

Week-1 – Internet, Terminál, Html

1 Internet/ Áttekintés

- Bandwith (Sávszélesség)
- Bitrate (Bitráta) bitsebesség
- **Packet (Csomag).** A digitális hálózatokban az adatátvitel egyik módja az átküldendő adat kisebb csomagokra bontása, és azok továbbítása, ezek a csomagkapcsolt hálózatok. Egy csomag az alkalmazott hálózati protokollnak megfelelő struktúrával (formátummal) bír, és nem léphet túl egy bizonyos méretet. A csomagokra bontás lényege a kommunikációs vonalak előnyösebb kihasználásában rejlik.
- Router ("Útválasztó")
- **Transmission Control Protocol - TCP (átvitelvezérlő protokoll)** az internet gerincét alkotó TCP/IP protokollcsalád egyik fő protokollja. . A TCP a család két eredeti komponense közé tartozik, az Internet Protocolt (IP) egészíti ki, így együtt TCP/IP néven szokás hivatkozni rájuk. A TCP/IP protokollhierarchia szállítási rétegét valósítja meg. Az internet legfontosabb szolgáltatásainak nagy része TCP-n keresztül érhető el: ilyen pl. a World Wide Web és az e-mail. Más alkalmazások, melyeknél a kisebb késleltetés fontosabb a csomagvesztés elkerülésénél, a User Datagram Protocolt (UDP) használhatják.
- **Internet protocol - IP (Internet protokoll)** az internet (és internetalapú) hálózat egyik alapvető szabványa (avagy protokollja). Ezen protokoll segítségével kommunikálnak egymással az internetre kötött csomópontok (számítógépek, hálózati eszközök, webkamerák stb.). A protokoll meghatározza az egymásnak küldhető üzenetek felépítését, sorrendjét stb.
- Internet Service Provider - ISP (Internet-szolgáltató)
- **IP Address (IP cím)** egy egyedi hálózati azonosító, amelyet az internetprotokoll segítségével kommunikáló számítógépek egymás azonosítására használnak.
- Domain Name System - DNS (tartománynév szolgáltató)
- HTTP HyperText Transfer Protocol
- HTTPS Hypertext Transfer Protocol Secure
- SSL Az SSL tanúsítványok arra szolgálnak, hogy létrejöhessen egy biztonságos, titkosított csatorna a kliens és a szerver között.
- HTTP methods (HTTP metódusok)
- HTTP status codes (HTTP állapot kódok)
- Encryption (Titkosítás)
- Public Key (Publikus kulcs)
- Private Key (Privát kulcs)
- Virus
- **DoS, DDoS** A szolgáltatásmegtagadással járó támadás (**Denial of Service** vagy **DoS**), más néven túlterheléses támadás, illetve az elosztott szolgáltatásmegtagadással járó támadás (**Distributed Denial of Service, DDoS**) informatikai szolgáltatás teljes vagy részleges megbénítása, helyes működési módjától való eltérése.
- Phishing (adathalászat)
- Backdoor (hátsó kapu)
- Bot Net (bot hálózat)

2 Terminal/Áttekintés

- Utasítások
 - pwd - aktuális könyvtár elérési útja – print work directory
 - cd - könyvtár váltás – change directory
 - ls / dir - könyvtár tartalmának listázása –list directory
 - mkdir - könyvtár létrehozása –make directory
 - cp - fájl/könyvtár másolása -copy
 - rm - fájl/könyvtár törlése -remove
 - rm -R - könyvtár törlés rekurzívan
 - touch - fájl létrehozása
 - mv - fájl mozgatása/átnevezése -move
- Struktúra
 - Szülő könyvtár - ?
 - Gyermekek könyvtár - ?
 - Gyökér könyvtár ("root" könyvtár) - ?
 - Home könyvtár - ?
- Billentyűparancs
 - Előző utasítás újra elvégzése - fölfelé nyíl + enter
 - [Automatikus kiegészítés \(auto complete\)](#) - tab

3 HTML/Áttekintés

- **HTML** HyperText Markup Language=hiperszöveges jelölőnyelv

a weblap szerkezetét határozza meg (főcímek, bekezdések, stb), különböző elemeket (képek, videók) ágyazz webs dokumentumba.

- **Markup Language** (Leíró nyelv)
- **Programozási nyelv** – xhtml, html, html5
- **Alap HTML Struktúra**
 - `<!DOCTYPE html>` - dokumentum típus definiálása
 - `<html></html>` / - adatait, utasításait tartalmazza
 - `<head></head>` - fejléc
 - `<body></body>` - dokumentumtörzs Ezen elemek között kell elhelyezni mindent: a szöveget, hivatkozásokat, képeket, stb. (A keretek és a JavaScript kódok kivételével!)
 - `<TITLE>` - Ide jön az oldal címe, ezt a szöveget fogod látni a böngésző címsorában`</TITLE>`
 - `<footer></footer>` lábjegyzet

HTML Szintaxis

- Elemek
 - Nyitó Tag
 - Attribútumok
 - név
 - érték
 - Tartalom
 - Záró Tag

Betűtípusok, stílusok <i> <u> <tt>		
Ha ezt a kódot beírod a dokumentum törzsbe		ez lesz az eredménye
Kövér betűk (bold) 		Kövér betűk (bold)
<i>Dőlt betűk (italic)</i>		Dőlt betűk (italic)
<u>Aláhúzott betűk (underlined) </u>		Aláhúzott betűk (underlined)
<tt>Írógép betűk (teletype) </tt>		Írógép betűk (teletype)

- Elemek

- <h1> - <h6> - A fejlécek - mint ahogy a szövegszerkesztésnél is megszokhattuk - a html oldalak logikai felosztását teszik lehetővé. pl. h1 az oldal címe, h2 egy alcím, h3 annak az alcíme és így tovább. A HTML oldalak esetén 6 fejlécet használhatunk)
 - • <hr> - Vízszintes vonal
 - •
 - Sortörés
 - <p> - elem segítségével a szöveget bekezdésekre tördelhetjük
 - - tag a soron belüli elemek (pl. bekezdésen belüli szavak, szócsoporthoz, mondatok stb.) formázásánál lehet jó
 - <a>
 - href="" A hiperlinkekkel hozhatunk létre kapcsolatot szövegrészek, illetve dokumentumok között. A böngészők a linkeket aláhúzással illetve eltérő színnel jelölik, persze ezek a beállítások megváltoztathatóak.
 - - kép megjelenítése src kép helye, alt a kép leírása
 - - egyszerű felsorolás
 -
 - - sorszámozott felsorolás
 -
 - <dl> - meghatározás lista
 - <dt> - 1 fogalom
 - <dd> - 1 fogalom megmagyarázása
 - <div> - blokk definiálása
 - <style> - stílus adatok megadása html en belül
- id - egy egyedi azonosító, amivel egyértelműen meghatározható, hogy melyik konkrét elemre hivatkozunk a html-ben.
 - Kommentek - <!-- Ide írd a kommented --> :)))

