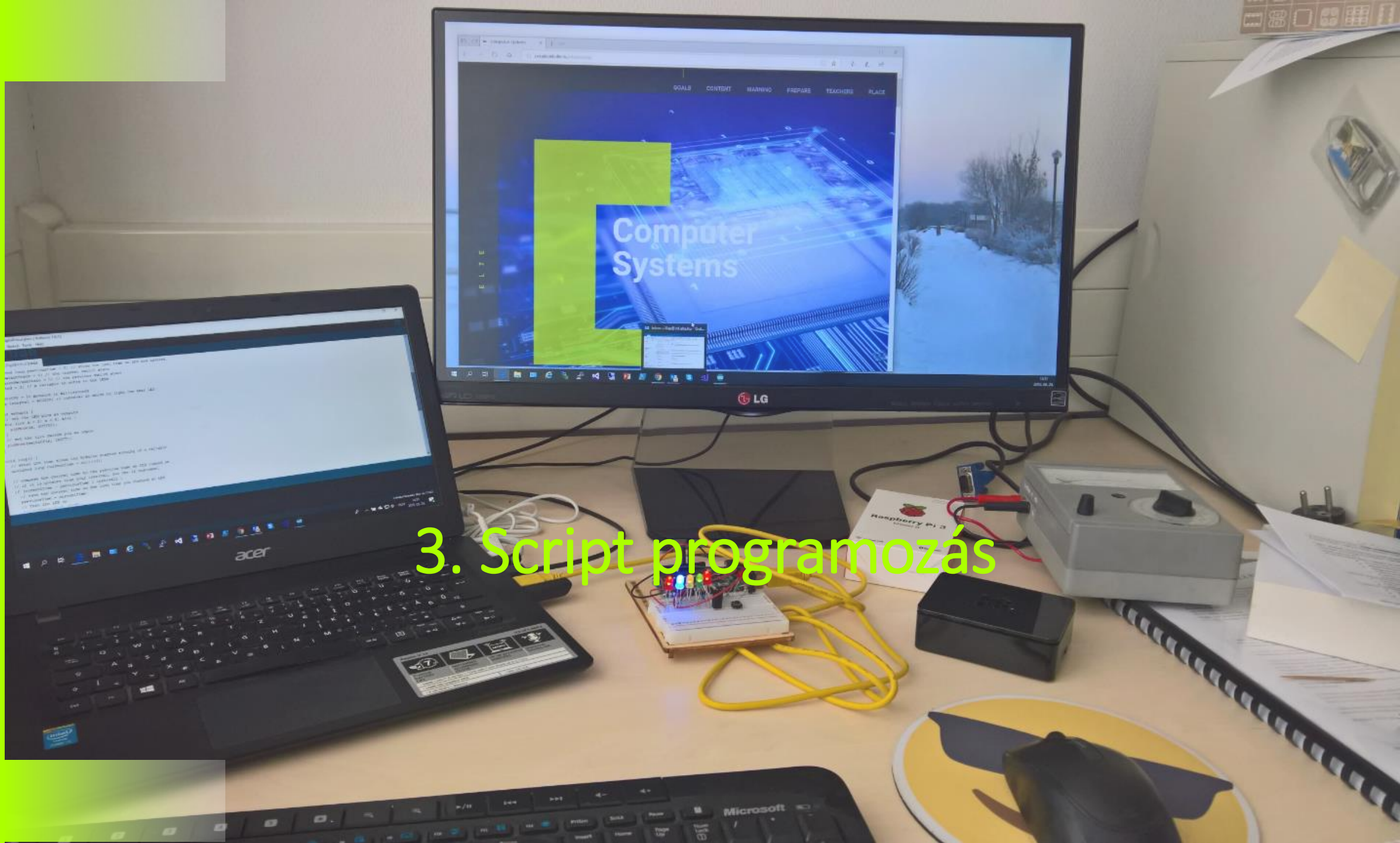


3. Script programozás



Visszatekintés

- Számítógépek, jelek, információk, tárolásuk
- Számábrázolás
 - Fixpontos, lebegőpontos számábrázolás
- Kódolás
 - ASCII, UTF-8 stb kódtáblák
- Felépítés, operációs rendszer szerep
- Kliens-Szerver különbségek
- Grafikus, karakteres kapcsolat
- Fájltrendszerek, alapvető parancsok

Mi jön ma?

