Szűrők

A szűrők olyan programok, amelyek egy adatfolyam feldolgozására hivatottak. Alapvető működésű elvük, hogy a standard bemeneten fogadják az adatokat, majd az eredményt a standard kimenetre irányítják. Legalapvetőbb használatuk épp emiatt a csővezetékekben történik. Természetesen ettől el lehet térni - a már korábban említett - átirányítási lehetőségek segítségével felül definiálhatjuk a bemenetet és a kimenetet egy eszközzel vagy fájllal.

cat

Minden argumentumként megadott fájl(oka)t összefűz, és a standard kimenetre írja a végeredményt. Amennyiben nincs fájlnév megadva, úgy a standard bemenetről olvas. Példa a cat parancs használatára:

```
adamkoa@it:~$ cat > x.txt
átirányítja a standard
outputot az x.txt
fájlba                          # bevitel befejezése CTRL+D billentyűkombinációval
adamkoa@it:~$ cat
Vajon miért ír ki mindent 2x? :-)
Vajon miért ír ki mindent 2x? :-)
adamkoa@it:~$
```

A UNIX világában számos apró kis névjáték van, így alakult ki a `tac` programocska is, ami a cat szó fordítottja - és a működése is a fordítottja, azaz a sorokat az utolsótól az elsőig írja ki.

colrm

A standard inputon érkező szöveg megadott oszlopait törli, majd az így kapott szöveget a standard outputra írja.

Szintaktika:

```
colrm [kezdőindex] [végindex]
```

Példa a colrm parancs használatára:

```
adamkoa@it:~$ cat nevek.txt                 # kiíratás
Áron 5 34
Attila 3 20
Barna 5 31
Betti 2 8
adamkoa@it:~$ cat nevek.txt | colrm 2 4     # törlünk a 2-tól az 5. oszlopig
Á 5 34
Ala 3 20
Ba 5 31
Bi 2 8
adamkoa@it:~$
```

cut

A standard inputon (vagy a megadott fájlból) érkező szöveg soraiból törli a megadott sorszámú karaktereket, majd az eredményt a standard outputra írja.

Szintaktika:

```
cut [kapcsoló] [indexek] [fájlnév]
```

Kapcsoló: -c :Az eredeti működés ellentéte. Csak a megadott sorszámú karaktereket írja ki.

Példa a cut parancs használatára:

```
adamkoa@it:~$ cut 3           # kiírja a 3. karaktert törli
abcdefgh
abdefgh
adamkoa@it:~$ cut -c 3        # kiírja a 3. karaktert az adatfolyamban
abcdefgh
c
adamkoa@it:~$ cut -c 1,3-8    # kiírja az 1 helyen és 3-tól 8-ig található
karaktereket
abcdefghijk
acdefgh
adamkoa@it:~$
```

Kapcsoló: -d : A cut alapesetben a TAB karaktert használja a bemeneten érkező karakterfolyam mezőkre bontására, amennyiben ettől eltérőt szeretnénk használni, akkor ezt a kapcsolót használva megadhatjuk a mezőket egymástól elválasztó karaktert.

Kapcsoló: -f : amennyiben a bemenet tagolható, akkor az egyes részekre, amelyet tovább szeretnénk engedni a kimenetre, azokat itt kell felsorolni.

```
[adamkoa@kkk ~]$ cat /etc/passwd | tail
n4sadm:x:512:515:SAP System Administrator:/home/n4sadm:/bin/csh
sdb:x:513:514:MmaxDB Software Owner:/home/sdb:/bin/bash
sqdn4s:x:514:515:SAP Database Administrator:/home/sqdn4s:/bin/csh
arato:x:501:501:Arató Mátyás:/home/arato:/bin/bash
uuidd:x:104:105:UUID generator helper daemon:/var/lib/libuuid:/sbin/nologin

[adamkoa@kkk ~]$ cat /etc/passwd | cut -d: -f1,5,7 | tail
n4sadm:SAP System Administrator:/bin/csh
sdb:MmaxDB Software Owner:/bin/bash
sqdn4s:SAP Database Administrator:/bin/csh
arato:Arató Mátyás:/bin/bash
uuidd:UUID generator helper daemon:/sbin/nologin
[adamkoa@kkk ~]$
```

grep

A grep valójában nem egy, hanem három parancsot takar: grep, az egrep (extended) és az fgrep (fast). Grep a megnevezett bemeneti fájlokban a megadott mintához illeszkedő sorokat keres. Amennyiben nincs bemenő fájlnév megadva, a standard bemenetet olvassa.