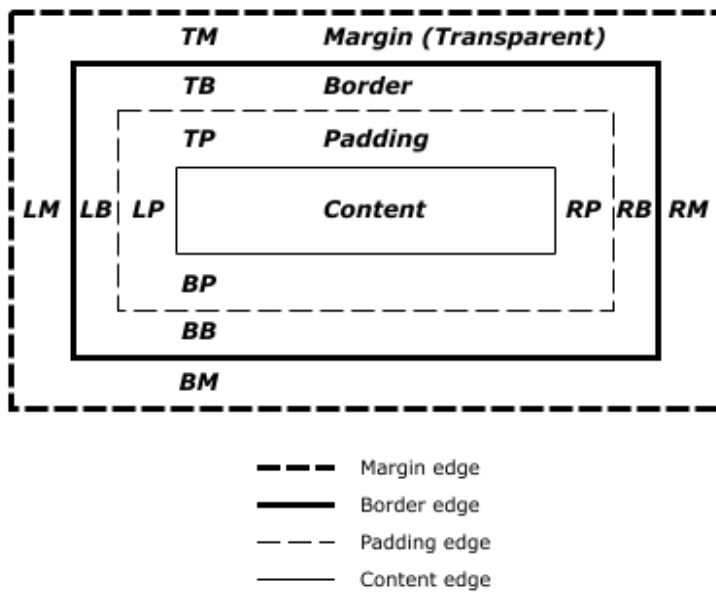
Week-2 – Css

- CSS - „cascading style sheets” kifejezés rövidítése, ami magyarul „egymásba ágyazott stíluslapokat” jelent. A hangsúly a „stíluson” van – míg a HTML a weblap szerkezetét határozza meg (főcímek, bekezdések, stb.), és lehetővé teszi, hogy különböző elemeket (képek, videók) ágyazz webes dokumentumba, addig a CSS a weblap vizuális stílusáért felel – az oldal elrendezéséért, a színekért, a betűkészletekért, azok méretéért, és így tovább.
- `<link rel="stylesheet" href="fileName.css">` css hozzárendelése a html hez a headben
- Alapértelmezett Böngésző Stílusok - Ezek olyan stílusok, amelyek bármely más stílusinformáció hiányában diktálják a HTML elemek megjelenését és érzését. A mester stíluslap használatával törölheti az alapértelmezett böngésző beállításokat.
- Kiválasztó (selector) – A HTML-elemek kiválasztására szolgál, melyekhez stílust rendelhetünk.
- Szabály (rule) - A CSS szabály egy vagy több CSS property csoportja, ami egy vagy több HTML element-re alkalmazandó. Egy selectort és propertiket tartalmaz. A selector határozza meg, amelyikre a CSS rule vonatkozik. A property határozza meg a stílust. Pl.:
- `div { border : 1px solid black;`

`font-size : 18px; }`

CSS konfliktusok - ha ugyanarra az elemre mutat egy vagy több rule

- Lépcsőzetes rendszer (cascading) - a stílusok lépcsőzetesen adhatók meg: böngésző alap stílusa, külső stíluslap, dokumentum szintű stíluslap, HTML elem szinten megadott stílus
- Öröklődés (inheritance) - A HTML elemek a stílustulajdonságaik értékeit megörökölhetik az őket tartalmazó HTML elemtől.
- Gyermek (child) elem - Az öröklődési sorrendben a parrent után következő elem
- Unoka (grandchild) elem - Az öröklődési sorrendben a gyermek után következő elem
- Szülő (parent) elem - Olyan HTML elem, amely tartalmaz további elemeket.
- Nagyszülő (grandparent) elem - Olyan HTML elem, amely tartalmaz szülő elemeket
- Leszármazó kiválasztó
- Doboz modell (box model) : A HTML dokumentumokban minden téglalap alakú. Minden ilyen doboznak van:
 - margin (margó)
 - border (keret)
 - padding (kitöltés)
 - Elem
 - width (szélesség)
 - height (magasság)



RGB Calculator



`rgb(255, 0, 0)`

`#ff0000`

`hsl(0, 100%, 50%)`

R:

G:

B:

RGB Calculator



`rgb(0, 255, 0)`

`#00ff00`


`hsl(120, 100%, 50%)`

R:


G:


B:


RGB Calculator



`rgb(0, 0, 255)`
`#0000ff`
`hsl(240, 100%, 50%)`

R: 

G: 

B: 

RGB Calculator



`rgb(255, 255, 255)`
`#ffffff`
`hsl(0, 0%, 100%)`

R: 

G: 


B: 

RGB Calculator



`rgb(0, 0, 0)`
`#000000`
`hsl(0, 0%, 0%)`

R: 

G: 

B: 

HTML 4: a VGA-16 színszabvány 16 színe és a nekik megfelelő hexadecimális kód

black	#000000	silver	#C0C0C0
maroon	#800000	red	#FF0000
green	#008000	lime	#00FF00
olive	#808000	yellow	#FFFF00
navy	#000080	blue	#0000FF
purple	#800080	fuchsia	#FF00FF
teal	#008080	aqua	#00FFFF
gray	#808080	white	#FFFFFF

Week-3 /Week-4- A Struktúrált Programozás Alapjai

1.Parancssor&Git

Tananyag Ellenőrzés

- cd - Könyvtár váltás change directory
- . and .. - .=aktuális könyvtár, ..=szülő könyvtár parent directory
- pwd - aktuális könyvtár a teljes elérési úttal print work directory
- ls - könyvtár tartalmának listázása list
- touch - fájl létrehozás
- mkdir - könyvtár létrehozás make driectory
- cp - fájl/könyvtár másolása copy
- mv - fájl/könyvtár mozgatása, átnevezése move
- rm - fájl/könyvtár törlése remove
- ``>és>>` és `|` - parancs kimenetének átirányítása and
 - > parancs kimenetének fájlba irányítása
 - >> parancs kimenetének a fájlba írása, hozzáadás a meglévő tartalomhoz
 - | pipe, parancsok összefűzése
- echo - kiíratás
- sort - parancs eredményének sorba rendezése
- uniq - parancs eredményéből csak a különböző elemeket írja ki
- grep - parancs eredményében szűrés kifejezésre
- cat - szöveges fájl olvasása
- git
 - init - repository létrehozása
 - clone - repoizitory klónozása helyi könyvtárba (local working copy)
 - status - változások kiíratása

- diff - módosított és eredeti fájl közötti különbség megjelenítése
- add - változások hozzáadása a lokális working directoryban a staging területhez
- commit - változások mentése a lokális working directoryhoz
- push - becommitolt változások hozzáadása távoli repóhoz
- fetch - letölti a változásokat a távoli repóból
- pull - letölti a változásokat a távoli repóból és összemergeli a lokális változattal
- .git könyvtár - minden olyan adatot tartalmaz, ami a verziókezeléshez kell: commitok, változások, stb.
- .gitignore - minden olyan könyvtárat, fájlt tartalmaz, amit a git-nek nem kell verziókövetni, csak lokálisan fognak létezni, nem kerülnek fel a távoli repóba

Alap utasítások/ Nyisd meg a bash parancssort

- Írasd ki a jelenlegi könyvtár elérési útját - pwd
- Írasd ki a fájlokat és könyvtárakat a jelenlegi mappában - ls
- Hozz létre egy greenfox nevű könyvtárat - mkdir greenfox
- Mozogj a greenfox könyvtárba - cd greenfox
- (Elmozgás nélkül) írasd ki a szülő könyvtár tartalmát - ls ..
- Hozz létre egy üres index.html fájlt - touch index.html
- Hozz létre egy images nevű könyvtárat - mkdir images
- Hozz létre egy css nevű könyvtárat - mkdir css
- Mozogj a images könyvtárba - cd images
- Mozogj vissza a szülő könyvtárba - cd ..
- Hozz létre legalább egy fájlt minden könyvtárban
- Hozz létre egy first-task könyvtárat a greenfox könyvtárban
- Mozgass be minden más fájlt és könyvtárat a first-task könyvtárba
- Innentől dolgozz mindig a greenfox mappában

Fájl műveletek

- Lépj be az újonnan létrehozott repository-ba
- Listázd ki a részletes információkat a benne található fájlokról és könyvtárakról
- Másold le a index.html fájlt about.html néven
- Hozz létre egy temp_images nevű könyvtárat
- Másold bele a 1.jpg és 2.jpg fájlokat a temp_images könyvtárba az images-ből
- Mozdasd a 6.jpg fájlt az images könyvtárba a css-ből
- Töröld ki a 7.jpg fájlt a css-ből
- Commit-old és Push-old a változtatásaidat