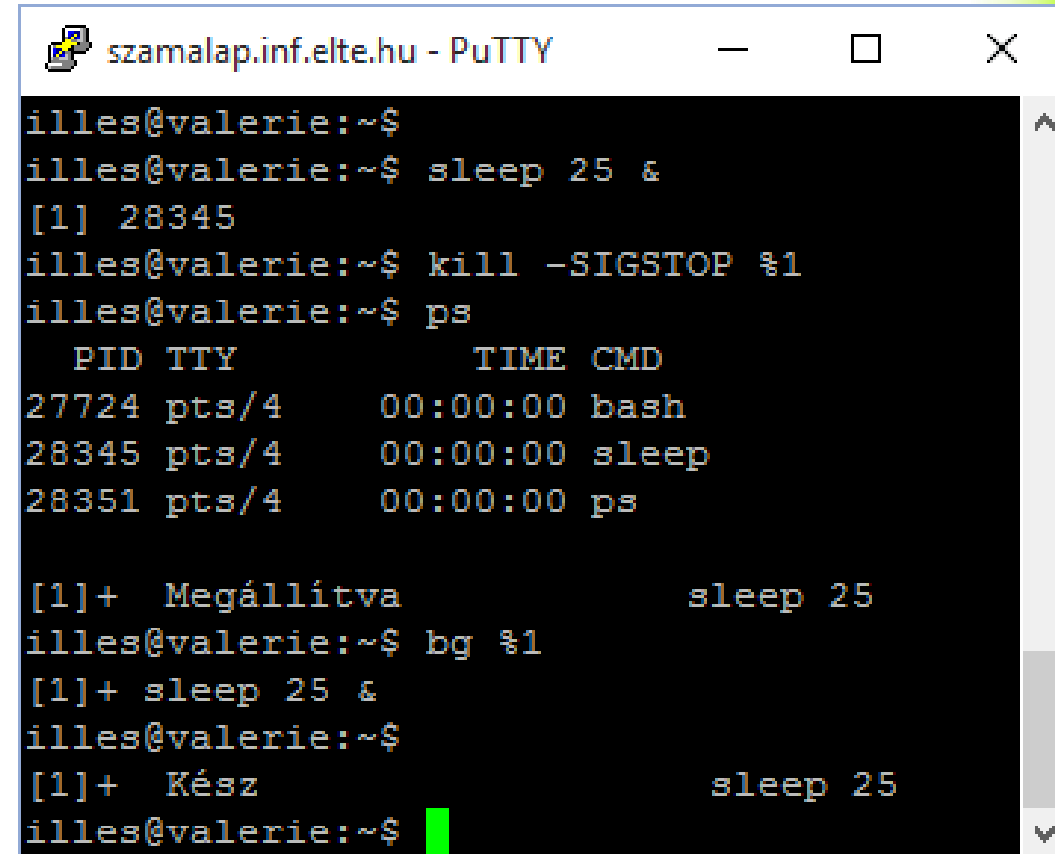
- További parancsok
- Folyamatok előtérben, háttérben
- I/O átirányítás
- Szűrők
- Irány programozni!
- Reguláris kifejezések

További parancsok

- Parancsok előtérben: normál parancskiadás
- Parancs futtatás a háttérben: & karakter a parancs végén
 - & megnevezései: ampersand, and, és jel, at jel
- `sleep 15 # 15` másodperc várakozás, közben nincs parancskiadási lehetőség
 - CTRL+Z stop jelet küld a folyamatnak
- `sleep 15 & #` a várakozó folyamat a háttérben fut, közben újabb parancsot adhatunk ki
 - Eredményül kapjuk: [1] 28321, az első a „jobszám”, a második a pid
- `jobs` – listázza a futó parancsainkat(„jobjainkat”)
- `ps` – process status parancs, láthatjuk a háttérben futó `sleep`-et is

Folyamatok háttérben

- Mi van ha egy háttérfolyamatot előtérbe akarunk tenni: `fg %1`
- Egy előtérben futó folyamatot ha leállítottunk(`ctrl-z`) a háttérben is folytathatjuk: `bg %1`
- Jelzés küldés: `kill –signál folyamat`
 - A folyamatot vagy a job sorszámával (kell a % jel)vagy a processz azonosítóval jelöljük.



```
szamalap.inf.elte.hu - PuTTY
illes@valerie:~$
illes@valerie:~$ sleep 25 &
[1] 28345
illes@valerie:~$ kill -SIGSTOP %1
illes@valerie:~$ ps
  PID TTY          TIME CMD
 27724 pts/4        00:00:00 bash
 28345 pts/4        00:00:00 sleep
 28351 pts/4        00:00:00 ps

[1]+  Megállítva                  sleep 25
illes@valerie:~$ bg %1
[1]+  sleep 25 &
illes@valerie:~$
[1]+  Kész                        sleep 25
illes@valerie:~$
```

Még többet a folyamatokról!

