Operációs rendszerek alapjai

2. gyakorlat

Mai gyakorlat

- Otthoni Linux használat
- Ssh és ITK-s szerverek
- Shell script alapok
  - (környezeti) változók
  - string, egész aritmetika
  - adatszerkezetek

Linux telepítése

- Rendes telepítéshez szükséges
  - egy számítógép
  - egy ISO lemezkép-fájl
  - egy formázható pendrive (vagy CD)
- Virtuális géphez szükséges (lehet egyébként Linux-ból telepíteni virtuális Windows-t!)
  - egy számítógép
  - például VirtualBox
  - egy ISO lemezkép-fájl

Linux telepítése - számítógép előkészítése

- Ha megválnál a Windows-tól, akkor készen is vagyunk
- Ha nem állsz készen a Windows nélküli életre
  - készítsd elő a merevlemezed partícióit, ahogy szeretnéd
  - javaslat: a Windows és Linux partíció mellett legyen egy közös, amiben közös fájlok vannak (ez a közös partíció legyen NTFS fájlrendszer, a Windows csak azt tudja használni)
  - kapcsold ki Windows-on a gyors indítást

Linux telepítése - lemezkép-fájl

- Lemezkép-fájl
  - a kedvenc disztribúciód honlapjáról le tudod tölteni
  - ha először telepítesz, akkor érdemes LTS (Long-Term Support) verziót
- Telepítő pendrive
  - fontos a pendrive tárhelyének mérete, például Ubuntu esetén legalább 4 GB kell
  - Rufus programmal formázd a pendrive-t és égesd rá a lemezkép-fájlt

Linux telepítése

- A pendrive behelyezése után indítsd újra a gépet
- Lépj be a BIOS-ba
  - amikor kiír valami olyasmit, hogy “to interrupt normal setup, press Enter”, akkor kell az adott gombot megnyomni
- A bootolható eszközök közül válaszd ki a pendrive-odat
- Csináld, amit a telepítő mond
- Ha bármelyik lépés gondot okoz, akkor keress rá a problémára például azzal az információval kiegészítve, hogy milyen géped van

VirtualBox használata

- A program telepítése és egy lemezkép-fájl beszerzése után csináld azt, amit a User Manual mond
- Telepíthető Windows-ra, Linux-ra és Mac-re is
- A megfelelő lemezkép-fájllal telepíthető vele Windows, Linux és Mac is

PuTTY - ha nem telepítenél

- TTY - teletypewriter
- Ingyenes SSH és Telnet kliens Windows-ra
- Letöltés után tetszőleges szerverrel tudsz SSH kapcsolatot létesíteni

SSH - Secure Shell (RFC 4252-4256)

- Régebbi protokollok titkosítás nélkül kommunikáltak a szerverrel
  - így könnyen lehallgathatók volt az egész kommunikáció (felhasználók, jelszavak, fájlok stb.)
  - például: Telnet, rlogin (remote login), rsh (remote shell)
- TCP/IP protokollt használ alapértelmezetten a 22-es porton
- Adat csomagokra bontása, csomagok tömörítése (opcionális) és titkosítása
- Titkosítás fajtája a szervertől és klienstől függ, általában több lehetőség van
  - Nyilvános kulcsú titkosítás
- Ezen felül egy kapcsolati réteg létrehoz egy vagy több folyamatosan nyitott csatornát
- Lehetőség van kapcsolatok továbbítására is
- Lehetőség van grafikus adatokat is továbbítani, így nem “csak” egy terminál adott (X forwarding opciót kell bekapcsolni, Windows-hoz kell X server)

