočíta odtlačok z prijatého dokumentu

8) porovná prijatý odtlačok s novým

9) príjemca číta dokument a myslí si, že prišiel od odosielateľa, ale v skutočnosti číta dokument od útočníka

15.6.2 Pravosť dokumentu

Odtlačok nám zabezpečuje integritu dokumentu a jeho pravosť zabezpečuje **digitálny podpis**. Digitálny podpis je založený na princípe asymetrickej kryptografie – pár kľúčov. Postup pri podpisovaní je nasledovný:



1) odosielateľ napíše dokument

2) vypočíta odtlačok z dokumentu

3) odtlačok šifruje svojím súkromným kľúčom, vznikne tým digitálny podpis

4) pošle dokument aj s digitálnym podpisom

5) príjemca vypočíta odtlačok z prijatého dokumentu

6) príjemca dešifruje digitálny podpis odosielateľovým verejným kľúčom

7) porovná prijatý odtlačok s novým

8) príjemca číta dokument

15.6.3 Certifikovaný kľúč, certifikačná autorita a elektronický podpis

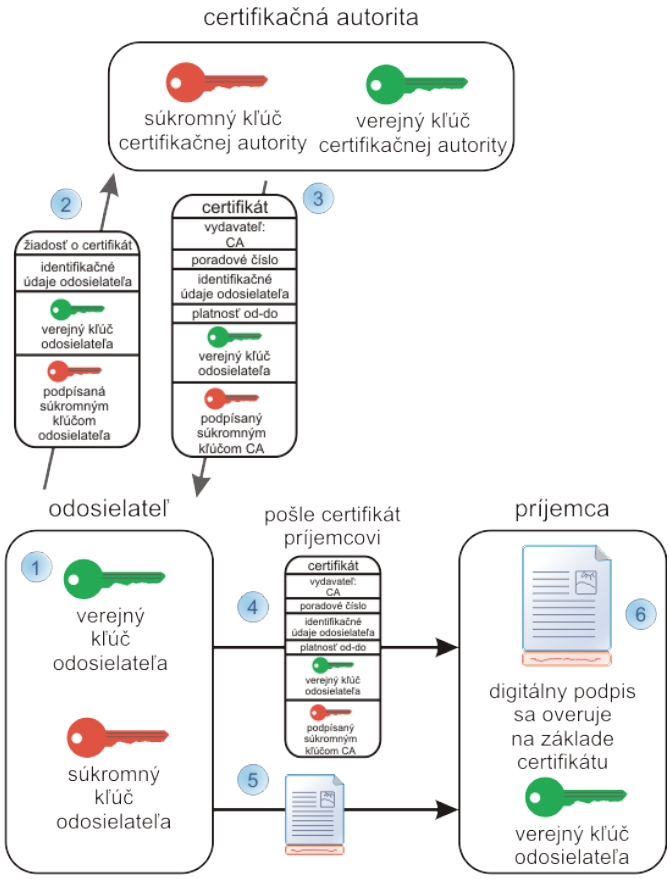
Otázne je ako ochrániť verejný kľúč pred podvrhnutím. Možností je viac:

- kľúč dostaneme priamo na stretnutí od jeho vlastníka (je iste že je to jeho kľúč),
- prípadne si ho overíme iným spôsobom (napr. telefonicky).

V prostredí Internetu sa na zaručenie pravosti používa „tretia osoba“ (podobne ako keď niekto pozná našu totožnosť a zaručí sa, že naozaj sme tí čo tvrdíme). Táto tretia osoba sa nazýva certifikačná autorita a svojím digitálnym podpisom potvrdzuje, že daný verejný kľúč patrí ku danej osobe. Potvrdenie sa realizuje formou certifikátu pravosti. Takto získame certifikovaný kľúč a potom pri komunikácii postačí keď príjemca dostane certifikát odosielateľa. Z certifikátu získa osobné údaje odosielateľa a jeho verejný kľúč, pomocou ktorého môže overiť digitálny podpis. Spojením digitálneho podpisu a certifikátu dostávame elektronický podpis.