- top parancs: látjuk a futó folyamatok adatait, globális rendszer állapotot!
 - Mindenki egyenrangú?
 - Unix rendszerek prioritása: -20-tól 19-ig (40 szint)
 - A kisebb szám a nagyobb prioritás!
 - nice parancs, módosítja az indítandó parancs prioritását.
 - Alapból a prioritás 0, top vagy ps -l parancs NI oszlop!
 - nice -n 5 sleep 20& # 5 lesz az új prioritás, 0-hoz 5-öt ad
 - nice -n -10 sleep 20& # növelni root joggal lehet!!!!

Folyamatok, prioritások

- A Unix, Linux prioritás alapú folyamatkezelést végez!
- POSIX4 – IEEE1003.1b (1993), Valós idejű kiterjesztés megjelenés
- Két prioritás lista
 - nice -20-19, ahogy láttuk
 - Valós idejű prioritási lista, 0-99 közti értékekkel.(100 prioritási szint)
 - Ebben a listában a nagyobb szám jelenti a nagyobb prioritást
 - A lista közös értelmezése a következő:
 - 99,98...1,-20,-19,..0,1...,19, ezt gyakran 140-es prioritás intervallumnak neveznek, amiben különböző eltolások(mapping) lehetségesek.

Parancsok időzített futtatása

- Unix rendszereken kétféle időzített futtatási támogatás van.
 - cron (démon), rendszergazda jogosítvány kell, vagy a /etc/cron.allow-ban engedély
 - /etc/crontab – rendszer időzítések, /etc/cron.d/ - felhasználói időzítések
 - Ezen időzítések megadják, hogy a cron milyen programokat milyen időpontokban futtasson.
 - at – a parancs használható, ha az atd szolgáltatás fut.
 - at parancs a standard inputról várja a bemenetét(a futtatandó parancsot)
 - at now + 5 minutes <parancsfájl # az at-nek az időpont kell mint paraméter, amit sokféleképpen megadhatunk, lásd man!

Parancsok befejezése

- Normál befejezés – a parancs nem akar tovább futni!.
- Egy parancs maximum addig fut, amíg az indító felhasználó a rendszerben van!
 - Mindegy, hogy el előtérben vagy háttérben indítottuk el!
- **nohup parancs**
 - nohup parancs & # lehet & nélkül is, csak akkor előtérben fut!!!
 - A parancs kimenete a nohup.out állományba kerül!
 - Ez az állomány indításkor létrejön, a későbbi outputot ehhez hozzáfűzi!
 - Saját kimenet állomány is megadható: nohup parancs >sajat.nohup

Idézőjelek

- Parancssorban legális karakterek: .,_,-,számok, normál karakterek
- Idézőjelek: ',",\ – módosítják a klasszikus karakter értelmezést
- 'Aposztróf karakterek között megszűnik minden speciális \ \$ % stb. hatás'
 - pl: `echo 'alma\fa' #alma\fa`
- "Macskakörmön belül a \$, a \, a ` és a ' karakterek hatása megmarad"
 - Példa: `a=fradi; echo "hajrá $a !" # hajrá fradi`
- \x karakter: módosítja x eredeti jelentését
 - Példa: `fa=virág; echo alma\fa #alma$fa`
 - `echo alma$fa #almavirág`

Kimenet, Bemenet, átirányítások

- Kimenet, bemeneti eszközök
 - stdin (0) - billentyűzet, alapértelmezett bemenet
 - stdout (1) - monitor, alapértelmezett kimenet
 - stderr (2) – monitor, alapértelmezett hibakimenet
- Átirányítás
 - Kimenet: > jel segítségével, új állomány létrejön
 - Pl: echo alma barack szilva >gyumolcs #gyumolcs új fájl lesz
 - echo alma >&2 #hogya ne a 2 nevű állomány legyen
 - Bemenet: < jel segítségével
 - Pl: passwd juli <alma; cat alma # almafa, almafa

I/O átirányítások II.