Átírányítás

- Írj szöveges tartalmat egy fájlba - echo "blabla" >valami.txt
- Irányítsd át a fájl tartalmát egy másikba - cat valami.txt > másik.txt
- Írasd ki egy fájl sorba rakott tartalmát - cat valami.txt
- Írd át egy fájlba egy másik fájl sorrendbe rakott tartalmát - sort valami.txt > másik.txt
- Szedd ki az ugyanolyan sorokat a tasks.txt fájlból - uniq tasks.txt
- Írd ki az összes sorát a tasks.txt fájlban ami tartalmazza a basic szót - cat tasks.txt | grep basic
- Commit-old és Push-old a változtatásaidat

2. Kifejezések és vezérlési szerkezetek

Áttekintés

- Mit jelent az a fogalom, hogy programozási nyelv? - Ember által olvasható, értelmezhető utasítás sorozat, amivel a számítógépet utasítjuk valamilyen feladat végrehajtására.
- Milyen típusú programozási nyelv a java? - Általános célú objektumorientált nyelv
- Mi kell ahhoz, hogy programot írjunk? - egy szövegszerkesztő, a fordításhoz fordító
- Mi az az utasítás (utasítás)? - a program utasítások sorozata, elemi feladatok végrehajtása
- Mi az a változó? - egy megcímzett memóriaterület, névvel, típussal, értékkel
- Mi az a változó deklaráció? - változó típusának és nevének megadása
- Mi az, hogy változóhoz való hozzárendelés? - érték hozzárendelése a változóhoz, érték adása
- Milyen változó típusokat ismerünk? - boolean, int, float, double, char, string
- Mi az az operator? - műveleteket írnak le
- Milyen típusú operátorokat ismertek? - aritmetikai, relációs, értékadó, logikai, feltételes, inkrementáló, bitenkénti
- Mi az a Hello World! applikáció?
- Mi az a String összefűzés? - Stringek egymás utáni megjelenítése + operátorral
- Mi az a blokk? - egy logikai kapcsolatba tartozó kódsorok
- Mi az az if utasítás? - feltételes elágazás
- Mi az a for ciklus utasítás? - iteráció egy megadott értéktől, megadott feltétel, ameddig fut, megadott értékig iterálva
- Mi az a while ciklus utasítás? - elől tesztelő ciklus, akkor lép be a ciklusba és addig iterál, amíg a feltétel teljesül
- Mi az a do ciklus utasítás? - hátul tesztelő ciklus, egyszer biztosan belép a ciklusba, addig fut, amíg a megadott feltétel teljesül
- Mi az a switch utasítás? - elágazó utasítás, a megadott feltételeknek teljesülése esetén belép az adott ágba és végrehajtja az ottani utasításokat.
- Hogyan tudsz a felhasználtól a parancssorból inputot kérni? - scanner object használatával

3. Tömbök

Áttekintés

- **Mi az a tömb? Milyen tulajdonságaik) vannak?**

A tömb egy olyan változó, amely több azonos típusú adatot tartalmaz. A tömb hossza a létrehozáskor dől el, és attól kezdve a tömb egy állandó méretű adatszerkezet.

A tömb méretét mindenképpen azelőtt kell megadni, mielőtt használni szeretnénk. Fontos azt is tudni, hogy habár a tömb méretének megadásakor a tömbbe mi még nem helyeztünk el elemeket, a tömb nem üres. A méret megadásának kulcsa a new operátor.

Ami viszont fontos: a tömb mérete csak egyszer adható meg, amikor deklaráljuk a tömböt.

Különösen figyelni kell erre akkor, amikor nem tudod, hogy hány elemet szeretnél tárolni, akkor kénytelen vagy az elméleti maximális méretet beállítani, amit a feladat ad meg.

- **Hogyan deklaráljuk őket?**

A tömböt logikailag ugyanúgy kell deklarálni, mint egy egyszerű változót. Megadjuk a típusát és nevét.

```
<elemtípus>[] <tömbAzonosító>;
```

vagy

```
<elemtípus> []<tömbAzonosító>;
```

vagy

```
<elemtípus> <tömbAzonosító>[];
```

A tömb méretének megadása megtörténhet közvetlenül a deklarációkor. Ezt akkor célszerű így használni, ha már ekkor tudod, hogy hány elemet szeretnél tárolni benne:

```
int[] tomb = new int[10];
```

Az is előfordulhat, hogy szükségem lesz egy egészeket tartalmazó tömbre, de még nem ismert számomra, hány darab elemet szeretnék tárolni. Jellemzően ez fájlbeolvasáskor fordul elő, esetleg kiválogatás, szétválogatás témakörben, melyeket majd később ismertetek. Ilyenkor a tömb deklarálása után tetszőleges programkódok lehetnek, melyek nem használják még a tömböt, nem is használhatják, hiszen nincs mérete. Maga a tömb már név szerint létezik, de még nem foglalt le neki a rendszer memóriát az elemek tárolásához. Ellenben a méret megadása előtt megszámlálhatom, hogy majd mekkora tömbre lesz szükségem, és akkor állítom be a méretet, ha már biztosan tudom mekkorára van szükségem.

```
int[] tomb;
```

```
// ...
```

```
// ...
```

```
tomb = new int[10];
```

- **Hogyan adunk értékeket egy tömbnek?**

A tömböt logikailag ugyanúgy kell deklarálni, mint egy egyszerű változót. Megadjuk a típusát és nevét.

```
<elemtípus>[] <tömbAzonosító>;
```

vagy

```
<elemtípus> []<tömbAzonosító>;
```

vagy

```
<elemtípus> <tömbAzonosító>[];
```

A tömb méretének megadása megtörténhet közvetlenül a deklarációkor. Ezt akkor célszerű így használni, ha már ekkor tudod, hogy hány elemet szeretnél tárolni benne:

```
int[] tomb = new int[10];
```

Az is előfordulhat, hogy szükségem lesz egy egészeket tartalmazó tömbre, de még nem ismert számomra, hány darab elemet szeretnék tárolni. Jellemzően ez fájlbeolvasáskor fordul elő, esetleg kiválogatás, szétválogatás témakörben, melyeket majd később ismertetek. Ilyenkor a tömb deklarálása után tetszőleges programkódok lehetnek, melyek nem használják még a tömböt, nem is használhatják, hiszen nincs mérete. Maga a tömb már név szerint létezik, de még nem foglalt le neki a rendszer memóriát az elemek tárolásához. Ellenben a méret megadása előtt megszámlálhatom, hogy majd mekkora tömbre lesz szükségem, és akkor állítom be a méretet, ha már biztosan tudom mekkorára van szükségem.

```
int[] tomb;
```

```
// ...
```

```
// ...
```

```
tomb = new int[10];
```

- **Hogyan adunk értékeket egy tömbnek?**
int[] array = {10, 12, 11};
- **Honnan kezdődik, mi az első *indexe* egy tömbnek?**

A kezdőérték minden elemnél 0, logikai típus esetén pedig false lesz. Később, ha mi magunk helyezünk el bárhová a tömbben egy értéket, csak az adott helyen lévő elem kezdő értékét írjuk át. A tömb new operátorral történő létrehozását – ezáltal kezdő értékekkel való feltöltését – inicializálásnak nevezzük.

