Hálózatok építése és üzemeltetése

Linux alapok gyakorlat

Gyakorlatok

Katalógus + Opcionális házi feladatok

Gyakorlatok menetrendje

- Cél: otthoni gyakorlás!
- BME Cloud vagy saját gépen munkakörnyezet (HaEpUz VM)
- Előre kiadott anyagok (slide-ok, mintafeladatok, video)
- Jelenléti gyakorlat (QBF12, QBF11)
  - tananyag áttekintése, példa megoldások bemutatása, konzultáció
- Otthoni gyakorlás
  - kiadott gyakorlófeladatok
  - kérdések Teams chatben, esetleg online konzultáció?
- Gyakorlat zárása: (Google Forms) kvíz beküldése
  - e-katalógus
  - 1. rész: “elmélet” (mint a ZH-n)
    - egy-két egyszerű kérdés alap részből
  - 2. rész: személyre szabott feladatok (mint a vizsgán)
    - feladatok megoldása a saját VM-ben, válaszok a kvízben
    - plusz pontok gyűjthetők a vizsgára
    - +2% gyakorlatonként, de (7 gyakorlaton) max 10%

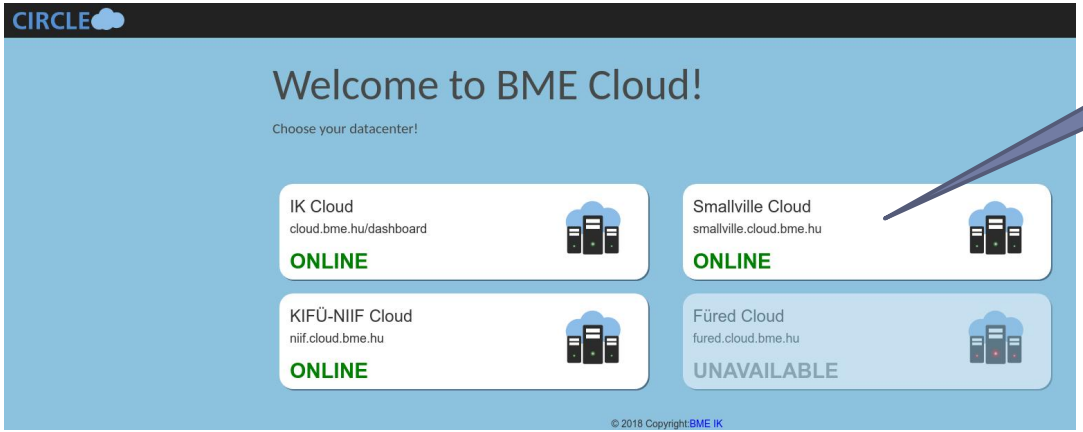
Jupyter Notebook

- by Pelle István
- VM felépítése (idén nem kell!)
  - Vagrant program segítségével
  - vagrantfile: https://goo.gl/33GBUQ
- Segédlet a telepítésről és a használatról
  - https://goo.gl/Yf5XRY