- Kimenet, hozzáfűzés (append)
 - >> fájlnev, Pl: echo dió >>gyumolcs
 - Ha nem létezik a fájl, akkor létre is hozza!
- Hibakimenet (stderr) átirányítás
 - 2>, 2>>
- Szimmetria miatt(😊): <<
 - Bemenet átirányítás a helyben megadott szövegre

```
$ cp 2>hiba  
$ mv ezt 2>>hiba  
$ cat hiba
```

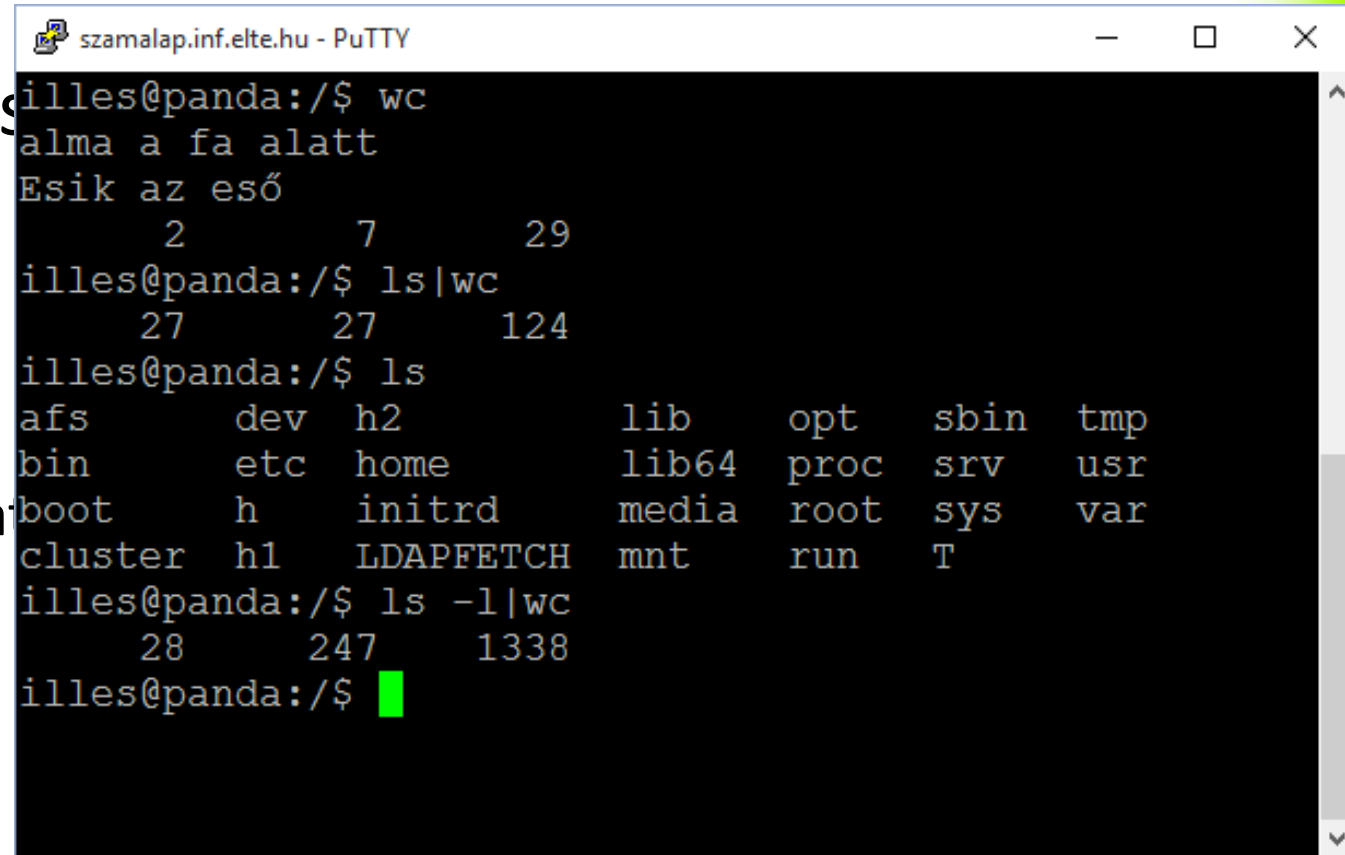
```
cat <<alma  
<input type=text name=X>  
<input type=button>  
alma
```

Szűrők

- Parancs vs. Szűrő
 - Képes egy parancs kimenetét saját bemeneteként fogadni!
- Önmagában egy szűrő nem szűrő!
 - Legalább 2 parancs összekapcsolásáról van szó!
- Műveleti jel: |
- Ilyen nagyon ismert, gyakran használt szűrőparancs például a WC!
 - Word Count!
 - Feladata: sorok, szavak, karakterek megszámlálása!
 - `wc [-lwc] [file]` #Paraméter nélkül várja ctrl-d-ig a bemeneti adatokat!