- **Hogyan kapjuk meg egy tömb összes elemének számát, a tömb nagyságát?**
<tömbAzonosító>.length
- **Hogyan lehet egy tömb elemein végigmenni (pl.: egyesével végig iterálni)?**
Pl feltölteni az indexekkel

```
for( int i = 0; i < tomb.length; i++ ){
```

```
    System.out.print(tomb[i] + " ");
```

```
}
```

- **Hogyan lehet egy tömb elemeit sorba rendezni?**
Pl:

```
double[] meresek = new double[]{53.27,53.109,52.94};
```

```
Arrays.sort(meresek);
```

- **Mit jelent az, hogy egy tömb többdimenziós?**

Nem csak sora oszlopa is van (jobb válaszom nincs rá). Több indexe is lehet.

```
típus[][]...[] tömbnév;
```

```
típus tömbnév[][]...[];
```

Két dimenziós tömb létrehozására egy példa:

```
típus[][] tömb;
```

```
tömb = new típus[méret1][méret2];
```

Egy konkrét példa kétdimenziós tömb létrehozására és feltöltésére. A tömböt legegyszerűbben egy **for** ciklussal tölthetjük fel.

```
int[][] tomb;
```

```
tomb = new int[10][9];
```

```
for (int i=0; i < tomb.length; i++){  
  
    for (int j = 0; j < tomb[i].length; j++) {  
  
        tomb[i][j] = (i+1)*(j+1)*9;  
  
    }  
  
}
```

A Java lehetővé teszi, hogy a tömböket a definiálás során konstans értékekkel töltsük fel. Ilyenkor a tömböt a fordító hozza létre

```
int[][] matrix = { { 1, 2 }, { 3, 4 } };
```

Workshop

```
public class Arrays {  
    public static void main(String[] args) {  
        int[] array = {1, 2, 3, 6};  
  
        // Írasd ki a tömb második elemét  
        System.out.println("A tömb második eleme: " + array[1]);  
  
        // Írasd ki a tömb minden elemét  
        for (int i = 0; i < array.length; i++) {  
            System.out.println(array[i] + " ");  
        }  
    }  
}
```

4. Függvények

A metódus utasítások összessége, amelyet meghívhatunk a metódus nevére való hivatkozással. Az osztály metódusainak deklarálási sorrendje tetszőleges (a main metódust célszerű az első vagy utolsó metódusként megadni.)

Egy függvénynek nem feltétlenül van értelmezhető visszatérési értéke, bár visszatérési típust mindenképpen meg kell adni. A Java-ban a void típus jelzi azt hogy az adott függvény nem tér vissza semmivel (ez esetben is használhatjuk a return utasítást, érték nélkül). Az ilyen függvényeket tipikusan azok melléghatásáért szoktuk használni, és helyesebb volna eljárásnak nevezni őket, mivel lényegében semmi közük sincs a matematikai értelemben vett függvényekhez.

- **újrahasználhatóság**

többször fel lehet használni ugyanazt a fgv, akárhányszor meg tudod hívni

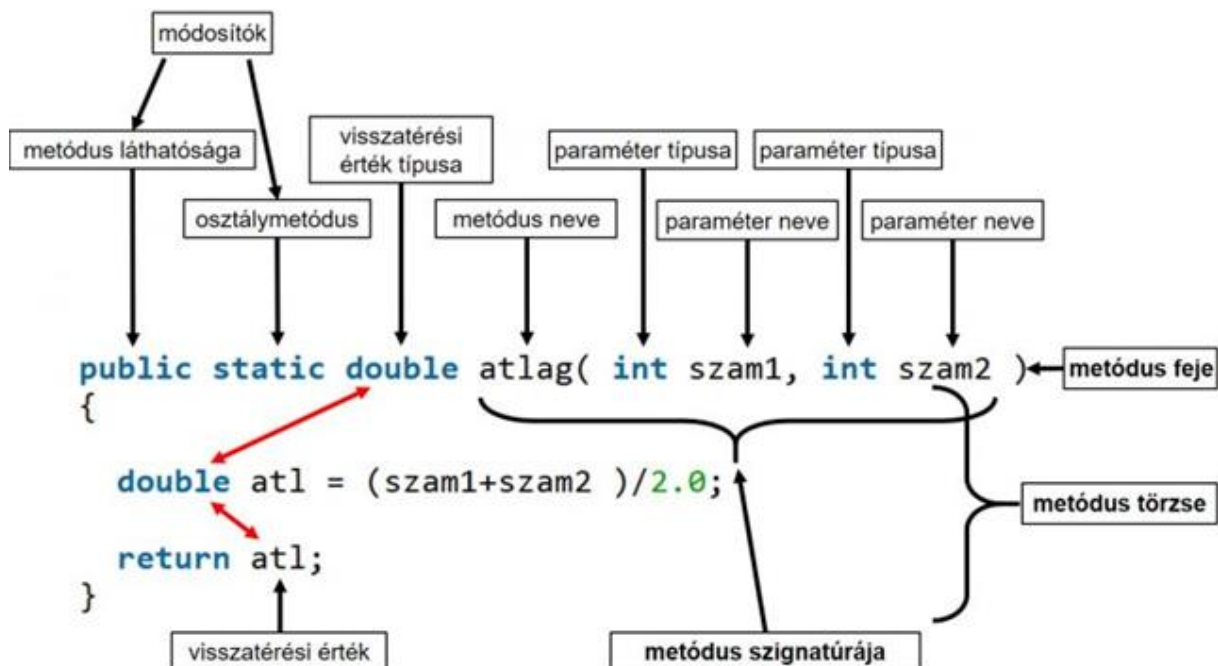
- **kódismétlés csökkentése**

Ha egy utasítás sorozatot többször akarunk felhasználni, v feltehetőleg a későbbiekben használni akarjuk, célszerű metódust rni hozzá gy a későbbiekben a meghívásával csökkentjük a kódismétlést.

- **absztrakció, paraméterezés**

Olyan függvény az absztrakciós metódus, aminek nincs kimeneti értéke, hanem változókat deklarálunk bennük, hogy később felhasználhatók legyenek akár máshol is, meghívással

paraméterezés:



A metódusok különféle részeit a fent felsorolt példákon keresztül mutatnám be:

public: A metódusoknál különféle módosítókat, például hozzáférési szintet adhatunk meg, hogy az adott metódust milyen kód használhatja. Ebbe most részletesen nem mennék bele, egyelőre fogadjuk el, hogy a public olyan hozzáférést biztosít, hogy akármilyen kód használhatja ezt a metódust. A public kulcsszó akár el is hagyható, de a hiánya is egyfajta jogosultsági szintet jelent.

Privat kívülről nem hozzáférhető

static: Kötelező, ha a main-ben megírt kódjainkat szeretnénk kiegészíteni a metódusokkal. Nélküle a kód szintaktikai hibát tartalmaz. A saját objektumokba írt metódusoknál nagyon nem mindegy, hogy használjuk-e a static kulcsszót, vagy sem.

visszatérési érték típusa: A static után meg kell adni, hogy a metódusnak van-e visszatérési értéke, vagy nincs. Ha van, akkor a visszatérési érték típusát kell megadni (3-4 példák), ha nincs, akkor a void kulcsszót kell használni (1-2 példák). Az ábrán vörös nyíllal kiemelttem, hogy ezeknek a típusoknak kötelezően egyeznie kell!

metódus neve: A metódus neve a névadási szabályoknak megfelelően bármi lehet, de illik olyan nevet adni, ami utal arra, hogy a metódus milyen feladatot lát el. A név jellemzően egyedi, de bizonyos körülmények együttállása esetén azonos nevek is megengedettek, ezekről a felsorolás után írok. A metódusok nevénél is illik használni a camelCase elvet, vagyis több szóból álló nevek esetén az első szó kisbetűvel kezdődik, utána minden szó kezdőbetűje nagybetű.

zárójelek: Kötelező formai elemek, de csak akkor kell szerepeltetni benne valamit, ha a metódus paramétert vagy paramétereket kap.