Munkakörnyezet: HaEpUz VM

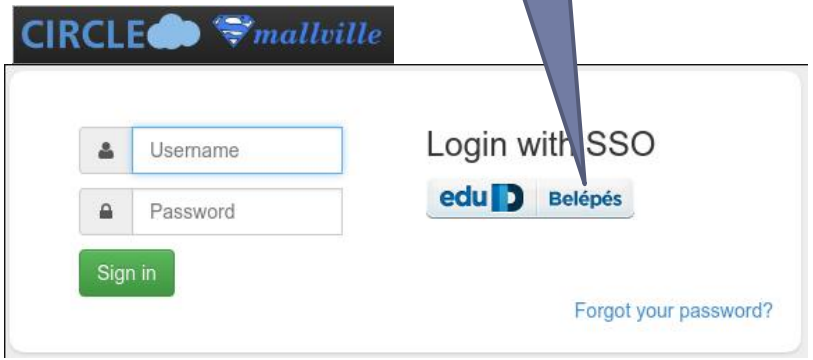
BME VIK Cloud - Smallville

https://cloud.bme.hu



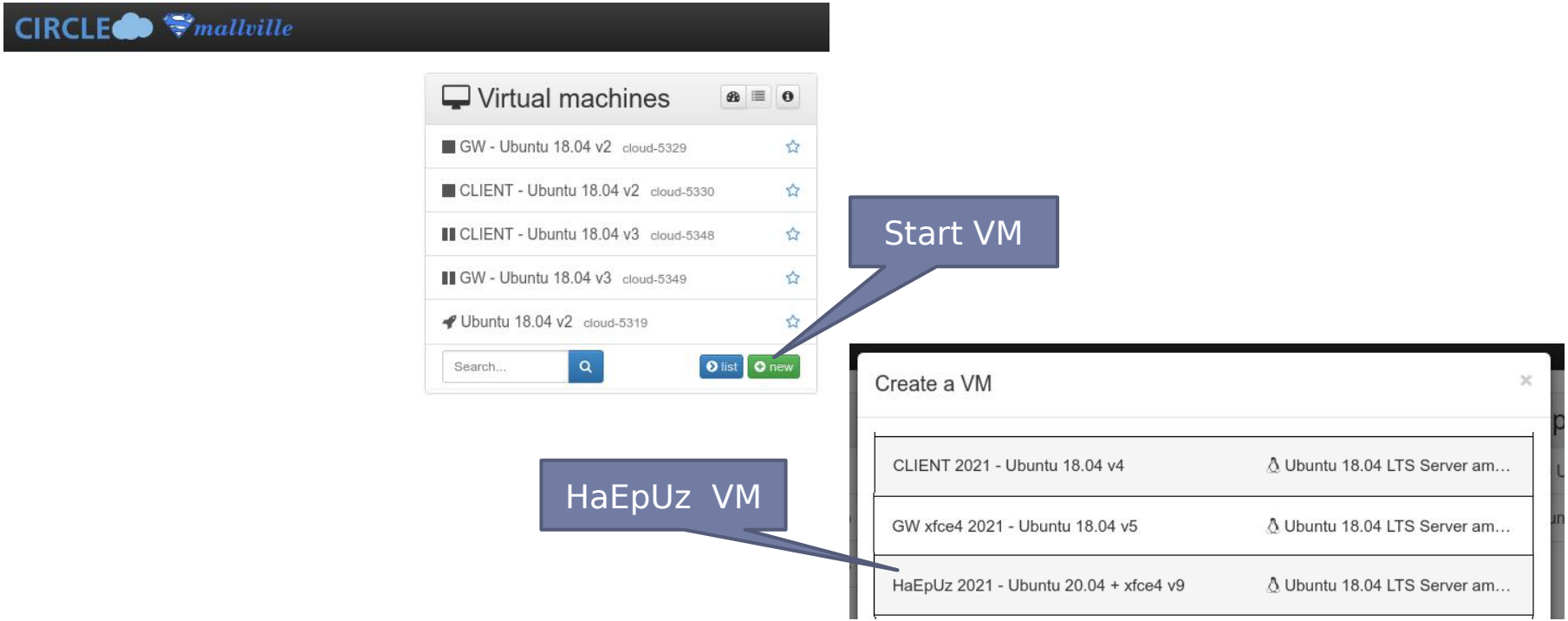
Smallville
BME VIK

Címtáras
belépés

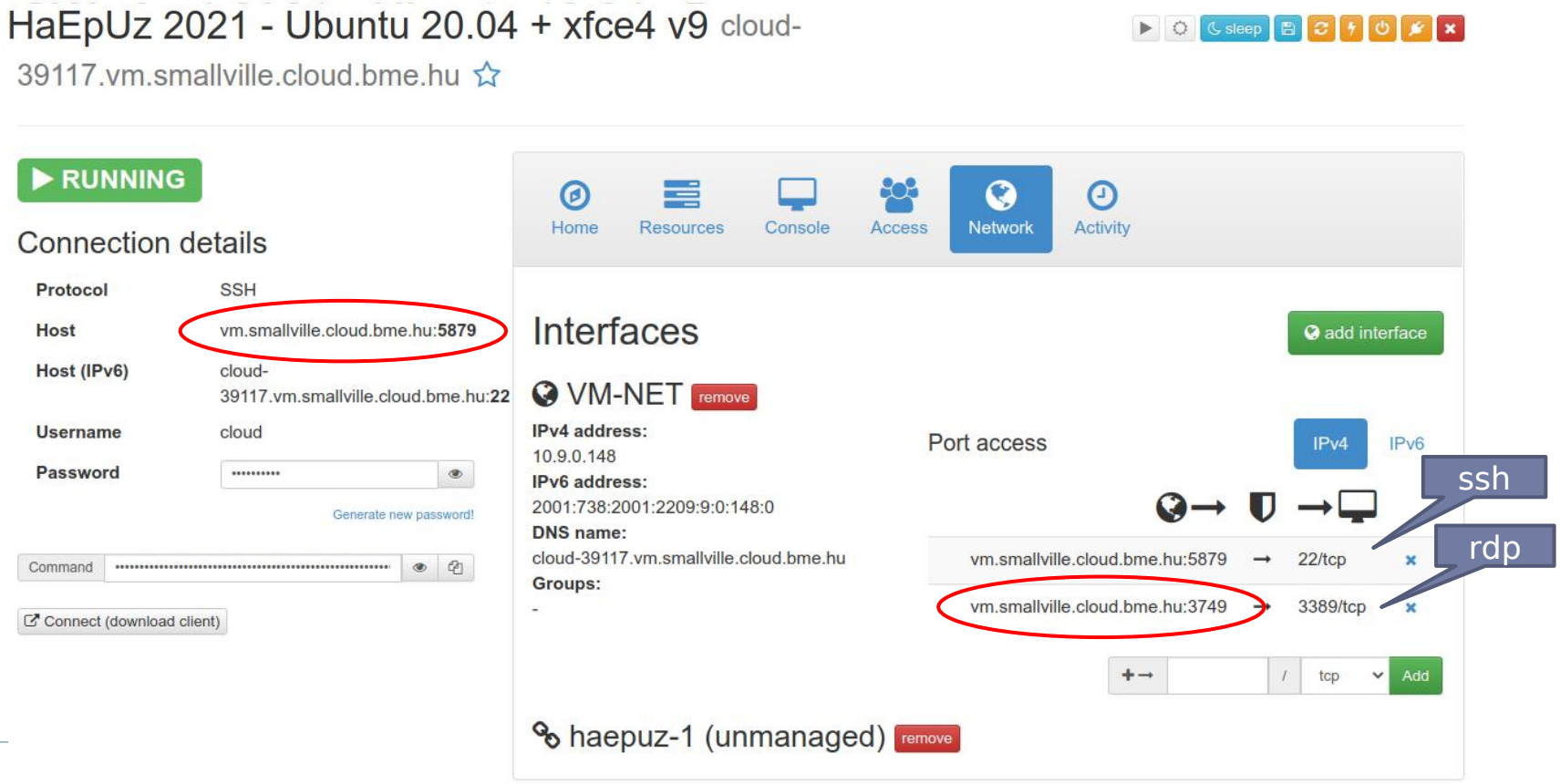


Special thanks to:
Szeberényi Imre (IIT) & CIRCLE Cloud team

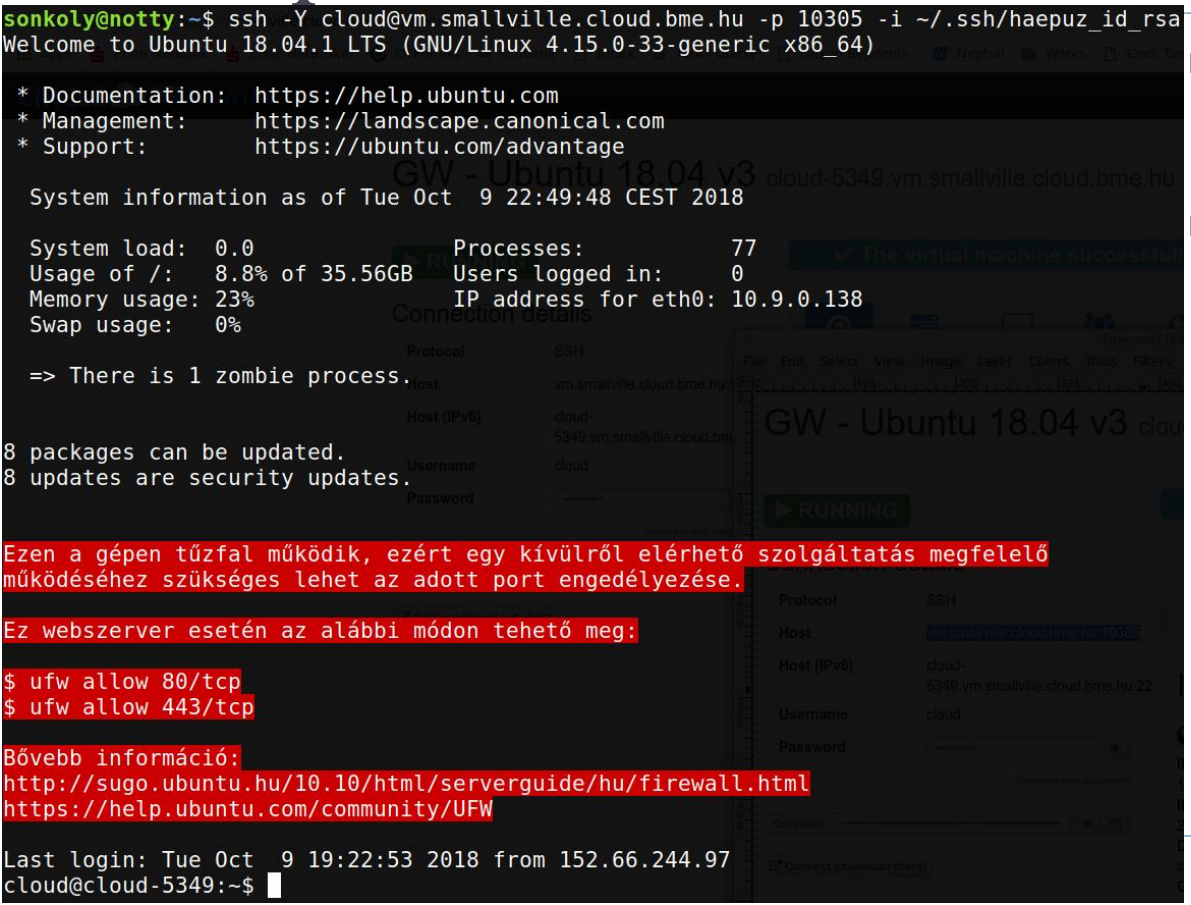
VM indítása


CIRCLE mallville
Virtual machines
GW - Ubuntu 18.04 v2 cloud-5329
CLIENT - Ubuntu 18.04 v2 cloud-5330