Certifikačná autorita je dôveryhodná organizácia, ktorá vydáva certifikáty verejných kľúčov. Základným majetkom každej certifikačnej autority je pár kľúčov. Verejný kľúč je zverejniteľný na Internete, a súkromným kľúčom sú podpisované všetky vydávané certifikáty. Strata súkromného kľúča by znamenala neplatnosť všetkých vydaných certifikátov verejných kľúčov klientov.

Požívateľ (klient), ktorý si chce certifikovať svoj verejný kľúč, si musí najskôr pripraviť žiadosť a odoslať ju certifikačnej autorite. V žiadosti uvedie všetky svoje osobné údaje, priloží verejný kľúč a všetko podpíše príslušným súkromným kľúčom. Podpísaním žiadosti svojim súkromným kľúčom žiadateľ dokazuje, že je naozaj vlastníkom súkromného kľúča k certifikovanému verejnému kľúču.



Obrázok 191 Princíp podpisu certifikačnou autoritou

1) odosielateľ si vytvorí (vygeneruje) dvojicu kľúčov napr. pomocou DH algoritmu

2) vytvorí žiadosť o certifikát a pošle ju certifikačnej autorite (CA)

3) ak je žiadosť v poriadku, certifikačná autorita vystaví certifikát a pošle ho odosielateľovi

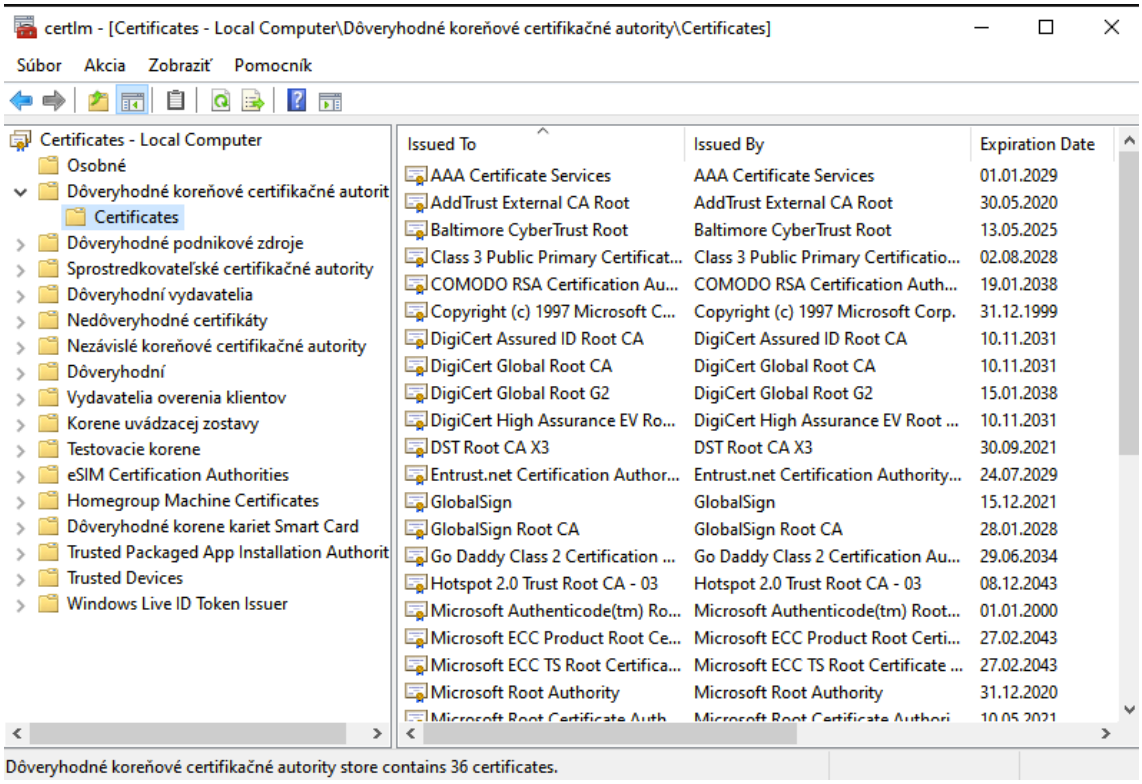
4) odosielateľ pošle svoj certifikát príjemcovi