ITK-s szerverek

- users.itk.ppke.hu
  - web szerver, bárki csinálhat rá weboldalt
  - kb. 5 éve nem frissítettek rajta semmit de vannak 10 éves dolgok is (PHP 5, Python 2.6.6, GCC 4.4.5)
- cortex.itk.ppke.hu
  - ezen érdemes futtatni bármilyen kódot (különösen a hosszan futó programokat)
  - Python 3.4.2, GCC 4.9.2 és alapvetően mindenből frissebb van (bár ezek is gyakran 5+ éves verziók)
- medea.itk.ppke.hu
  - Windows-os profilokat tárolja, elérhető rajta minden fájl
- tempus.itk.ppke.hu
  - Linux-os profilokat tárolja

Szövegszerkesztők

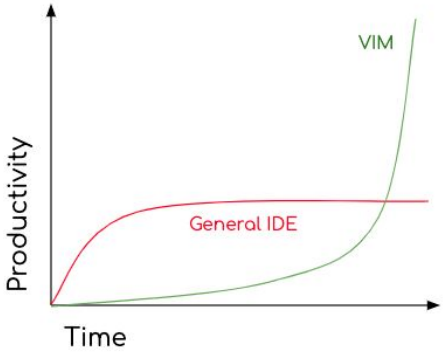
- `gedit`
  - grafikus alapú, szintaxis kiemeléssel
  - C-ben és Python-ban írták
  - GNOME Core Applications része
- `nano`
  - GNU Project része
  - terminál alapú, szintaxis kiemeléssel
  - C-ben írták

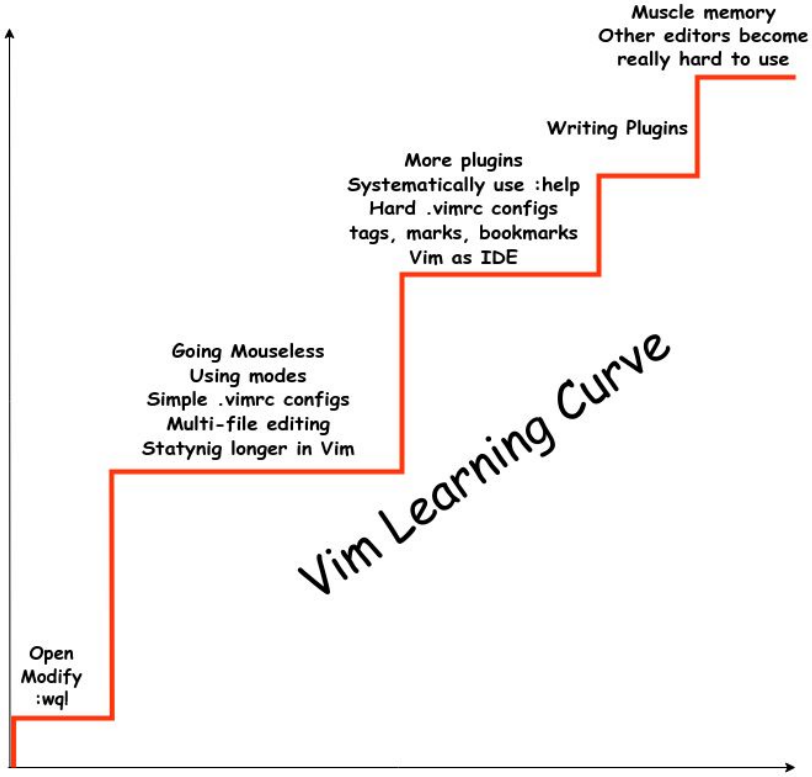
Vim - A szövegszerkesztő

- Terminál alapú, szintaxis kiemeléssel és mindenféle plugin-al
- C-ben és Vim script-ben írták
- Magas testreszabhatóság
- Mindenhol megtalálható (JetBrains, Matlab, Atom, PDF olvasók, böngészők, 9GAG stb.)
- Filozófia
  - kódolás közben a fájlok közti navigálás, inspekció és szerkesztés ne vigyen el sok időt
  - előnyös tanulási görbe
  - különböző módok a különböző fázisokhoz
    - normal mód - navigálás és inspekció
    - insert és replace módok - szerkesztés
    - visual és select módok - kijelölés
    - command-line mód - belső és külső parancsok futtatása