Szűrők használata

- Használhatók normál parancs alakban, paraméter nélkül, ekkor a bemenetről (billentyűzet) vár adatokat!
 - Input vége: CTRL+D
 - Paraméteres formában módosulhat a bemenet!
 - Pl: `wc alma.fa`
- Szűrő formában, | karakterrel!



The screenshot shows a terminal window titled 'szamalap.inf.elte.hu - PuTTY'. The user 'illes@panda' is in the directory '/'. The following commands and their outputs are shown:

```
illes@panda:/$ wc
alma a fa alatt
Esik az eső
      2      7      29
illes@panda:/$ ls|wc
      27      27      124
illes@panda:/$ ls
afs      dev  h2      lib      opt      sbin  tmp
bin      etc  home    lib64    proc     srv   usr
boot     h    initrd  media    root     sys   var
cluster  h1   LDAPFET mnt      run      T
illes@panda:/$ ls -l|wc
      28      247      1338
illes@panda:/$
```

Fontosabb szűrők

- Fontosabb kész szűrők:
 - tee, tr, cut, sort, uniq, wc, grep, stb.
 - Példa:

```
$ who >nevek
```

```
$ sort nevek #sorok szerinti sorrend
```

```
$ who|sort -r -u # fordított sorrend, egyedi sorok
```

```
$ who|wc -l #bejelentkezett felhasználó szám
```

- Ami biztos: man (ual) lekérdezése
 - Példa: man sort, man -k kulcsszó

Tee -, tr parancsok

- tee – a paraméterként kapott fájl(ok)ba másolja a csővezetéken áthaladó tartalmat.
 - Példa: `ls -l | tee dir.txt | wc -l, who | tee -a users.txt report.txt | sort -r`
 - -a paraméter: append
- tr – translate, paramétere kér karakter csoport, a szabványos bemenetre érkező adatokban lévő első karaktercsoport elemeit a második karaktercsoport elemekre cseréli.
 - Példa: `echo alma | tr am et` # elte
 - Hasonló lehetőség a sed y parancsa! (később látni fogjuk)

CUT - kivágás

- Standard inputon vagy fájlból vághatunk ki mező(ke)t, oszlop(ka)t, sorokból!
- `cut -c1-5` 1-5 karakteroszlop kivágás
 - példa: `date|cut -c4-8` # Oct
- `cut -f1,3,5-7` 1,3,5,6,7 mezők kivágása
 - Alapértelmezett mezőelválasztó: Tab
 - Új mezőelválasztó: `-d char`
 - Példa: `cat /etc/passwd|cut -f1,7 -d: # név, shell`

Grep – sorok szűrése

- A paraméterül adott mintával rendelkező sorok kiválasztása.
- Fontosabb paraméterek:
 - -v mintát nem tartalmazó sorok
 - -i kis és nagybetűket nem különböztet meg
 - -w Csak önálló szóként találja meg (traPista nem)
 - -r Rekurzívan a paraméterül adott könyvtárra.
 - -l Csak a fájl neveket írja ki. (fájlban keres)
 - -c csak a sorok számát írja ki
 - -n megszámozza a sorokat

Grep használata

- Szűrő példa:
 - `cat nevsor | grep Pista` # Eredményül kapjuk a Pista-t tartalmazó sorokat.
- Parancs forma:
 - `grep -r 'fradi' ./script` # script könyvtárban a fradi-s sorokat (fájlokban) keresi
 - `grep -r -l par *` # Az összes állományban, alkönyvtárakban keresi a par szót, eredményül csak a fájl nevét írja ki.

Mintaillesztés, reguláris kifejezések

- Egy szövegminta általános megadása – Reguláris kifejezések-speciális karakterek
 - ^ Sor elejétől kell egyezni a mintának.
 - Pl: '^alma' : a sor elején alma szó áll
 - \$ Sor végétől kell egyezni a mintának.
 - Pl: 'barack\$' : a sor végén a barack szó áll
 - . Egy tetszőleges karakter
 - * Előző minta ismétlése 0 vagy többször!
 - Pl: '^alma.*fa\$ ' - alma és fa között akárhány(0 is lehet) karakter

Példák I.

- `Cat fájl | grep „^$”` # Üres sor kiválasztás
 - Hány üres sor van egy fájlban? `Cat fájl | grep „^$” | wc -l`
- `Cat fájl | grep „^$ FTC.*\$$”` # Sor elején, végén \$, első \$ után FTC, utána bármi!
- `Cat fájl | grep „/-”` # Valahol a sorban /- karakterek.
- `Cat fájl | grep „- /”` # Hiba, mert – jelet parancs kapcsolónak tekinti, és nincs / jelű kapcsoló!
- `Cat fájl | grep „\ - /”` # Ez már OK, a - / karakterek bárhol szerepelhetnek!