CLIENT - Ubuntu 18.04 v3 cloud-5348
GW - Ubuntu 18.04 v3 cloud-5349
Ubuntu 18.04 v2 cloud-5319
Search...
list
new
Start VM
HaEpUz VM
Create a VM
CLIENT 2021 - Ubuntu 18.04 v4
Ubuntu 18.04 LTS Server am...
GW xfce4 2021 - Ubuntu 18.04 v5
Ubuntu 18.04 LTS Server am...
HaEpUz 2021 - Ubuntu 20.04 + xfce4 v9
Ubuntu 18.04 LTS Server am...

Belépés: ssh vagy rdp



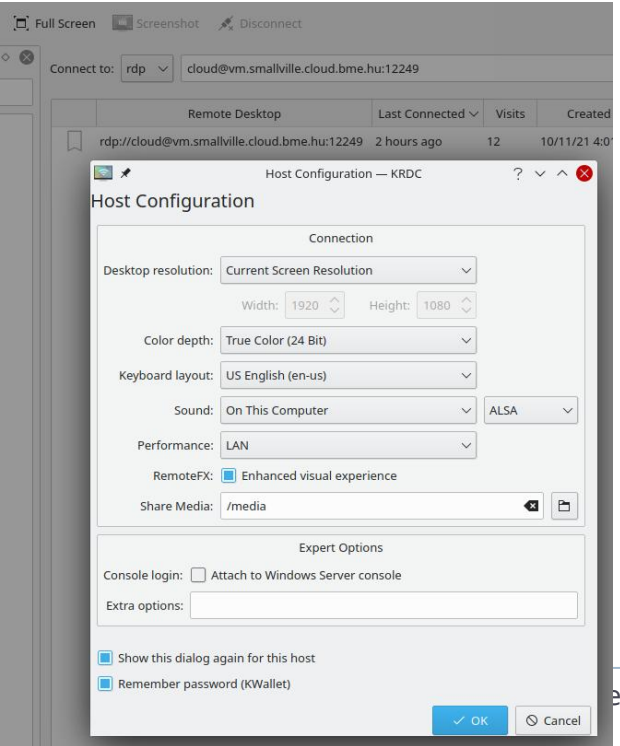
Belépés: ssh (GUI nélkül)