Vim - tanulás

- `vimtutor` (25-30 perc)
- Rendszeres használat minden feladatra





Shell

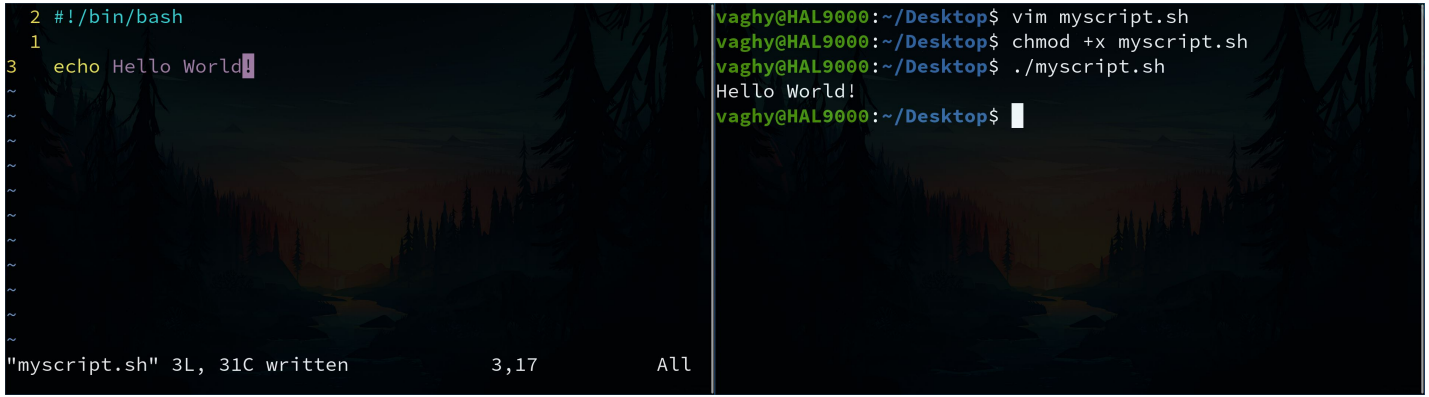
- `sh` - Bourne Shell
  - 1979-ben írta Stephen Bourne Unix-hoz
- `bash` - Bourne Again Shell
  - Bourne Shell továbbfejlesztett verziója, 1989-ből
  - alapértelmezett a legtöbb Linux disztribúcióban
- `csh` - C Shell
  - C-szerű nyelvi elemeket tartalmaz
- `zsh` - Z shell
  - alapértelmezett macOS Catalina-n
- `fish` - Friendly Interactive Shell
  - autocomplete
  - könnyű konfigurálni és script-elni

Az első script

- Nyissunk meg egy szövegszerkesztőt és írjuk bele a kedvenc parancsunkat
- `bash myscript.sh`