paraméterek: Amennyiben szükség van rájuk, minden paramétert típussal és névvel kell megadni a zárójelen belül, vesszőkkel elválasztva. A paraméter típusának a hozzátartozó értékhez passzolnia kell, neve a névadási szabályoknak megfelelően bármi lehet. A kapott paraméter hatóköre a metódus blokkjára korlátozódik, csak a metóduson belül értelmezett. A változó értékét csak a metódusban lehet használni. Más metódusokban is használhatjuk ugyanezt a változó nevet, ezek között nincs átfedés, nincs hatásuk egymásra. A kapott paraméter neve meg is egyezhet a metódus nevével, de szerintem az egyezést célszerű kerülni.

blokk: A metódusnak egy speciális esetet leszámítva kötelező blokkot nyitni. A blokkba írjuk meg azokat az utasításokat, amelyek a feladatot végrehajtják, vagy előállítják az eredményt.

return: Amennyiben a metódusnak van visszatérési értéke, a blokknak mindenképpen tartalmaznia kell legalább egy return szóval kezdődő utasítást, ami után szerepelnie kell annak a változónak, értékének, kifejezésnek, amely az eredményt határozza meg. Ha a blokkon belül több feltételhez kötött ág is található, biztosítani kell, hogy return utasítással mindenképpen találkozzon a végrehajtás. Visszatérési érték nélküli (void) típusú metódus esetén is használható return utasítás. Ilyenkor a return után csak a lezáró pontosvessző állhat. A szerepe az, hogy a metódus végrehajtása azonnal véget ér, és a return utáni utasítások végrehajtása nem történik meg. A vezérlés ebben az esetben azonnal visszakérül a metódust hívó utasításhoz.

A metódusnak odaadott paraméterek száma tetszőleges lehet. Ha így nézzük a dolgot, akkor a már négyféle alkalmazási módnál járunk:

1. Egy feladat végrehajtását kérjük tőle visszatérési érték nélkül, paraméterek nélkül.
2. Egy feladat végrehajtását kérjük tőle visszatérési érték nélkül, paraméterek segítségével.
3. Egy visszatérési értéket kérünk tőle paraméterek nélkül.
4. Egy visszatérési értéket tőle paraméterek segítségével.

- **void vagy return (érték visszaküldés)**
- **használható a kód szétválasztására (olvashatóvá tételére)**

return visszatérési értéke, void típus jelzi azt hogy az adott függvény nem tér vissza semmivel

- **saját változókkal rendelkeznek**

Összességében rögzítsük a következőket:

Minden primitív változó (egész, valós, char és boolean) minden esetben érték szerint kerül átadásra metódus híváskor. Minden egyéb adatszerkezet (tömb, lista, objektumok, stb) pedig referenciaként, vagyis a változóban tárolt címként kerül átadásra. Ezeknél az adatszerkezeteknél, ha a metódus módosítja a kapott referencia típusú adatszerkezetet, akkor annak tartalma a metódust meghívó helyen is megváltozik!

- **paraméterek használata**

lehet-e egy függvénynek bemeneti paramétere? igen

- **Debuggolás**

- **töréspont** (breakpoint) sorok száma mellett, rákattintani - megjelenik egy piros telt kör, a futtatásnál itt fog megállni, útközben kiírja a részeredményeket
- **léptetés** (step over) F8-al - tovább halad, ha ciklusban van a töréspont, abban forog végig
- **megfigyelések** (watches) itt összesíti az eredményt, hasznos főleg ha több 100 oldalas a program.
- **állapot elemzése** [21:45] ilyen felületesen

.5 Adatstruktúrák

Tananyag áttekintés

- **ArrayList**

- **Object**
- *add* és *remove* jelentése
- példányosítás (initialize)
- `.add()` (a következő pozícióba, egy indexre)
- `.addAll()`
- `.remove()` (index vagy érték (value) által)
- `.set()`
- `.size()` méret
- `.isEmpty()`
- `.clear()`
- keresés: `.contains()`, `.containsAll()`
- tömb közötti váltás (*másolás* `.toArray()`, `.addAll()`)
- átalakítás Stringgé `.toString()`

- **String**

- `.length()` hossza
- `.substring()`
- `.replace()`
- `.charAt()`
- `.equals()`
- `.compareTo()`
- `.toLowerCase()` kisbetűs

- `.contains()` tartalmazza
- `.endsWith()` végződik
- `.startsWith()` kezdődik
- `.indexOf()`
- `.lastIndexOf()`
- `.split()`
- **HashMap**
 - kulcs (key)
 - érték (value)
 - `.put()`
 - `.get()`

Week-5

1 Szerver Hardware Infrastruktúrák

Material Review

- Megismerni a szerver elhelyezésének környezetét, elvárásokat, az alábbi szinteken
 - gépterem helyszínével szembeni elvárások
 - lokalizációs megfontolások
 - biztonsági követelmények
 - elektromos áramellátás
 - klimatizálás
 - kábelezés
- Megvizsgáljuk a rack szekrény felépítését, beépítési szabályokat
- Áttekintjük a szerver architektúrákat és kiegészítő eszközöket
 - központi tárolók
 - hálózati eszközök
 - szerver architektúrákat

Workshop

Keress utána, hogy milyen gépterem besorolási szintek vannak! Készíts összehasonlító táblázatot!

TIER besorolás

Fogalom definíció:

Az adatközpontokat rendelkezésre állás szempontjából legsikeresebben az Uptime Institute TIER ajánlása osztályozta a következő 4 csoportba:

- Tier 1: Alapinfrastruktúra egyirányú energiaellátással, redundancia nélkül;
- Tier 2: Az infrastruktúra egyes elemei tartalékkal rendelkeznek;

- Tier 3: Az ICT rendszerek leállása nélkül karbantartható infrastruktúra;
- Tier 4: Hibatűrő infrastruktúra, bármely elem hibája esetén is képes ellátni az összes ICT eszközt.

A gépterem egyes alkotó elemeinek (falazat, klíma, UPS, távközlési rendszerek stb.) osztályba sorolása lehet különböző, de ebben az esetben a létesítményt a "leggyengébb láncszemnek" megfelelő osztályba sorolják.

	TIER I	TIER II	TIER III	TIER IV
Aktív kiszolgáló egységek az ICT eszközök ellátására	N	N+1	N+1	bármely elem hibája után is ,N‘
Ellátási útvonal	1	1	1 aktív, 1 tartalék	2 egyidejűleg aktív
Szolgáltatás-kiesés nélkül karbantartható	Nem	Nem	Igen	Igen
Hibatűrő	Nem	Nem	Nem	Igen
Független ellátási útvonalak	Nem	Nem	Nem	Igen
Folyamatos hűtés	Terhelésfüggő	Terhelésfüggő	Terhelésfüggő	Biztosított
Rendelkezésre állás	99.67%	99.75%	99.98%	99.99%
Éves tervezett leaállás és kiesett idő	28.8 óra	22 óra	1.6 óra	0.8 óra

1. táblázat: TIER besorolások összefoglalása az Uptime Institute 2010-es definíciója alapján

Mi a különbség egy Tier 3 és Tier 4 besorolású gépterem között?

lásd táblázatban. Tier 3: Az ICT rendszerek leállása nélkül karbantartható infrastruktúra; Tier 4: Hibatűrő infrastruktúra, bármely elem hibája esetén is képes ellátni az összes ICT eszközt.

Keress Tier 3 és Tier 4 géptermekeket Budapesten!

Tier 3 Invitech Solution Kft – H1 Solution üzemelteti DC 10 – (Tier3), 11, 13

Lehet-e gépteremben vízzel oltani? Indokold!

Oltó berendezés kiválasztása, kialakítása esetében figyelni kell arra, hogy egyrészt elektromos tűz oltására is alkalmasnak kell lennie a rendszernek, valamint arra, hogy ne okozzon további károkat az IT infrastruktúra aktív és passzív elemeiben. Fontos, hogy automatikusan lépjen működésbe (tűz és füstérzékelő aktiválja), valamint legyen lehetőség késleltetni és/vagy felfüggeszteni az oltást, mert a gépteremben rekedt személy kimentését előbb meg kell oldani.