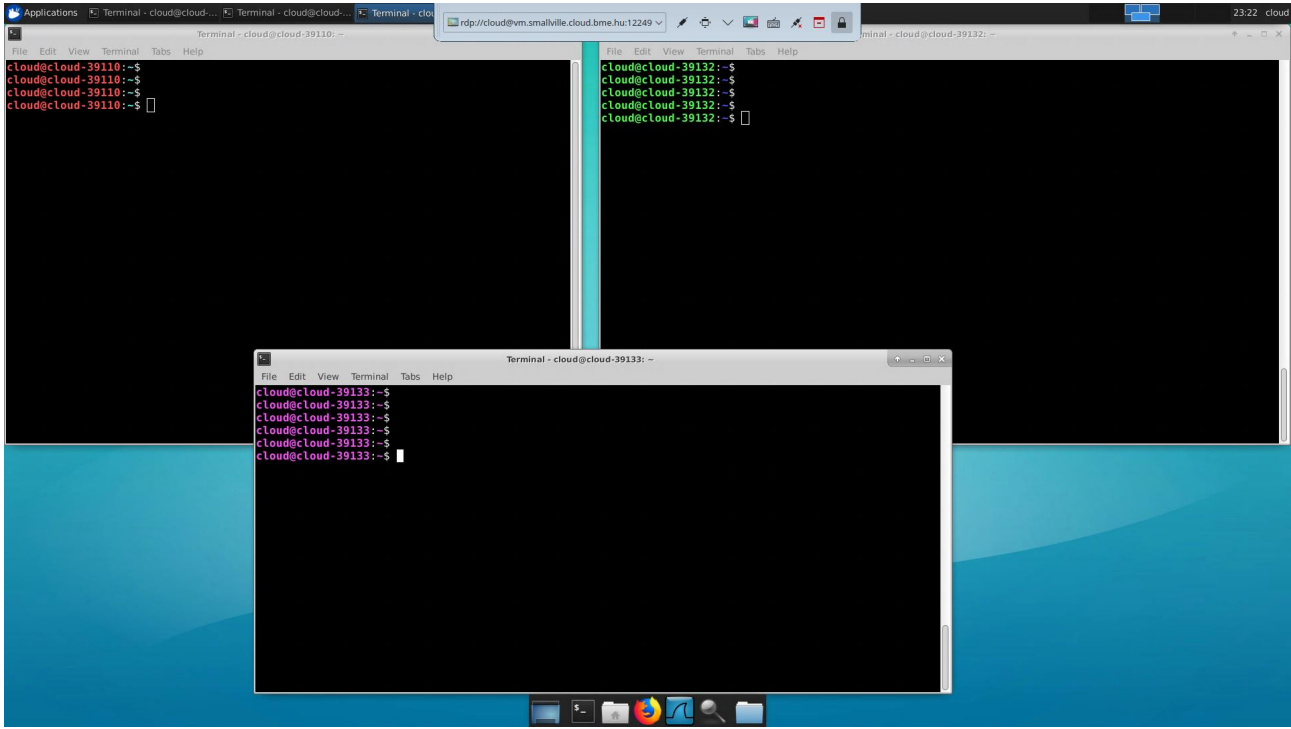


- célszerű Linux kliensről
  - pl.: HaEpUz VM
- célszerű ssh kulcs feltöltése
  - My profile
    - SSH public keys
    - kulcs pár (priv, pub)
      - generálása (pl.: ssh-keygen)
      - vagy meglévő feltöltése
  - VM/Home
    - Install SSH keys

Belépés: rdp (xfce4 desktop)

- rdp kliens
  - pl.: krdc, remmina, MS...





Alapvető parancsok

Parancsok

- Parancsok a shell-ben adhatók ki
- általános felépítésük:
  - `parancs kapcsolók argumentumok ...`
  - kapcsolók legtöbbször “-” jellel kezdődnek
  - ha fájlnév helyén áll “-”, akkor a standard inputot vagy standard outputot jelenti
  - pl: `ls -l *.txt` (részletes lista a .txt végződésű fájlokról)

Hasznos parancsok

- **man**
  - leghasznosabb parancs, minden UNIX-alapú rendszer részét képezi
  - felhasználói kézikönyv, mely az összes parancs, függvény, API hívás leírását tartalmazza, valamint a főbb konfigurációs fájlokét
  - a man oldalakról a q billentyű lenyomásával lehet kilépni
  - pl: `man ls`
- **mc**
  - Midnight Commander: könnyen használható fájlkezelő program sok hasznos segédfunkcióval
- **shutdown, halt, reboot, poweroff**
  - rendszer leállítása vagy újraindítása
- **su, su username; sudo, sudo -u username**
  - superuser jogosultság megszerzése (su), illetve adott felhasználói jogosultság megszerzése
  - sudo használatával egy parancs hajtható végre az adott jogosultsággal

Fájlrendszerrel kapcsolatos műveletek

- **cd, pwd, mkdir, rmdir, ls, find, tar**
  - Könyvtárműveletek: aktuális könyvtár megváltoztatása, aktuális könyvtár kiíratása, könyvtár létrehozása, üres könyvtár törlése, könyvtár tartalmának kiíratása, fájlok keresése a könyvtár hierarchiában nevük vagy tulajdonságaik alapján, könyvtárak archiválása vagy visszaállítása
  - nem üres könyvtárak törlésére a fájltörlési parancs rekurzív változatát kell használni (rm -r)