5) ďalej už posiela podpísané dokumenty

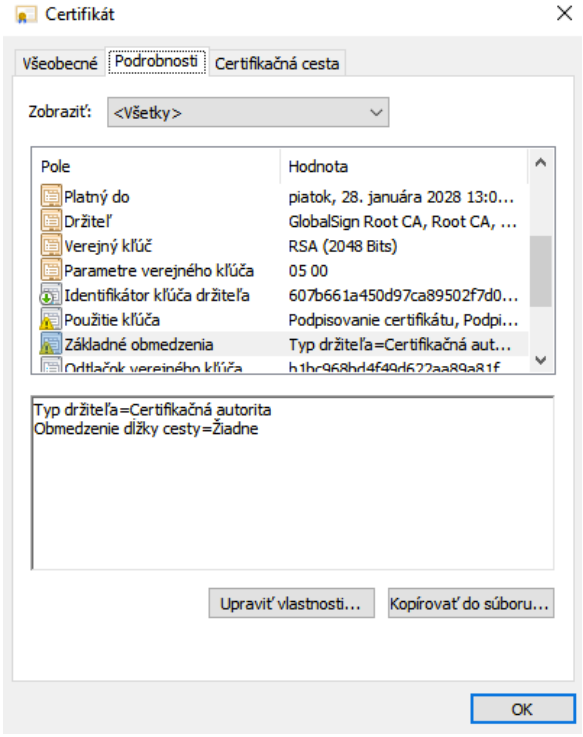
6) príjemca overuje odpis na základe certifikátu.

Certifikát je dátová štruktúra, ktorej základom je certifikovaný verejný kľúč. Ďalej obsahuje identifikačné údaje vlastníka tohto kľúča, názov vydavateľa certifikátu, poradové číslo certifikátu a ďalšie voliteľné údaje. Celý je podpísaný súkromným kľúčom certifikačnej autority.

V systéme Windows nájdite v Ovládacom paneli „Spravovať certifikáty počítača“.