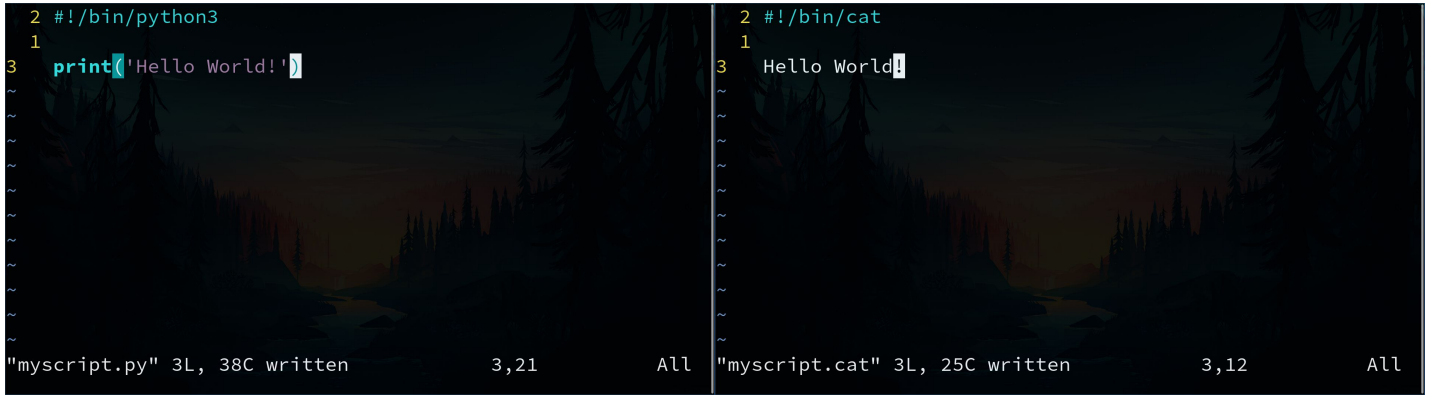
Script futtatása

- Az imént közvetlenül meghívtuk a shellt a futtatáshoz
- Remek lenne, ha az operációs rendszer tudná, hogy milyen programmal futtasson



Shebang (#!)

- Az első sor tartalmát a shell interpreter direktívaként parse-olja
- Szükséges a használatához, hogy futtatható legyen a fájl



Környezeti változók

- Gyakorlatilag globális változók, melyet a különböző programok felhasználnak és akár módosíthatnak
- `echo $SHELL`
- `echo $HOME`
- `echo $PWD`
- `echo $PATH`
  - a shell itt keresi a futtatni kívánt programot (futtatható vagy bináris)
  - `which -a command` parancs megmutatja az összes helyet a PATH-ban, ahol megtalálható

Fájlkiterjesztések

- Mi legyen a script-jeim kiterjesztése?
- A fájlkiterjesztés egy Windows-os dolog, a valóságban kétféle fájl létezik: szöveges és bináris
  - ami szöveges, azt lehet értelmesen szövegszerkesztőben szerkeszteni
  - ami bináris, ahhoz speciális szerkesztő kell (például .docx és Microsoft Word)
  - ami futtatható (x kapcsoló), az futtatható, ha benne van a PATH-ban, vagy az aktuális mappában
- Tehát értelmes keretek között mindegy, hogy mi a script kiterjesztése
  - természetesen célszerű felismerhető kiterjesztést használni

Változók

- Untyped változók, dinamikusan változhat a típus (string vagy egész szám)
- Létrehozás
  - `bar=5`
  - `let foo="Hello World!"`
  - `declare -r var1` - írásvédett változó
  - `declare -i var2` - egész változó
- Elérés
  - `let bar=$bar+1`
  - `echo $foo` - kitörli a tabulátorokat és az új sorokat
  - `echo "$foo"` - megőrzi a formázást
  - `echo '$foo'` - string literál
- Törlés
  - `unset var`

Command substitution

- `command1 $(command2)`
  - `command2` lefuttatása egy subshell-ben
  - eredmény behelyettesítése
  - `command1` lefuttatása az új argumentummal
- `foo=$(command)`
  - `command` lefuttatása
  - eredmény értékadása

Egész aritmetika

- `let expression`
  - `+, -, /, *, %, ++, --, **, <<, >>, &, ^, |, &&, ||`
  - `=, +=, -=, /=, *=, <<=, >>=, &=, ^=, |=`
- `echo $(expr operand1 operator operand2)`
  - `+, -, /, %, \*`
- `echo $((expression))` - arithmetic expansion
  - `+, -, /, *, %, ++, --, **, <<, >>, &, ^, |, &&, ||`
  - `=, +=, -=, /=, *=, <<=, >>=, &=, ^=, |=`
  - `$` nélkül visszatérési érték nélkül hajtja végre a műveletet
    - például változó inkrementálása `((var++))`

String aritmetika I.