- **touch, rm, cp, mv, ln**
  - Fájlműveletek: fájl létrehozása vagy "megérintése" (dátumok aktuálisra állítása rajta), törlése, másolása, mozgatása, linkelése (szimbolikus link létrehozásához használjuk a -s opciót)
  - cp, mv, ln paraméterei mindig forrás – cél sorrendben követik egymást
  - -r (vagy -R) kapcsolóval lehet rekurzívan (alkönyvtárakkal együtt) végeztetni fájlműveleteket

Fájlrendszerrel kapcsolatos műveletek

- Példák
  - könyvtárak, fájlok kezelése
    - `cd`
    - `pwd`
    - `ls`
    - `ls -l`
    - `ls -lah`
    - `mkdir linux`
    - `cd linux`
    - `touch test test2`
    - `rm test`
    - `cd ..`
    - `cd..` (???)
    - `cd .`
    - `rmdir linux`
    - `rm -fR linux`

- Példák
  - szimbolikus link létrehozása a szülő könyvtárban elhelyezkedő prog fájlra proglink néven:
    - `touch prog`
    - `mkdir valami`
    - `cd valami`
    - `ln -s ../prog proglink`
    - `ls -l`
  - az aktuális könyvtártól (“.”) rekurzívan keresi a .html fájlokat és a talált fájlokról részletes információt ad (mindegyik találatra végrehajtja az ls -l parancsot):
    - `mkdir a; cd a`
    - `touch a.html b.html c.html`
    - `cd ..; touch x.html`
    - `find . -name '*.html' -exec ls -l '{}' \;`

Fájlrendszerrel kapcsolatos műveletek

- 1. Feladat
  - A rendszerbe való belépés után indíts el egy terminált. A saját home könyvtáradban hozz létre egy 'linux-alapozo' alkönyvtárat
    - Milyen jogosultságokkal rendelkeznek az egyes felhasználók a létrehozott könyvtárhoz?
  - Lépj be a 'linux-alapozo' könyvtárba és hozz létre három tetszőleges tartalmú szöveges fájlt f1, f2 és f3 néven.
    - Milyen védelmi kóddal jöttek létre a fájlok?
    - Állítsd be úgy a jogosultságokat, hogy az f2 fájlra a csoportod többi tagja is rendelkezzen írási joggal.
  - Hozz létre egy 'linkek' alkönyvtárat, majd ezen belül hozzál létre szimbolikus linkeket az előbb létrehozott három fájlra f1link, f2link, f3link névvel.

Szűrők

Szűrők

- Nagyon hasznos Unix eszközök
- egyszerű programok
  - standard bemenetüket a megfelelő művelet elvégzése után a standard kimenetükre másolják
  - sok önmagában nagyon egyszerű műveletet megvalósító szűrő van
  - általában a segédprogramok képesek szűrőként is működni
  - szűrők egymás után kapcsolhatók a pipe (csővezeték) segítségével
- shell által végrehatjott programok alapból 3 megnyitott állománnyal indulnak
  - standard input (0)
  - standard output (1)
  - standard error (2)
  - ezek átirányíthatók

Szűrők

- Bemenet/kimenet átirányítás
  - prog < file: standard input átirányítása (vagy hosszabban: 0<)
  - prog > file: standard output átirányítása (vagy hosszabban: 1>)
  - prog 2> file: standard error átirányítása
  - prog 2>&1: standard error átirányítása standard outputba
  - prog 1>&2: standard output átirányítása standard errorba
  - prog1 | prog2: pipe, prog1 kimenetének prog2 bemenetére irányítása
  - prog >> file: standard output hozzáírása (append) a megadott fájlhoz

Egyszerű példák

- **echo, cat, tee**
  - Paraméterként átadott szöveg kiíratása (echo), illetve fájlok kiíratása és összefűzése (cat). Gyakran használjuk a standard output átirányításával, vagy pipe-okkal együtt. A tee parancs a standard inputról másol a standard outputra, valamint a paraméterként megadott fájlokba is (adatfolyam elágaztatása).
  - pl:
    - `cat /etc/passwd`
    - `echo 'asdf' > f1`
  - pl: cat program kimenetét átirányítjuk az f1 fájlba, így a standard bemeneten bevitt sorok az adott fájlba íródnak egészen a fájlvége jel (ctrl-d) beviteléig:
    - `cat >f1`
- **more, less**
  - Fájlok kiíratása úgy, hogy egyszerre egy képernyőnyi tartalom jelenik meg, illetve navigálási lehetőség biztosítása. A less a kifinomultabb változat.
    - `less /etc/apache2.conf`
    - `cat /etc/apache2.conf | less`

Egyszerű példák

- **head, tail**
  - Fájlok első (head), illetve utolsó (tail) n sorának kiíratása.
    - `head -2 /etc/group`
    - `sudo tail -n 20 /var/log/syslog`
    - `sudo tail -f /var/log/syslog`
- **tr**
  - Alapértelmezésben karakterfordítást végez (translate): az első paraméterként megadott karaktereket cseréli a második paraméterben megadottakra. Tartomány is megadható, pl. [0-9] a számokat jelenti, [a-z] a kisbetűket. Ha az első paramétere -d, akkor törli a második paraméterben megadott karaktereket. Pipe részeként vagy átirányítással használjuk.
    - `echo 'abcd' | tr bc xy`
- **wc**
  - Kiírja a sorok, szavak és karakterek számát ("word count").
    - `wc /etc/passwd`
    - `wc -l /etc/passwd`
    - `wc -w /etc/passwd`
    - `wc -c /etc/passwd`