További mintaillesztés

- Karakterhalmazok megadása: [FTC] # Vagy F vagy T vagy C
 - Karakter intervallum megadása(- jel): [a-z] (kisbetű)
 - [^a-c] Nem a, b vagy c betű
 - [A-Za-z0-9] Alfabetikus (szám vagy betű)
 - \w Alfabetikus, mint előző
 - \W Nem alfabetikus
 - \d számjegy, azonos a [0-9]-el
 - \s szóköz, tab, sortörés
- Szavak illesztése
 - \< Szókezdet, Pl: grep "\<Zol"
 - \> Szóvég, Pl: grep "tán\>"
 - \b Szó elején, végén helyez Pl: grep '\bpista\b'

Példák II.

- `Cat fájl | grep [-a]` # A – vagy az a betű keresése
 - `Cat fájl | grep [a-]` # Ugyanaz. A – az első vagy utolsó helyen állhat!
 - `Cat fájl | grep [a-c]` # Az a-c intervallum, azaz a vagy b vagy c betű
 - `Cat fájl | grep [-ac]` # A – vagy a vagy c betű
- `Cat fájl | grep [FTC]{})` # Olyan minta keresés, ami vagy az egyik betű ÉS utána {)}
- `Cat fájl | grep [FTC\\]{})` # [] jelen belül nincs speciális jelentés, pl `\d` vagy `.`
- `Cat fájl | grep []FTC{)}` # Vagy] vagy ...

E(F)Grep – mintaillesztés I.

- egrep – bővített grep -E, létezhetnek különböző implementációk
- fgrep - fixed grep, grep -F, minták sorokban találhatóak
 - `ali|éva` , vagy kapcsolat,
 - Példa: `ls | egrep "par|pelda"`
 - + Előző minta legalább egyszer
 - Példa: `ls | egrep "\<p+f"` # szó elején 1 vagy több p, majd f betű.
 - ? Előző minta nulla vagy egyszer ismétlődik
 - Példa: `ls|egrep „param\d?”` #param, param1, param2,..
 - {n} előző karakter pontosan n szer!
 - Példa: `cat almafa|egrep ^[a-b]\w{5}`

E(F)Grep – mintaillesztés II.

- {2,4} Előző minta 2,3 vagy 4-szer ismételve
 - {1,} Előző minta legalább egyszer
- () Egy csoportba fogunk egy mintát.
 - Ismétlődéshez célszerű, azaz ezt követi egy ismétlésre vonatkozó utasítás +,{n}
 - Példa: `[0-9]{8}(\s[0-9]{8}){1,2} #bankszámla`
 - A `[0-9]` helyett `\d` is jó lenne.
- Speciális karaktert célszerű `\` mögé írni.
 - Példa: `\.$ #A sor végén pont van!`
 - Példa: `^[+-]? \d+ ([,.] \d+)? #előjeles szám tizedes ponttal, vagy vesszővel, nem kell \ mert [] belül van!`
- Többi lehetőséghez: man