Szintaktika:

`grep [minta] [fájlnév] [kapcsolók]`

Alapértelmezés szerint grep a mintához illeszkedő sorokat a standard outputra írja. A mintát a fentebb leírt reguláris kifejezésekkel lehet megadni.

Kapcsolók:

- -c : elhagyja a szokásos kimenetet, ehelyett az illeszkedést mutató sorok számát írja ki minden fájl esetére
- -i : nem különbözteti meg a kis- és nagybetűket sem a mintában, sem a bemeneti fájlban

- -l : elhagyja a szokásos kimenetet, és csak azon fájlok neveit adja meg, amelyekből származna kimenet
- -v : megfordítja az illeszkedés értelmét: a mintához nem illeszkedő sorokat választja ki
- -w : csak azokat a sorokat választja ki, amelyekben az illeszkedés teljes szavakból származik. Azaz az illeszkedést mutató szövegrész előtt és után nem állhat betű, szám vagy aláhúzásjel.

A minta lehet egyszerű sztring, és használhatók metakarakterek is. Ha a mintát megtalálja egy sorban akkor azt mondjuk, a minta illeszkedik.

Minta metakarakterek

A minta felépítéséhez a grep a reguláris kifejezések nyelvét használja, amely a Perl konvencióját alkalmazza. Lássuk az egyes jeleket és azok jelentését:

Metakarakterek:

karakter	jelentése
.	Bármely karakter Pl.: M.m illeszkedik minden karaktersorra ahol a nagy M és a kis m betű között pontosan egy bármilyen karakter van.
[felsorolás]	A felsorolásban lévő bármelyik karakterre illeszkedik. Pl.: [aeiou] Bármely angol magánhangzóra illeszkedik.
[tól – ig]	A határok által megadott karakterek mindegyikére illeszkedik. Pl.: [a-z] Bármely a és z közötti bármelyik karakterre illeszkedik.
[^minta]	Bármely karakterre illeszkedik a mintát alkotó karakterek kivételével (negálás)
^	A sor eleje. Pl.: ^A Csak a sor elején álló nagy A betűre illeszkedik
$	A sor vége. Pl ^$ Üres sorokra (sor eleje után rögtön a vége) illeszkedik.
[:karakterosztály:]	előre definiált karakterosztályok vannak (lásd man grep) ● [:alpha:] betűkre illeszkedik ● [:lower:] kisbetűkre illeszkedik ● [:upper:] nagybetűkre illeszkedik ● [:digit:] számjegyekre illeszkedik stb.

Ismétlődés jelzők:

karakter	jelentése	példa
?	előző tag 0-szor vagy egyszer	Pl: al?ma Illeszkedik az alma és az ama karaktersorra.
*	előző tag 0-szor vagy többször	Pl.: al*ma illeszkedik: ama alma, allma, alllma stb.
+	előző tag 1 szer vagy többször	Pl.: al+ma u.a. mint előbb, de az ama nem illeszkedik
{n}	előző tag pontosan n szer	Pl.: k{8} pontosan 8 k betűhöz illeszkedik

karakter	jelentése	példa
{n,}	előző tag legalább n-szer vagy többször	
{ ,m}	előző tag maximum m-szer	
{n,m}	előző tag legalább n-szer legfeljebb m-szer	Pl.: r{2,4} legalább kettő, maximum 4 egymás melletti r betűre illeszkedik

head

Kiírja az állomány első néhány sorának tartalmát a standard outputra.

Szintaktika:

```
head [filenév]
```

paste

A paraméteréül kapott fájlok sorait egymás mellé fűzi és az eredményt a standard kimenetre írja.

Szintaktika:

```
paste [fájlnév1] [fájlnév2]
```

rev

Visszafele kiírja a bemenetként kapott sorokat.

Szintaktika:

```
rev [fájlnév]
```

Példa a rev parancs használatára:

```
adamkoa@it:~$ rev nevek.txt
norÁ
alittA
anraB
itteB
icaL
ótáneR
adamkoa@it:~$
```

sed

Folyamszerkesztő (stream editor), tulajdonképpen egy nem interaktív programozható szövegszerkesztő/manipuláló program. A program a megnevezett fájlokat (alapértelmezés szerint a standard bemenetet) a standard kimenetre másolja, de közben egy parancsokat tartalmazó szkriptnek megfelelően megszerkeszti.