Egyszerű példák

- **cmp, diff, comm**
  - Fájlok összehasonlítása: bájtról bájtra (cmp) vagy szöveges fájlokat sorról sorra (diff). A comm parancs két fájl közös sorainak kiíratására használható.
    - `diff .bashrc .bashrc~`
- **sort, uniq**
  - A két parancsot általában együtt (egymásba pipe-olva) használjuk és ilyen sorrendben: a sort rendezi a bemenetet, míg a uniq a rendezett bemenet ismétlődő soraiból csak egyet-egyet hagy meg.
  - A uniq paraméterezésével többféle működés is elérhető, pl. a sorok különféle számolása (pl. uniq -c), csak a többször szereplő (uniq -d), vagy az egyedi sorok kiíratása (uniq -u). A sort paraméterezésével számok és stringek rendezése is megoldható.
  - Például az alábbi egymás után kapcsolt szűrők a jelszófájlt rendezik a 3. oszlop szerint (-k3) numerikusan (-n) csökkenő sorrendben (-r) és az utolsó két sor lesz az eredmény (tail -n 2). A jelszófájlban a mezők közti szeparátor a :, ami a rendezésnél a -t kapcsolóval adható meg:
    - `cat /etc/passwd | sort -t: -n -k3 -r | tail -n 2`

Egyszerű példák

- 2. Feladat
  - Készíts egy szűrőt, mely a standard bemenetét a kimenetre másolja úgy, hogy közben azt nagybetűssé konvertálja (a kisbetűket nagybetűkké alakítja, a többi karaktert változatlanul hagyja).
  - Készítsd el az előző szűrő módosított változatát, amely invertálást végez, vagyis a nagybetűket kisbetűre, míg a kisbetűket nagybetűre cseréli.

Egyszerű példák

- 2. Feladat
  - Készíts egy szűrőt, mely a standard bemenetét a kimenetre másolja úgy, hogy közben azt nagybetűssé konvertálja (a kisbetűket nagybetűkké alakítja, a többi karaktert változatlanul hagyja).
    - **`echo 'asdfASDF' | tr a-z A-Z`**
  - Készítsd el az előző szűrő módosított változatát, amely invertálást végez, vagyis a nagybetűket kisbetűre, míg a kisbetűket nagybetűre cseréli.
    - **`echo 'asdfASDF' | tr a-zA-Z A-Za-z`**

grep

- **grep, egrep, fgrep**
  - Reguláris kifejezés-illesztő. A paraméterben (idézőjelben!) megadott reguláris kifejezésre (regexp) illeszti a bemenetet.
  - Fontosabb paraméterei:
    - -A, -B: az illesztett sor környezetét (előző / következő, adott számú sorokat) is megmutatja
    - -v: fordított működést eredményez (nem illesztett sorokat mutatja)
    - -q: nincs output, csak a visszatérési értéket állítja be (if feltételeként szoktuk használni)
  - grep és sed parancsok reguláris kifejezéseiben az operátorokat "escape-elni" kell
    - (különben karakternek tekinti őket a program)
    - pl: “\|” a vagy operátor, míg “|” a pipe karakter
  - példák:
    - `$ echo "bcd" | grep "a.*"`
    - `$ echo "bcacb" | grep "a.*b"`
    - `bcacb`
    - `$ echo "baaa" | grep "a*"`
    - `baaa`
    - `$ echo "baaa" | grep "^a*$"`

sed

- **sed**
  - Teljes funkcionalitását tekintve sorszerkesztő, mi reguláris fordítóként fogjuk használni.
  - Erre az s parancsa szolgál:
    - s/kif1/kif2/
    - kif1 reguláris kifejezést fordítja kif2 kifejezésre mindazon sorokon, amelyekre kif1 illeszkedik.
    - a(z escape-elt) zárójelbe tett kifejezésrészletekre vissza lehet hivatkozni kif2 -ben a \1, \2, ... referenciákkal.
    - ha az s parancs záró / -je után még egy g paramétert írunk, akkor soronként többször is végez illesztést
  - Példák:
    - **`$ echo 'xxxaaaxxx' | sed 's/aaa/bbb/'`**
    - **`xxxbbbxxx`**
    - **`$ echo "a0001b" | sed 's/a\([0-9]*\)b/x\1y/'`**
    - **`x0001y`**

Reguláris kifejezések

c	Maga a c karakter, ha az nem speciális karakter.
\c	Kikapcsolja a c karakter speciális jelentését. Pl: \[: zárójel kezdődik
^	Sor eleje.
$	Sor vége.
.	Egy darab bármiyen karakter. (Az újsor kivételével minden karakter illeszkedik rá.)
[abc]	Bármelyik karakter a halmazból.
[^abc]	Bármelyik karakter, amelyik nincs a halmazban.
[a-z]	Bármelyik karakter a megadott tartományból.
r*	r reguláris kifejezés tetszőlegesen sokszor (akár 0-szor).
r+	r reguláris kifejezés 1-szer vagy sokszor. (extended regexp)
r?	r reguláris kifejezés 0-szor vagy 1-szer. (extended regexp)
r1r2	r1 és r2 egymás után úgy, hogy r1 a lehető leghosszabban illeszkedjen.
r1¦r2	r1 vagy r2. (extended regexp)
(...)	egymásba ágyazott kifejezések. (extended regexp)
r{n}	r reguláris kifejezés n-szer megismétlődik. (extended regexp)
r{n,}	r legalább n-szer megismétlődik. (extended regexp)
r{n,m}	r legalább n-szer, legfeljebb m-szer megismétlődik. (extended regexp)
\(r\)	r reguláris kifejezés önmaga, amire később hivatkozni lehet \n alakban.
\n	hivatkozás az n-edik \(r\) reguláris kifejezésre.