Obrázok 192 Certifikáty počítača v systéme Windows



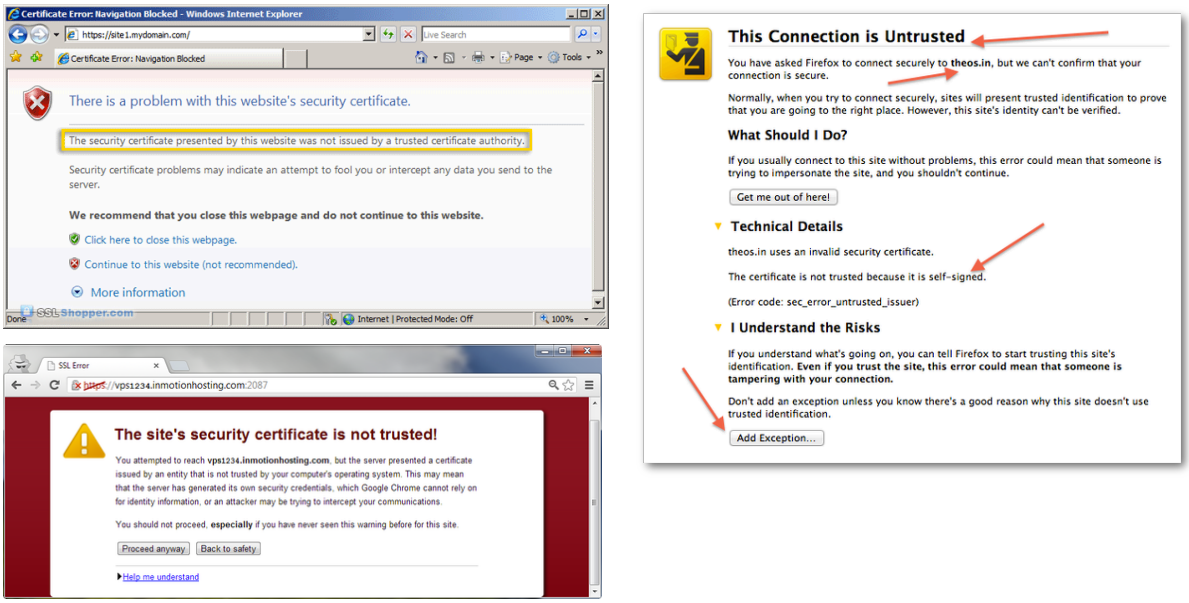
Obrázok 193 Detail certifikátu

15.7 Certifikáty na webe

Z dôvodov už vyššie spomínaných dnes šifruje komunikáciu už snáď každý web. Certifikáty na webe môžeme rozdeliť podľa úrovne „bezpečnosti“ do niekoľkých kategórií. Slovo bezpečnosť je v úvodzovkách len preto, že pri všetkých kategóriách sú naše údaje v bezpečí rovnakým spôsobom. Jednotlivé kategórie rozoznáme v adresnom riadku prehliadača podľa farby zámku alebo pásiku.

- **DV – Domain Validated (zelený zámok)** – Cenovo sa jedná o pomerne lacné riešenie. Certifikačná autorita overuje vlastníctvo domény napríklad zaslanie mailu na danú doménu (väčšinou na adresu webmaster@nazovdomeny), vytvorením nejakého vopred zadaného súboru na webe alebo vytvorením špeciálneho záznamu DNS a pod.). Riešenie, ktoré je zadarmo poskytuje autorita Letsenctypt.
- **OV – Organization Validation (zelený zámok)**. Cenovo drahšie riešenie, pri ktorom sa overuje sa pravdivosť informácií o organizácii, ktoré sú uvedené v certifikáte. Overenie sa vykonáva vo vládnych registroch, telefonuje sa na telefónne číslo zverejnené na webe ale aj inde ako len na vlastnom webe, realizuje sa rozhovor.
- **EV – Extended Validation (zelený pás s menom organizácie)** Najdrahšie riešenie, pri ktorom okrem overenia organizácie vo vládnych registroch sa vyžaduje zaslať aj oficiálne dokumenty o vzniku, overenie notárom apod.

V minulosti správcovia webu, ktorí nechceli platiť za certifikát si vytvárali vlastnú certifikačnú autoritu a to podpisovali vlastné certifikáty. Prenos údajov síce bol šifrovaný, ale používateľ nemal istotu, či komunikuje s pravým webom. Prehliadače zároveň zobrazujú varovanie a ponúkajú používateľovi možnosť pridať výnimku. Keďže dnes už máme Letsencypt, nie je dôvod na takého správanie. **Ak prehliadač zobrazí varovanie o certifikáte, treba z takého webu odísť.**



Obrázok 194 Varovanie o certifikáte od neznámej autority

ÚLOHA

Otvorte web https://www.ukf.sk a kliknite na ikonu zámku (certifikátu). Zobrazia sa detaily certifikátu. Zistite informácie, ktoré nesie certifikát (doba platnosti, vydavateľ, názov organizácie, a pod.)

15.8 Kvalifikovaný elektronický podpis

V slovenskom právnom systéme nahrádza kvalifikovaný elektronický podpis (KEP) náš podpis, ktorý je overený u notára alebo na mestskom či obecnom úrade. Certifikáty k podpisovaniu máme uložené v čipe občianskeho preukazu. Pred jeho vydaním nám informácie o našej identite overil pracovník na oddelení dokladom policajného zboru a certifikát je podpísaný Národným bezpečnostným úradom.

KEP nám zaručuje

- **autenticitu** – možno jednoznačne overiť identitu subjektu, ktorý podpis vytvoril,
- **integritu** – možno preukázať, že po podpísaní dokumentu nedošlo k žiadnej úmyselnej alebo neúmyselnej zmene obsahu dokumentu, aký bol v čase jeho podpisovania,
- **nepopierateľnosť** – autor nemôže tvrdiť, že nevyhotovil daný podpis elektronického dokumentu.