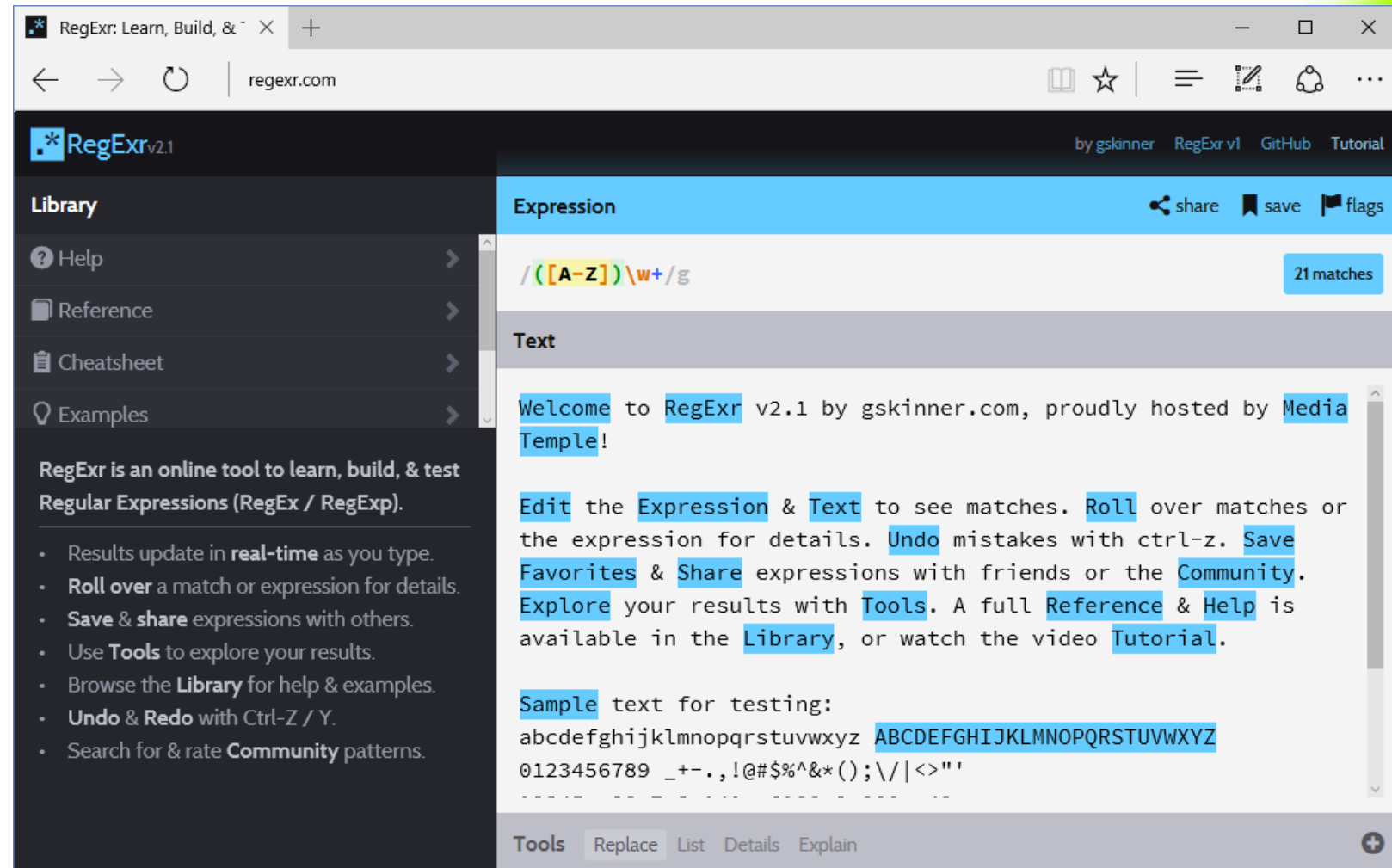
További minták

- E-mail cím(rövidebb változat):
 - `\w+[-._]?.*@\w+(\.\w+)+`
- Óra:perc:másodperc
 - `(([01][0-9])|(2[0-3])):[0-5][0-9]:[0-5][0-9]`
- Stb.
- Minden programozási nyelv tartalmazza ezt vagy ennek kicsit bővített, módosított lehetőségeit!

Reguláris kifejezések -online

- Több online rendszer is elérhető

- <http://regexpr.com>
- <http://regex101.com>
- Stb.



Szövegszerkesztők – I.