Reguláris kifejezések: példák

- Az /etc/passwd fájlból írassuk ki az összes olyan sort, amelyben az 'r' és a 't' karakterek között tetszőleges számú 'o' szerepel:
  - `$ cat /etc/passwd | grep 'ro*t'`
- Írassuk ki az aktuális könyvtár összes olyan könyvtárát, amihez mindenkinek írási joga van:
  - `$ ls -l | grep '^d.......w.'`
- Írassunk ki minden olyan sort, amiben egymás után szerepel ugyanaz a betű:
  - `$ cat /etc/passwd | egrep '(.)\1'`
- Cseréljünk le minden .conf fájlnév részletet .CONFIG-ra a kimeneten:
  - `$ ls | sed s/\.conf/.CONFIG/`
  - miért rossz? `$ ls | sed s/.conf/.CONFIG/`

Reguláris kifejezések: példák

- Első és második karakter felcserélése egy fájlban:
  - `$ cat file1 | sed 's/\(.\)\(.\)/\2\1/'`
- Jelszófájl első két mezőjének felcserélése (mező szeparátor a kettőspont):
  - `$ cat /etc/passwd | sed 's/^\([^:]*\):\([^:]*\):/\2:\1:/'`
- Számoljuk meg melyik login shell hányszor szerepel az /etc/passwd fájlban (utolsó oszlop), majd ezt rendezzük csökkenő sorrendbe és írjuk ki a két legelsőt:
  - `$ cat /etc/passwd | sed 's/.*:\([^:]*\)/\1/' | sort | uniq -c | sort -n -r | head -n2`

Bash alap(f)ok

Bourne Again Shell

- Parancsértelmező és egy programozási nyelv is egyben
- segítségével egyszerűen kialakíthatók ún. shell scriptek, melyekkel a parancskészlet tetszőlegesen bővíthető
- paraméterezhetősége hasonló a normál programokéhoz
- Linux rendszerek egyik alapértelmezett shellje
- egyszerű adminisztrációs programok készíthetők
- bemenet/kimenet átirányítás és a pipe alkalmazásával nagyon hatékony eszköz
- Bash-ben a sorok lezárhatók enterrel vagy pontosvesszővel
- hosszabb kódrészlet: do ... done blokkba (hasonlóan, mint a C programok { ... } blokkja)
- Futtatható script:
  - első sora: hashbang + a bash elérési útja (#!/bin/bash)
  - **`chmod a+x script`** paranccsal futtathatóvá kell tenni (enélkül **`bash script`** paranccsal történik)
- aktuális könyvtár alapértelmezésben nincs benne a PATH környezeti változóban
  - így ./script paranccsal indíthatjuk
  - (vagy hozzáadjuk a PATH -hoz az aktuális könyvtárat a PATH=$PATH:. paranccsal)

Hasznos shell funkciók

- alt-F1,F2,... szöveges terminálok közti váltás
- ctrl-alt-F1,... másik terminálra váltás grafikus terminálról
- ↑ és ↓ history, korábbi parancsok behívása
- ctrl-r history, parancs illesztése az első megfelelőre (reverse search)
- TAB állománynév kiegészítés
- shift-PgUP képernyő tartalmának léptetése
- shift-PgDown
- ctrl-a sor elejére ugrás
- ctrl-e sor végére ugrás
- ctrl-l képernyő újrarajzolása

Változók, idézőjelek

- Változókat nem kell deklarálni, név szerint lehet rájuk hivatkozni
  - értékadás: VAR=valami (egyenlőségjel baloldalán nincs szóköz!)
  - érték lekérése: $VAR vagy ${VAR}
  - standard inputról változóba: read parancs
- Idézőjelek:
  - szimpla idézőjel: ‘echo $VAR’
    - az idézett szöveg változtatás nélkül kerül feldolgozásra
  - dupla idézőjel: “echo $VAR”
    - az idézett szövegen lefut a változóhelyettesítés, majd változtatás nélkül kerül feldolgozásra
  - vissza idézőjel: `echo $VAR`
    - az idézett szöveget parancsként értelmezi, és az eredményét adja át
- parancshelyettesítés másképp, pl: current_dir=$(pwd)
- aritmetikai helyettesítés, pl: c=$(($1+$2)) vagy $((c=$1+$2))

Wildcardok

- Állománynév-helyettesítés
  - * tetszőleges számú tetszőleges karakter
  - ? pontosan egy tetszőleges karakter
  - [abc] bármelyik karakter a halmazból
  - [a-z] bármelyik karakter az adott intervallumból
  - {a,b,c} “brace expansion”
- Példák:
  - **`$ ls -l image[1-3].jpg`**
  - **`$ ls -l image[1-3].{jpg,bmp}`**
    - shell erre fordítja (ha ezek a fájlok mind léteznek): **`$ ls -l image1.jpg image1.bmp image2.jpg image2.bmp image3.jpg image3.bmp`**

Speciális változók

- $0 scriptfájl neve
- $\# bemeneti paraméterek száma
- $i az i-edik bemeneti paraméter
- $? utolsó visszatérési érték
- $@ és $* az összes paraméter
- $\$ processz ID
- $HOME home könyvtár
- $HOSTNAME gép hosztneve
- $PATH elérési utak az állományokhoz
- $UID aktuális user ID
- $PS1 aktuális prompt
- $IFS input mező szeparáló karakter