Aby sme mohli vytvoriť KEP potrebujeme

- elektronický dokument,
- súkromný kľúč uložený na bezpečnom zariadení,
- verejný kľúč patriaci k súkromnému kľúču, na ktorý bol vydaný akreditovanou certifikačnou autoritou kvalifikovaný certifikát,
- prostriedok na vyhotovenie elektronického podpisu certifikovaný NBÚ SR

V súlade so zákonom o e-Govermente je občiansky preukaz s čipom bezpečným prostriedkom na uloženie kryptografických kľúčov (súkromný, verejný) a kvalifikovaného certifikátu určených na vytváranie kvalifikovaného elektronického podpisu. Kľúče a certifikát sú uložené v pamäti čipu občianskeho preukazu, kde súkromný kľúč nie je možné žiadnym spôsobom z tejto pamäte vyexportovať, t. j. existuje len v jedinom vyhotovení v konkrétnom občianskom preukaze.

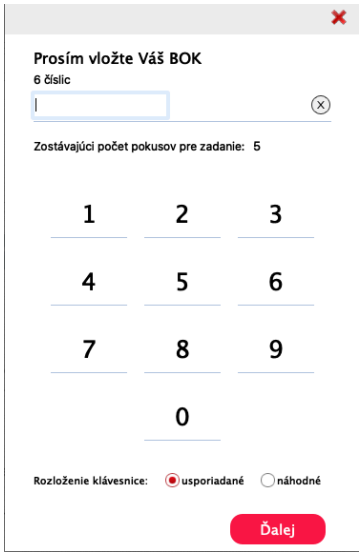
Obrázok 195 Občiansky preukaz s čipom

Dokument podpísaný občianskym preukazom obsahuje naše osobné údaje a preto neodporúčame aby ste ho posielali komukoľvek. Primárny účel je na komunikáciu so štátom a verejnou správou. Napriek tejto skutočnosti si však podpísanie dokumentu a overenie platnosti vyskúšame.

Na webe https://zep.disig.sk/ máme možnosť podpísať aj overiť podpis dokumentu. Pripravte si ľubovoľný z podporovaných dokumentov, čítačku k občianskemu preukazu a občiansky preukaz.

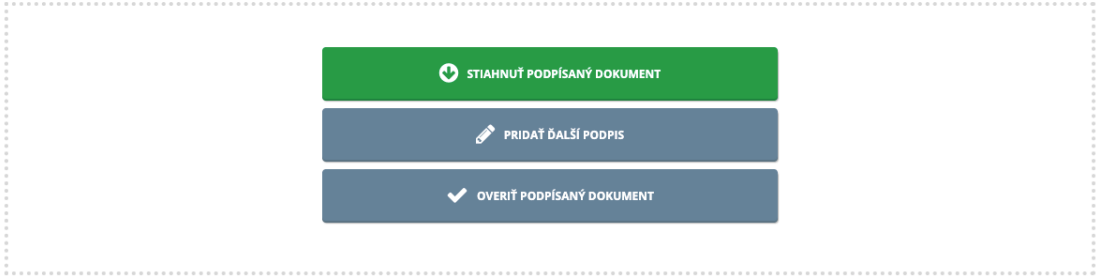
Obrázok 196 Podporované dokumenty na podpis na zep.disig.sk

Po vybraní súboru sa zobrazí možnosť podpísať dokument dvoma spôsobmi. Buď cez občiansky preukaz alebo pomocou mobilného telefónu (QR kód). Pre komunikáciu so štátom potrebujeme originál občiansky preukaz. Po stlačení Podpísať sa načíta dokument z v pravom dolnom rohu máme opäť tlačidlo Podpisať. Následne sa zobrazí výzva na zadanie BOK a KEP kódov.