- elektromos berendezések vízzel való oltása tilos és életveszélyes,

Milyen oltórendszerek/készülékek használhatók gépteremben?

Gázzal oltó rendszer beépített tűzjelző és füstjelző rendszerrel.

Mi a különbség egy „rack mounted” és egy „blade” szerver között?

Megfigyelhető, hogy mivel a gépterem kialakítása, üzemeltetése vagy bérlete magas költségekkel jár, ezért a cél a rendelkezésre álló hely minél hatékonyabb kihasználása. Mivel a rack szekrények száma egy adott területen véges, valamint a rack szekrények magasság is kötött, ezért csak a gép „sűrűség” növelésével lehet a költségnövekedést elkerülni. Történelmileg így alakult át a tower szerverek rack mounted szerverekké, majd lettek egyre „alacsonyabbak” a szerverek. Egy adott szint alá (1U) azonban ez nem csökkenthető. Így a szerverek számának növeléséhez adott U helyen más módon kellett megoldani (fizikai szintű). Ez az igény hozta létre a blade technológiát.

Blade technológia esetén a rack szekrénybe beépítésre kerül egy blade keret (enclosure vagy frame). Ez 10U magas. A blade keretbe speciális szerverek: pengék építhetők be 12 db. Ez azt jelenti, hogy 10U helyre 12U helyigényű szerverparkot helyeztünk el. Ennek a megoldásnak a lényege, hogy a szerverek egyes alkatrészeit átintegráljuk a keretbe. 10U helyre 12U helyigényű szerverparkot helyeztünk el. Ennek a megoldásnak a lényege, hogy a szerverek egyes alkatrészeit átintegráljuk a keretbe.

7. Mi a különbség egy desktop gép és egy szerver között?

Munkaállomások esetében alapvetően hiányoznak a redundáns kiépítések, azaz a gépbe 1 CPU, 1 táp, több disk de azok nem RAID-be szervezve kerülnek beépítésre. Alaplap hőterhelésre való tervezése sem a fokozott igénybevételre van kialakítva. Valamint az egyes fő alkatrészek hibatűrése is alacsonyabb szintű. Ennek megfelelően egy alkatrész meghibásodása azonnali kiesést eredményez.

Munkaállomások hardware szinten nem rendelkeznek olyan mélységű szenzorokon alapú monitoring rendszerrel, amelyek képesek lehetnek (ha nem is minden esetben) előre jelezni a közel jövőben fellépő meghibásodásokat.

Továbbá, munkaállomás nem tartalmaz olyan management modult (HPE: iLO; DELL: iDRAC; IBM: IMM), amivel a távoli menedzsmentje megoldható lenne (az operációs rendszertől függetlenül). Azaz a gép csak akkor érhető el távolról, ha az operációs rendszer elindult, funkcionalitásának a távoli elérést

biztosító része fut. Amennyiben nem elérhető OS-en keresztül a gép, akkor azt csak fizikailag a konzolról lehet újra üzemképes állapotba hozni.

8. Oktatóanyag a HPE blade technológián keresztül mutatta be blade rendszer felépítését. Keresd meg ennek a megfelelőjét a DELL esetében!

Dell PowerEdge

2.Network Alapismeretek Számítógép hálózatok

Material Review

- **A hálózatok célja, alkalmazása, alapfogalmak**

Számítógéprendszerek valamilyen információ átvitelrel megvalósítható cél érdekében történő (hardveres és szoftveres) összekapcsolása. A kapcsolat lehet közvetlen (egy kábellel kialakított), vagy közvetett (a hálózati kapcsolat kialakítását segítő eszközökön, például modemeken keresztül).

A számítógép-hálózat tipikusan számítógépekből és perifériás elemekből (pl. hálózati nyomtató), hálózati kapcsolóelemekből, a fizikai összeköttetést megvalósító eszközökből (kábelekből) és a különböző hálózati alkalmazásokat megvalósító programokból (szoftverekből) épül fel.

Alapfogalmak: Switch, Hub.

- **Hálózati struktúrák**

PAN (Personal Area Network) – személyes hálózat. Max néhány méter. segítségével egy ember környezetében levő eszközök kommunikálhatnak egymással. Jellemző példa az olyan vezeték nélküli hálózat, mely egy számítógépet köt össze a perifériáival.

LAN (Local Area Network) - kis kiterjedésű hálózat, lokális hálózat. A LAN olyan magánhálózat, amely egyetlen épületen belül vagy annak környezetében üzemel, például egy lakásban, irodában vagy gyártelepen. Széles körben használják ezeket személyi számítógépek, valamint fogyasztói elektronikus eszközök összekapcsolására, lehetővé téve ezzel a közös erőforrások (például nyomtatók) megosztását és az információcserét. Amikor nagy cégek használnak LAN-okat, **vállalati hálózatokról** beszélünk (**enterprise network**). Jellemzője az egyedi kábelezés, gyors adatátvitel. Kiterjedtsége az 1 szobától néhány kilométerig terjed.

MAN (Metropolitan Area Network) - városi méretű hálózat A MAN egész városokat átölelő földrajzi kiterjedéssel rendelkezik, technológiailag mégis a LAN-hoz áll közelebb.

WAN (Wide Area Network) - nagytávolságú hálózat

-
-

- **Fizikai átviteli jellemzők és módszerek**

- multiplexelés frekvenciaosztással, multiplexelés szinkron idő beosztással, vonalkapcsolás, üzenet és csomagkapcsolás

- **Vezetékes átviteli közegek**

- csavart érpár (UTP, STP), koaxiális kábelek, alapsávú koaxiális kábelek, szélessávú koaxiális kábelek,

- **Vezeték nélküli átviteli közegek**

- infravörös lézer átvitel, rádióhullám, szórt spektrumú sugárzás, műholdas átvitel,

- **Lokális hálózatok**

- **TCP/IP protokoll és az internet** - A TCP/IP felépítése a rétegződési elven alapul, minden egyes réteg egy jól definiált feladatot végez el, és a rétegek egymás között szolgálatelérési pontokon keresztül kommunikálnak. Minden réteg csak a vele szomszédos réteggel képes kommunikálni, mivel ezek egymásra épülnek. Alapvetően négy réteg alkotta, melyet ötre bővítettek.

- **Mik a privát IP tartományok?** - Lokális, magán hálózat elérhető IP cím tartománya

- **Mi a subnet maszk?** - Elválasztja a host és network IP-t

- **Mi a default gateway?** - Két hálózat közötti átjárást biztosítja.

- **Mi a különbség az OSI és a TCP/IP között?** OSI funkciók alapján vannak a rétegek, TCP/IP protokollok alapján vannak a rétegek.

- **Mi a leglényegesebb különbség a TCP és az UDP között?** - UDP-nél nincs handshake és acknowledgement

- **Mi az APIPA?** - Automatic Private IP Addressing Windows-ban lehetővé teszi az automatikus IP hozzárendelést

- **Mi a localhost és hogyan érem el?** - a saját gépre vonatkozó host név

- **Mik a well known portok?** - HTTP 80 Web HTTPS 443 Web (secure) FTP 20,21 File transfer SFTP 22 File transfer (secure) FTPS 989,990 File transfer (secure) SIP 5060 VoIP (Internet phone) DNS 53 Find IP address SMTP 25 Internet mail POP3 110 POP mailbox IMAP 143 IMAP mailbox Telnet 23 Remote login SSH 22 Remote login (secure)

- **A TCP/IP modell szerint melyik rétegbe tartozik az IP (Internet Protocol)?** - hálózati réteg

- **A TCP/IP modell szerint melyik rétegbe tartozik a TCP (Transmission Control Protocol)?** - transport réteg

- **Mi a HTTP port száma?** - 8080

- **Mire jó a ping parancs?** - egy adott host elérhetőségét lehet vele vizsgálni a hálózaton

Workshop

- **Nézzünk utána mi a különbség az ADSL és a bérelt vonal között? Melyiket használják enterprise környezetben és miért?**

Adsl – Aszimmetrikus a vonal (letöltési sávszélesség nagyobb, mint a feltöltési, egyes userek között el van osztva a teljes sávszélesség, minél többen töltenek le kevesebb sávszélesség jut egy emberre, ezért van egy lehetséges sávszélesség.) Bérelt vonal:garantáltan a felhasználónak egy minimum, hogy a szolgáltató adja. Céges-enterprise a bérelt vonalat fogják használni, ami egy sávszélességet garantál.