Beépített parancsok

:	nem csinál semmit
. vagy source	más fájlok include-olása
alias/unalias	alias beállítása / eltávolítása
break/continue	ciklus elhagyása / következő iteráció
cd	aktuális könyvtár megváltoztatása
echo	argumentumok kiírása
eval	argumentum mint parancs végrehajtása
exec	argumentum végrehajtása, de nem indul új shell
exit	kilépés
export	shell változó exportálása
bg/fg	job háttérbe / előtérbe helyezése
let	aritmetikai kifejezés kiértékelése
pwd	aktuális könyvtár lekérdezése
read/readonly	változóba olvasás standard inputról
return	visszatérés függvényből
set/unset	változók lekérdezése / beállítása
shift	pozicionális paraméterek léptetése
test vagy [ ]	feltétel kiértékelése
times	futási idők

Vezérlési szerkezetek

- if
  - `if [ "$VAR" == "valami" ]; then`
  - `  echo "be van állítva"`
  - `else`
  - `  echo "nincs"`
  - `fi`

  - `if grep -q "hello"; then`
  - `  echo "szia"`
  - `fi`

- for
  - `for i in a b c; do`
  - `  echo $i`
  - `done`

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

Vezérlési szerkezetek

- while
  - `while read v; do`
  - `  echo "Új sor jött: $v"`
  - `done`

- case
  - `case "$VAR" in`
  - `hello)`
  - `  echo "szia"`
  - `;;`
  - `bye)`
  - `  echo "bye-bye"`
  - `;;`
  - `*)`
  - `  echo "nem értem"`
  - `;;`
  - `esac`

Feltételek kiértékelése (test vagy [ ])

- test s — igaz, ha s nem null string
- test -z s — igaz, ha s nulla hosszúságú string
- test -n s — igaz, ha s nem nulla hosszúságú string
- test s1==s2 — igaz, ha s1 string megegyezik s2-vel
- test s1!=s2 — igaz, ha s1 string nem egyezik meg s2-vel
- test n1 -eq n2 — igaz, ha n1 aritmetikailag egyenlő n2-vel
- -ne, -gt, -ge — nem egyenlő, nagyobb, nagyobb vagy egyenlő
- -lt, -le — kisebb, kisebb vagy egyenlő
- test -f — igaz, ha f file létezik és nem könyvtár
- test -r — igaz, ha f file létezik és olvasható
- test -w — igaz, ha f file létezik és írható
- kombinálhatók: ! (tagadás), -o (vagy), -a (és)

Példák

- Egész számok kiírása 1-től 100-ig:

```
#!/bin/bash
c=1
while [ $c -le 100 ]
do
    echo $c
    c=$(($c+1))
done
```

- hány fájl van az aktuális könyvtárban:

```
#!/bin/bash
n=0
for i in *; do
    if [ -f $i ]; then
        n=$(($n+1))
    fi
done
echo "$n fájl van a könyvtárban"
```

3. Feladat

- Írj egy bash scriptet, ami egy könyvtárban az összes .htm kiterjesztésű fájlt .html kiterjesztésűre cseréli.

- Készítsd el a fordított változatot is, ami .html-ről .htm-re cserél.

3. Feladat

- Írj egy bash scriptet, ami egy könyvtárban az összes .htm kiterjesztésű fájlt .html kiterjesztésűre cseréli.
  - `#!/bin/bash`
  - `for i in *.htm; do`
  - `    mv $i ${i}l`
  - `done`

- Készítsd el a fordított változatot is, ami .html-ről .htm-re cserél.
  - `#!/bin/bash`
  - `for i in *.html; do`
  - `    mv $i "$(echo $i | sed 's/.$//')"`
  - `    vagy:  mv $i "$(echo $i | sed 's/html$/htm/')"`
  - `    vagy: rename 's/\.html$/\.htm/' *.html`
  - `done`

4. Feladat

- Írj scriptet, mely az aktuális könyvtárban található összes .jpg, .bmp és .png képet 50%x50%-osan kicsinyíti. Ehhez használd a convert programot, mely az imagemagick csomag része. A convert program használatát, és az átméretezéshez szükséges opciót a man convert paranccsal nézheted meg.

- ha nincs imagemagick:
- **`sudo apt-get update`**
- **`sudo apt-get install imagemagick`**

4. Feladat

- Írj scriptet, mely az aktuális könyvtárban található összes .jpg, .bmp és .png képet 50%x50%-osan kicsinyíti. Ehhez használd a convert programot, mely az imagemagick csomag része. A convert program használatát, és az átméretezéshez szükséges opciót a man convert paranccsal nézheted meg.

- `#!/bin/bash`
- `for i in *.{jpg,png,bmp}; do`
- `    convert -resize 50% $i conv_$i`
- `done`

5. Feladat

- Írj scriptet, mely az aktuális könyvtár tartalmán rekurzívan végigmegy, és minden szimbolikus link helyére bemásolja azt a fájlt, amelyre az mutatott (ehhez közben a linket letörli).
  - Segítség: a find paranccsal nézd végig a fájlokat, melyekre ellenőrizd, hogy szimbolikus linkek-e. Ehhez if [ ] típusú feltételes elágazásra lesz szükséged, a megfelelő if -feltételt a man test oldalon találod meg. A linkek célpontját pl. az ls -l paranccsal is kinyerheted.

5. Feladat

```
#!/bin/bash

for i in `find`; do
    if [ -h $i ]; then
        target=`ls -l $i | sed 's/.*->.//'`
        vagy: target=`ls -l $i | sed 's/.*->\s\(.*\)/\1/'`
        vagy target=`ls -l $i | cut -d' ' -f 10`
        vagy target=$(file $i | awk '{print $5}')
        vagy target=$(readlink $i)
        rm $i
        cp $target $i
    fi
done
```

Kiugró

- https://sb.tmit.bme.hu/haepuz-gyak1