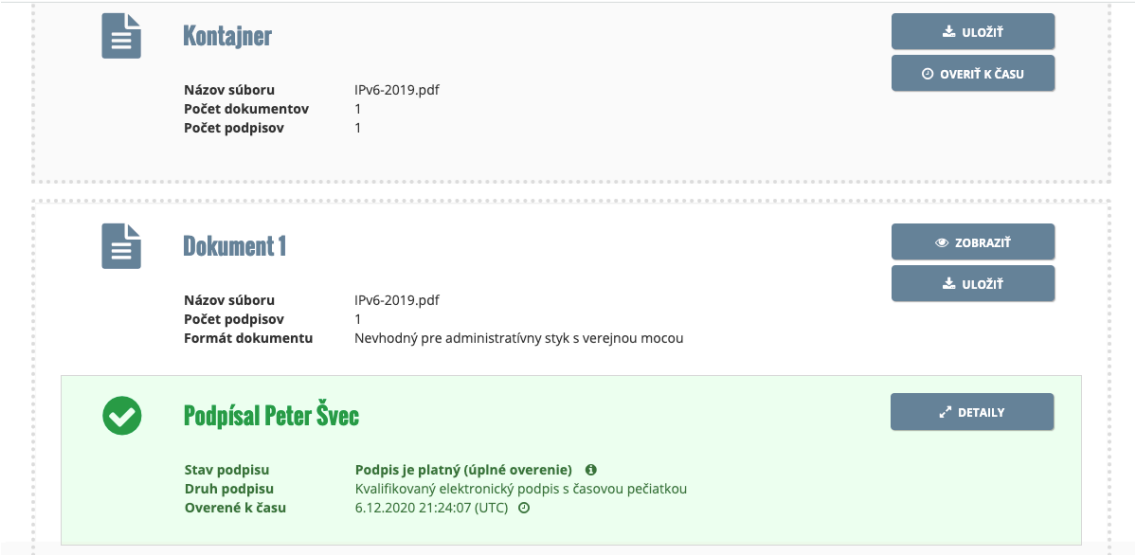
Obrázok 197 Zadanie BOK kódu

Následne môžeme dokument stiahnuť alebo overiť podpis dokumentu. Prípadne môžeme pridať ďalší podpis. To používame v situácii, keď jeden dokument má podpísať viac osôb.



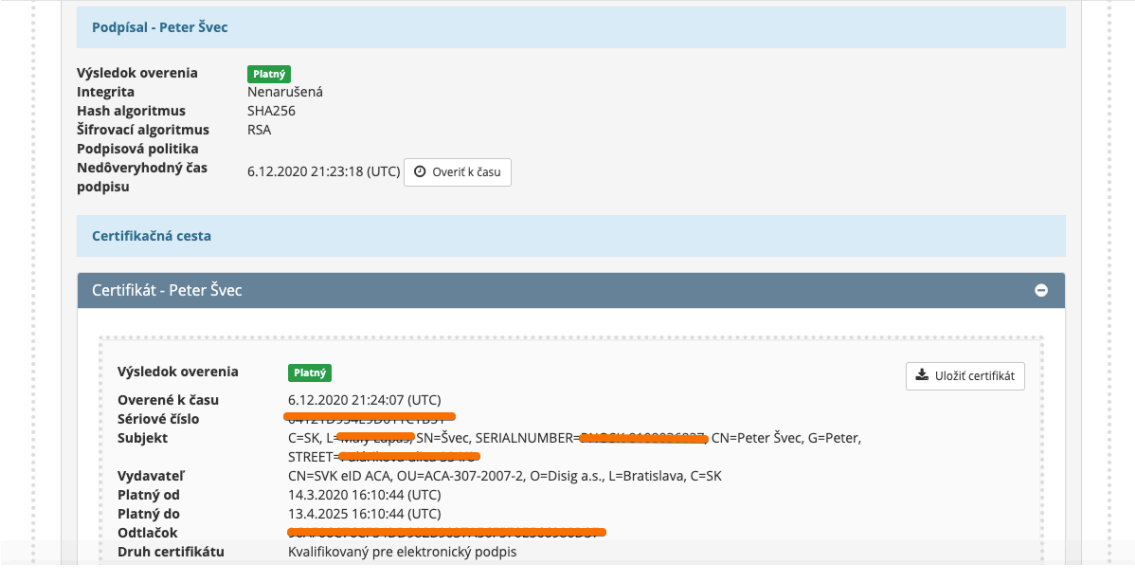
Obrázok 198 Informácia o podpísanom dokumente

Po stlačení tlačidla **Overiť podpísaný dokument** sa zobrazia ďalšie informácie o podpísanom súbore.