A szolgáltatás jellemzői:

- Széles sávszélesség tartomány 1 Mbps-10 Gbps
- Réz, optika és mikros hozzáférés, igény esetén redundáns kialakítással
- Rugalmas sávszélesség-módosítási lehetőség
- Fix IP-cím és igény esetén IP-cím tartomány
- 24 órás monitorozás
- Magas rendelkezésre állás

- **Kérdezzük le az otthoni gépünk IP beállításait. Milyen osztályú IP címmel rendelkezik?**

cmd: ipconfig

- **Hogyan kérhetnénk magunknak új IP címet?**

cmd: ipconfig /renew

- **Kérdezzük le az otthoni gépünk ARP tábláját**

cmd: arp -a

- **Milyen MX rekordot/rekordokat használunk?**

Mail X changer record, egy adott mail szervernek az Ip címét mutatja meg. Ilyen szolgáltatás érhető el a Mx toolbox.com-on. Pl. mail server: freemail.hu

- **Milyen parancssal kérdeznéd le, hogy a géped milyen TCP/UDP portokon figyel éppen (aktív kapcsolatok)?**

cmd: netstat -a

- **Milyen parancsot használnál, ha csak! a géped MAC címét szeretnéd kiíratni (Windows parancsosrban)**

cmd: getmac

- **Mire szolgál a tracert parancs? Mikor és mire használnád?**

Megmutatja, milyen network útvonalon érhető el egy adott host. Pl.: tracert freemail.hu

●

- **Hálózatunkon nincs engedélyezve a 3389-es port. Milyen szolgáltatás nem fog működni?**

RDP-n keresztül nem lehet elérni az adott windows hostot. Remote Desktop Protoll , ezen keresztül érhető el egy távoli windows szerver felhasználói felülete.

3.San, storage infrastruktúra alapok

Material Review

Ebben a fejezetben megismerkedünk néhány szerver architektúrát érintő további eszközzel:

- tárolók
- SAN, SAN hálózat

Tárolók, storage-ok: A storage feladata, hogy egy teljesen szeparált módon, és esetekben centralizált módon biztosítsa más szervereknek, vagy klienseknek a tárterületet.

Tároló típusok: DAS, NAS, SAN

DAS

Direct Attached Storage: közvetlenül kapcsolt tároló megoldást jelent. ugyan azt vagy ahhoz nagyon hasonló, közeli technológiát és kábelezést használjuk a két "doboz" (tároló - szerver) összekötésére, mint amit a gépen belül a diszkek, lemezek csatlakoztatására használunk. Ebben az esetben a kábel hossza elég rövid 3-4 méter hosszú lehet átlagosan. pl.: USB

- **előnyök:** egyszerű a bevezetése, olcsó, különösebb konfigurációt nem igényel, csak össze kell kötnünk a megfelelő kábeleket
- **hátrányok:** nem lehet nagy távolságokra elhelyezni a szerverektől és magát a tárolót, nem nyerünk nagyobb diszk sebességet, tehát lassúnak és már idejétmúlt technológiának mondható, kategorizálható. Általában redundáns kapcsolatokat és redundáns tápellátást sem támogatnak az ilyen eszközök és nem is nagyon skálázhatók, mivel kötött a maximálisan összekapcsolható eszközök száma is.
- **közvetítő közeg:** tipikusan SCSI kábel(vagy még eSata, USB csatlakozás is elképzelhető), de lényegében, ugyan az a kábel, protokoll ami a szerveren belül a belső SCSI felületű diszkeket is összeköti.
- **szolgáltatott tárhely típusa:** nyers diszkeket vagy LUN-okat látathatunk a szerver oldalán, tehát általában ebben a tárolóban nincs semmi extra hozzáadott technológia

NAS

(Network Attached Storage, akár Cloud Data Storage) mint az a diszkekkel teli "doboz", ami tipikusan egyszerű ethernet alapú **TCP/IP** hálózaton keresztül kapcsolódik a szerverhez és / vagy más ilyen tárolókhoz. a **NAS**-nak, mind pedig a server/client-nek szüksége van egy-egy **IP** címre a **NAS** fajtájától függően támogat majd úgynevezett file átviteli protokollokat

- **FTP, WebDAV, SAMBA/Windows megosztás, NFS, AFS**

- egyszerű asztali gépünkől is csinálhatunk **NAS**-t, olyan tárhellyel teli gép, ami mindenféle hálózati megosztásra képes kiszolgálni a szintén hálózatban lévő más gépeket, szervereket.
- **előnyök:** a hálózati kapcsolat miatt egyszerűen és olcsón tudjuk a hálózatba kapcsolni és bevezetni, a kezelése is egyszerű, hisz csak egy számunkra már ismert technológiával kell megosztásokat kezelnünk.
- **hátrányok:** mivel a forgalmat a hálózaton bonyolítjuk, az könnyedén kieshet, vagy belassulhat, tehát átlagosan egy nem túl biztos közvetítő közeg (hacsak nem használunk dedikált vonalat), illetve a megosztások miatt nem kezelhetünk alap esetben a fenti protokollokon keresztül direkt diszkeket.

SAN

(Storage Area Network) A SAN egy topológia, hálózati protokoll is, tehát egy teljes tárolási módszer / technológiát / rendszert takar.

SAN STORAGE: A SAN storage a SAN struktúrában a tényleges doboz amiben diskek vannak, csak kiegészítve azzal, hogy van neki egy **HBA kártyája**, aminek **FC** portjai vannak. Ebből is minimum kettőt érdemes tartani, hogy a **teljes SAN** infrastruktúránk hibátűrő legyen szintenként.

- **Előnyök:** Hihetetlen gyors, nagy távolságok is kialakíthatóak végpontok között, és teljesen hibátűrő struktúrát kínál. Elektromos hatások nem befolyásolják az adatáramlást.
- **Hátrányok:** Drága, speciális kártyát és kábeleket kíván. Sérülékeny, és sokszor komoly szakértelem kell a bevezetéséhez.
- **Közvetítő közeg:** Üvegszál.
- **Szolgáltatott tárhely típusa:** Nyers diskeket vagy LUNokat látathatunk a server/client felé.

RAID Redundant Array of Independent Disks rövidítése, azaz redundáns tömb különálló diskekből.

RAID 0

Diskek összefűzése.

Stripe: Diskeket csíkozzuk, úgy írunk rá adatokat. Ilyenkor fontos, hogy ugyan akkora méretű diskeket használjunk, különben a legkisebb méretű lesz a szűk keresztmetszet.

- Előnye: Gyorsabb, mindegyik disket kvázi egyszerre tudja használni
- Hátránya: Ugyan akkora méretű diskek kellene.

Concat

Diskek egymás után kötése. Ilyenkor vegyesen használhatunk, különböző méretű diskeket.

- Előnye: Vegyesen fűzhetünk össze diskeket és az össz méretet tudjuk kezelni.
- Hátránya: Lassabb, mert folyamatosan írja tele először az első disket, majd a következőt, és így tovább.