- összefűzés
  - `echo $string1$string2`
- értékadás
  - `string2=$string1`
  - `string2+=$string1`
- string hossza
  - `echo ${#string}`
  - `echo $(expr length $string)`

String aritmetika II.

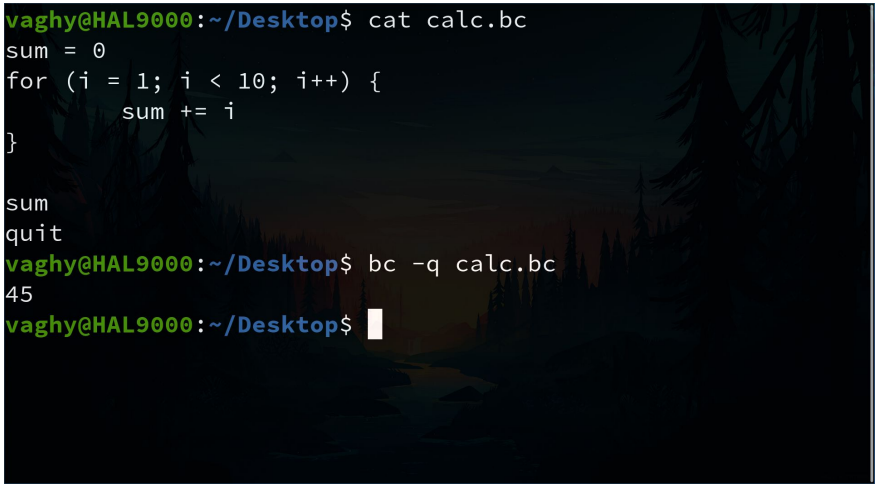
- substring
  - `echo ${string:pos:len}`
  - `echo $(expr substr $string pos len)`
- regex
  - `echo $(expr match $string $regex)` - visszaadja a találat utolsó indexét
  - `echo $(expr match $string \($regex\))` - visszaadja a találatot
- replace
  - `echo ${string/old/new}` - első találat helyettesítése
  - `echo ${string//old/new}` - összes találat helyettesítése

String aritmetika III.

- removal
  - `echo ${string#regex}` - kitörli a legrövidebb találatot
  - `echo ${string##regex}` - kitörli a leghosszabb találatot
  - `echo ${string%regex}` - kitörli a legrövidebb találatot hátulról
  - `echo ${string%%regex}` - kitörli a leghosszabb találatot hátulról

Float aritmetika

- `bc` - basic (best) calculator
  - `echo expression | bc`
  - lehet fájlban is számolni vele
    - változók
    - vezérlési szerkezetek
    - saját és beépített függvények
  - `man bc`



G02F01

- Egészítsük ki az előző `bc` script-t interpreter direktívával!
- Írjuk át a script-t úgy, hogy argumentumban fogadja a szummázás felső határát!
  - `bar=read()`
- Figyeljük meg, hogy az argumentumot lehet pipe-olni is!
- Próbáljuk ki elnevezett pipe-al is!

Adatszerkezetek

- Tömb
  - `bar=(1 2 3 4)`
  - `declare -a bar=(1 2 3 4)`
  - `bar+=(5)`
  - `echo ${bar[*]}`
  - `echo ${#bar[*]}`
- Asszociatív tömb
  - `foo=([key1]=val1 [key2]=val2)`
  - `declare -A foo=([key1]=val1 [key2]=val2)`
  - `foo[key3]=val3`
  - `echo ${foo[*]}`
  - `echo ${!foo[*]}`
  - `echo ${#foo[*]}`

Következő gyakorlat

- Archívumok, tömörítések
- Fájlkezelés
- Vezérlési szerkezetek
- Argumentumok kezelése
- Függvények

Érdekességek, olvasnivalók

- 7 Habits For Effective Text Editing 2.0
- How do I exit the Vim editor?
- Luke Smith: vimtutor Let's Play
- MIT Missing Semester: vim