Obrázok 199 Možnosť overiť podpísaný dokument

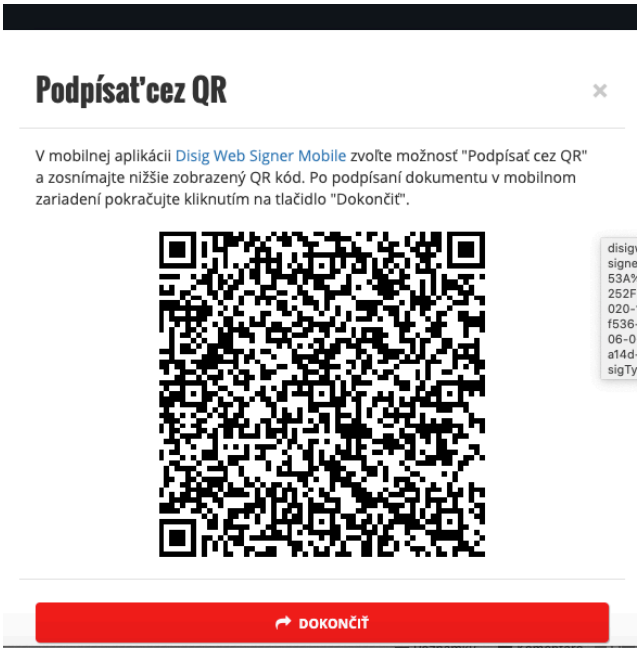
Stlačením tlačidla **Detaily** sa zobrazia detailné informácie o podpise.



Obrázok 200 Detail certifikátu podpisu

Podpisovať mobilom môžeme až vtedy, keď prejdeme procesom popísaným na https://zep.disig.sk/CertEnroll/Portal/

Vtedy stačí mobilom v aplikácii **Disig Web Signer Mobile** zosnímať QR kód. V tomto prípade nepotrebujem vkladať občiansky preukaz do čítačky.



Obrázok 201 Podpisovanie cez QR kód

ZHRNUTIE

Čo sme sa naučili

- Hashovacie funkcie slúžia na overenie originality dokumentu.
- Symetrická kryptografia je založení na spoločnom zdieľanom tajomstve.
- Asymetrická kryptografia je založená na súkromnom a verejnom kľúči.
- Certifikáty pre kvalifikovaný elektronický podpis máme uložené na občianskom preukaze.

LITERATÚRA POUŽITÁ V KAPITOLÁCH 11 AŽ 15

1. Cisco Network Academy . Understanding IPv6 Link Local Address.
2. Crispin M. – RFC 3501 Internet Message AccessProtocol - Version 4rev1
3. Crocker D. H. – RFC 822 ARPA INTERNET TEXT MESSAGES
4. Dostálek L. a Kabelová A. - Velký průvodce protokoly TCP/IP a systémem DNS
5. Fielding a kol., RFC 2616 HTTP/1.1
6. Klensin J. – RFC 2821 Simple Mail TransferProtocol
7. Myers J.,Mellon C., Rose M. - RFC 1939 Post Office Protocol - Version 3
8. Naik, D. – Internet, standardy a protokoly
9. Postel J., Reynolds J., RFC 959 File Transfer Protocol (FTP)
10. Pužmanová, R. - Moderní komunikační sítě od A do Z
11. Resnick P. – RFC 2822 Internet Message Format
12. Satrapa, P. . IPv6: internetový protokol verze 6. CZ. NIC, zspo.
13. St. Clair College in Windsor . Cisco network academy presentations.
14. Tanenbaum, A.S. – Computer Networks

Počítačové systémy a siete

Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autori: Peter Švec, Štefan Koprda, Jarmila Škrinárová, Vladimír Siládi

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2020

Vydanie : 1. vydanie

Počet strán: 244

ISBN: 978-80-89965-77-9

Bratislava 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.