- Széles skála áll rendelkezésre!
- Jellemzően: vi(m), pico, mcedit, joe, stb.
- Alapvető használatukban a segítség a képernyőn!(joe)

```
szamalap.inf.elte.hu - PuTTY

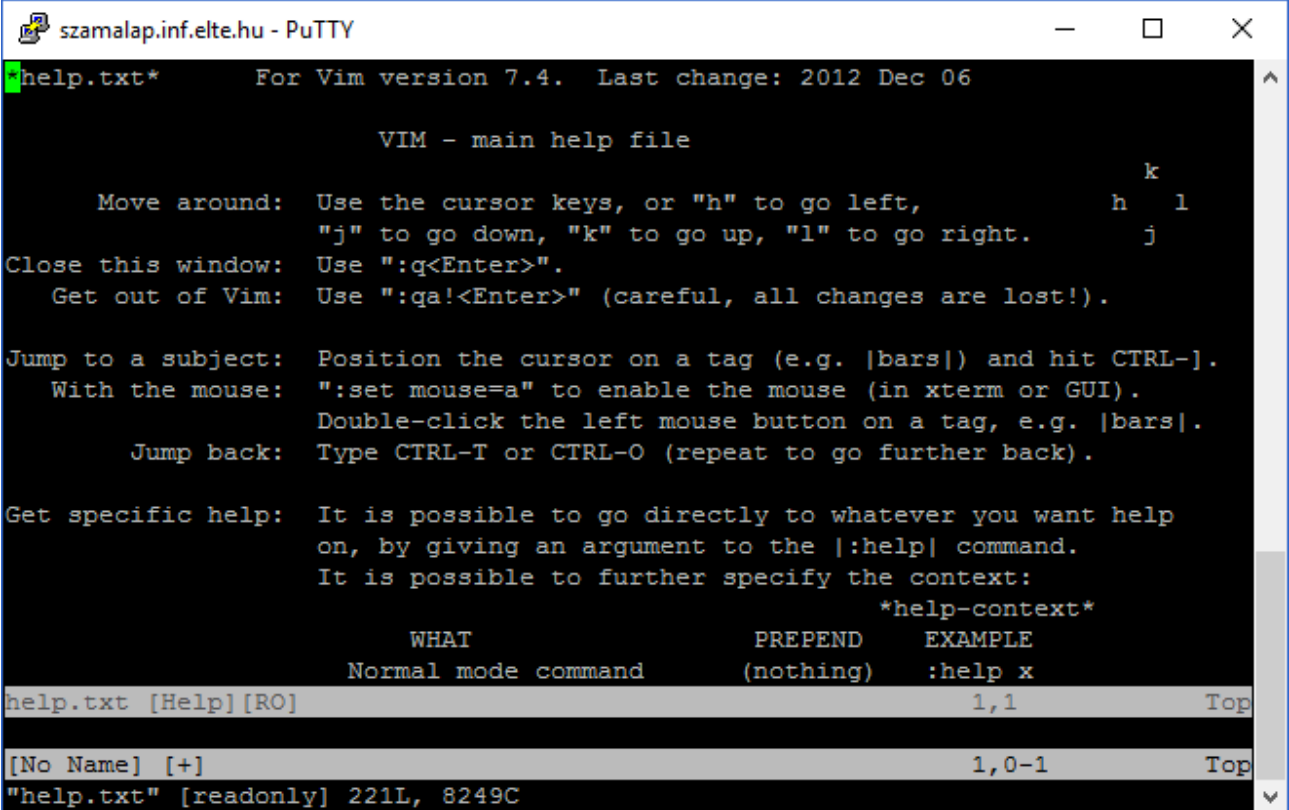
Help Screen      turn off with ^KH      more help with ESC . (^[.)

CURSOR          GO TO          BLOCK          DELETE          MISC          EXIT
^B left ^F right ^U prev. screen ^KB begin ^D char. ^KJ reformat ^KX save
^P up ^N down ^V next screen ^KK end ^Y line ^KA center ^C abort
^Z previous word ^A beg. of line ^KM move ^W >word ^T options ^KZ shell
^X next word ^E end of line ^KC copy ^O word< ^R refresh FILE
SEARCH          ^KU top of file ^KW file ^J >line SPELL ^KE edit
^KF find text ^KV end of file ^KY delete ^_ undo ^[N word ^KR insert
^L find next ^KL to line No. ^K/ filter ^^ redo ^[L file ^KD save

IW Unnamed Row 1 Col 1 8:38 Ctrl-K H for help
```

Szövegszerkesztők – II.

- Néhány vi jellemző!
 - Kétféle üzemmód, parancs vagy szerkesztő mód!
 - Parancsmódból szerkesztő módba: i, a, o, stb
 - Szerkesztő módból parancs módba: ESC
 - Parancs módban: mentés: w név, mentés és kilépés: wq (quit)
 - :help – a képen látható
 - Kilépés help-ből: :q
 - Kilépés mentés nélkül: q!



```
szamalap.inf.elte.hu - PuTTY
help.txt*      For Vim version 7.4.  Last change: 2012 Dec 06

          VIM - main help file

Move around:  Use the cursor keys, or "h" to go left,      k
              "j" to go down, "k" to go up, "l" to go right.  h  l
              Use ":q<Enter>".                                j
Close this window:
Get out of Vim: Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. |bars|) and hit CTRL-].
With the mouse:   ":set mouse=a" to enable the mouse (in xterm or GUI).
                  Double-click the left mouse button on a tag, e.g. |bars|.
Jump back:        Type CTRL-T or CTRL-O (repeat to go further back).

Get specific help: It is possible to go directly to whatever you want help
                   on, by giving an argument to the |:help| command.
                   It is possible to further specify the context:
                                     *help-context*
                   WHAT                PREPEND    EXAMPLE
                   Normal mode command (nothing)  :help x
help.txt [Help] [RO]                                     1,1      Top
[No Name] [+]                                           1,0-1    Top
"help.txt" [readonly] 221L, 8249C
```


Köszönöm a figyelmet!