Globálisan a következőket mondhatjuk el a RAID 0-ról:

- Több disket kapcsol össze, ezáltal gyorsítva vagy az írási/olvasási sebességet és egyesítve a tárhely kihasználtságot.
- Minimálisan kettő disk szükségesetük hozzá. Maximálisan pedig csak gyakorlati limite lehet, elméleti nem.

RAID 1

Köznapiabb neve a tükrözés. Minimálisan két disk szükségesetük, maximálisan akármennyi lehet, de kettőnél több esetén a többit már SPARE diskként fogja kezelni.

- Kétszer annyi tárterületre van szükség a tükrözés miatt, ami csak költség.
- Ha egyik disk meghal, a másiktól akkor is elérhetőek az adatok.

SPARE DISK

RAID tömbökhöz tudunk SPARE (tartalék) diskeket hozzárendelni. Ezek a diskek nem lesznek használva, csak üresen, használatlanul fognak állni. Egészen addig, amíg egy hibátűrő RAID tömbben egy disk meghibásodik.

RAID 5

A RAID5 egy újragondolása a RAID1 és RAID0-nek. Gyors, és takarékos megoldást akarunk, viszont hibatűrőssel.

Minimálisan három disket kell használnunk, és megjelenik egy új fogalom a Paritás.

A paritás nem más, mint egy ellenőrző összeg. Nem más, mint olyan érték, amit úgy kapunk, meg, hogy az egy csíkon helyezkedő adatok értékét összegezzük. A RAID5 egyetlen paritás bitet használ csíkonként, viszont minden csík esetében másik disken tárolja a paritást.

- Előnye: Gyors, és csupán egy disknyi területet veszünk csak a redundancia miatt, cserébe egy tetszőleges disk kiesését tűri el.
- Hátránya: csíkozással dolgozik, tehát ugyan akkora méretű diskek kellene.

RAID 0+1 ÉS RAID 1+0

Természetesen a RAID5 előtt kitalálták a RAID 0 és RAID 1 ötvöztetését. Mindkettőhöz négy-négy disk kell.

LUN logikai unit: a gazdaságosabb kihasználtság esetén szinte mindig LUN-okat osztunk ki a tárolókból. Ezek a LUN-ok csak egy logikai részek, aminek van nevük, számuk, és méretük.

HIBRID TECHNOLÓGIÁK

kombinálják a fenti technológiákat. Lesz egy eszköz, amiben diszkek lesznek, de annak lesz többféle csatlakozási pontja. Lesz rajta SCSI vagy USB, eSata (DAS), lesz rajta hálózati rezes vagy optikai ethernet port (NAS), és akár csak FiberChannel, FC port is (SAN).

ISCSI

Az iSCSI, azaz **Internet Small Computer Systems Interface**, a hagyományos SCSI tárolási felület kiterjesztése, ami lehetővé teszi a SCSI parancsok továbbítását egy IP alapú hálózaton. Az iSCSI lehetővé teszi a számítógépek számára a merevlemezhez való hozzáférést egy

hálózaton keresztül, azonos eljárással, mint amikor a merevlemez direkt módon kapcsolódik a számítógéphez.

- NAS-okkal ellentétben diszk alapú tárhely kiajánlást tesz lehetővé, a NAS-oknál megszokott ethernet alapú, TCP/IP hálózaton
- az iSCSI egy szoftveres megoldás, nem szükségeltetik hozzá külön speciális hardver, habár léteznek iSCSI képes HBA kártyák. Ilyenkor az átviteli számítási teljesítményt nem a rendszert terheli, hanem egy külön egység végzi magán a kártyán.
- a hálózat terheltsége és vonal megbízhatósága sajnos könnyen befolyásolhatja a diszk elérést

4. Operációs Rendszer alapok/Linux alapok/Windows alapok

Workshop

- Lehet-e grafikus felület nélkül installálni, használni, üzemeltetni MS Windows operációs rendszert?
nem
- Lehet-e grafikus felület nélkül installálni, használni, üzemeltetni Linux operációs rendszert?
igen
- Lehet-e MS Windows Server-en futó alkalmazást futtatni Linux operációs rendszeren?
Nem
- Szükséges-e driver installálása MS Windows OS-ek alatt? igen, de automatikusan történik
- Szükséges-e driver installálása Linux OS-ek alatt? igen
- MS Windows alatt a registry-t milyen alkalmazással lehet menedzselni? - registry editor

Linux operációs rendszer installáció (Centos 8.2)

Workshop

- Milyen gyártói előírások vannak CentOS 8 esetében a hardware-re vonatkozóan (minimal requirements és compatibility)?
 - 2 GB RAM.
 - 2 GHz or Higher Processor.
 - 20 GB Hard Disk.
 - 64-bit x86 **System**.
- Mi a különbség a "yum" és az "rpm" paramncsokkal való installálások között? - YUM feloldja installáláskor a függőségeket, RPM nem oldja fel.
- Mi a feltétele annak, hogy DHCP-s IP cím beállítást használjunk a Linux szerverünk installálása során? - egy DHCP képes router

- Miért térjünk át statikus IP cím használatára? - Az adott host így mindig ugyanazon az IP címen érhető el a hálózatról, e.g. NAS
- Installálás során miért nem kérünk NTP-n keresztüli időszinkronizálást, miért kell kézzel beállítani a dátumot és a pontos időt? - Amíg nincs hálózati réteg beállítva, addig nem lehet elérni hálózaton lévő NTP server-t, ezért kézzel kell beállítani.
- Milyen megszorítások vannak Linux operációs rendszert használó gép esetén a host nevére? - maximum 253 karakter hosszú lehet, a pontokat is beleértve, ASCII betűk, 0-9 számok és “-” tartalmazhat. Ne kezdődjön “-”-vel.
- Lehet-e LVM disk területen a /boot és /boot/efi? - nem
- Lehet-e swap partíció LVM disk területen? - igen
- Milyen file rendszereket használhatunk és használunk? - ext2, ext3, ext4, reiserfs, reiser4, xfs,

Windows Server 2016 install

Material Review

- Milyen szolgáltatások érhetőek el a Server Manager-ből alap esetben? - <https://docs.microsoft.com/en-us/windows-server/administration/server-manager/server-manager>

Workshop

- **Mi az NTP szerver? Miért fontos ennek a beállítása?**

A **hálózati idő protokoll** (angolul *Network Time Protocol*, *NTP*) számítógépes rendszerek óráinak szinkronizálására szolgáló hálózati protokoll.

- **Az NTP TCP vagy UDP porton kommunikál?**

UDP porton zajlik a kommunikáció.

- **Miért fontos az Windows Update használata? Nagyvállalati környezetben milyen módon szokták használni és milyen kritikus pontja vannak a rendszerek frissítésének?**

A rendszerben előforduló felfedezett biztonsági hibákat minél előbb ki kell javítani. Ennek módja a frissítés. Tervezetten, előbb tesztkörnyezetben tesztelik ezeket a frissítéseket, majd ha a környezet megfelelően működik, akkor kerül át az éles szerverre a változás. Fontos a biztonsági mentés frissítés előtt.

-
-
-

- **Hol tudjuk átállítani az elsődleges és másodlagos DNS szervert?**

Network Connections > Hálózati kártya kiválasztása > Properties > Internet Protocol version 4 majd a keret alatt Properties > Itt megadható alul a kívánt DNS szerver IP címe

- **Hol tudnánk a gépünket domain-be léptetni? Milyen jogosultság kell ehhez?**

A rendszerbeállítások alatt tudjuk megváltoztatni a név megváltoztatása alatti résznél, ehhez viszont Domain Administrator jogosultság kell, de nem minden esetben. A User is kaphat Domain jogot.

- **Tiltsuk le a tűzfalon a Remote Desktop használatát**

Windows Firewall > Jobb oldalon Allow an app and feature through Windows Firewall > kikeresni a Remote Desktop sort, majd minden pipát mellőle kivenni.

Jó tanulást ☺