Használat:

```
sed [kapcsolok] [script] [fajl]
```

Kapcsolók:

- -n : Alapértelmezésben a SED igen sokat ír a képernyőre, minden sor után folyamatosan tájékoztatva a felhasználót. Ezen kapcsoló megadásakor csak akkor ír a kimenetre mikor a szkriptben erre a p módosítóval utasítást adunk.

A szkript:

A szkript nagy általánosságban sed 's/mit/mire/módosító' alakú. A sed végignézi az összes sort (egyenként), és ha talál olyan szövegrészt, ami illeszkedik a "mit" mintára, lecseréli a "mire" mintának megfelelő szövegre. A szkriptben a "mit" helyén az összes fentebb ismertetett reguláris kifejezés használható a keresett minta definiálására. Ha nem adunk meg "módosító"-t, akkor hiába van több olyan szövegrész is a sorban, amire illeszkedik a "mit" minta , csak az elsőt cseréli!

A sed 's/mit/mire/N' csak az N-edik előfordulást cseréli (minden sorban újrakezdi a számolást), míg a sed 's/mit/mire/g' az összes előfordulást cserélni.

```
# A p (kiíratás) módosító használata: adamkoa@it:~$ cat nevek.txt | sed -n
'2,5p'  # csak a 2-tól az 5. sorig írja ki Attila Barna Betti Laci
adamkoa@it:~$ sed -n '/Zoli/p' nevek.txt     # csak a "Zoli" mintának megfelelő
sorokat írja ki
Zoli

adamkoa@it:~$ ls -l | sed -n '/xy.*/p'       # csak az "xy"-nal kezdődő
fájlneveket írja ki
-rwx------ 1 adamkoa prog1 16589 2007-02-12 18:26 xy
-rw-r--r-- 1 adamkoa prog1 61    2007-02-12 18:22 xy.c
-rw------- 1 adamkoa prog1 196   2007-02-12 18:26 xy.log
-rw------- 1 adamkoa prog1 6     2007-02-12 18:26 xy.out
adamkoa@it:~$

# A d (törlés) módosító használata:

adamkoa@it:~$ ls -l | sed '/xy.*/d'          # csak azokat a fájlneveket írja
ki, amelyek nem "xy"-nal kezdődnek
összesen 36
-rw------- 1 adamkoa prog1 66  2007-04-26 14:24 nevek.txt
-rw------- 1 adamkoa prog1 0   2007-04-26 16:22 x.txt
drwx------ 2 adamkoa prog1 144 2007-04-12 15:10 zh2
adamkoa@it:~$

# Az s (csere) módosító használata:

adamkoa@it:~$ ls -l | sed 's/ /:/g'          # minden szóközt ":"-ra cserél, a /
g biztosítja az összes előfordulás cseréjét
összesen:36
-rw-------:1:adamkoa:prog1::::66:2007-04-26:14:24:nevek.txt
-rw-------:1:adamkoa:prog1:::::0:2007-04-26:16:22:x.txt
-rwx------:1:adamkoa:prog1:16589:2007-02-12:18:26:xy
-rw-r--r--:1:adamkoa:prog1::::61:2007-02-12:18:22:xy.c
-rw-------:1:adamkoa:prog1:::196:2007-02-12:18:26:xy.log
-rw-------:1:adamkoa:prog1:::::6:2007-02-12:18:26:xy.out
drwx------:2:adamkoa:prog1:::144:2007-04-12:15:10:zh2
adamkoa@it:~$ ls -l | sed 's/ /:/2'          # csak a 2. előfordulást cseréli
összesen 36
-rw------- 1:adamkoa prog1 66 2007-04-26 14:24 nevek.txt
-rw------- 1:adamkoa prog1 0 2007-04-26 16:22 x.txt
-rwx------ 1:adamkoa prog1 16589 2007-02-12 18:26 xy
-rw-r--r-- 1:adamkoa prog1 61 2007-02-12 18:22 xy.c
-rw------- 1:adamkoa prog1 196 2007-02-12 18:26 xy.log
-rw------- 1:adamkoa prog1 6 2007-02-12 18:26 xy.out
drwx------ 2:adamkoa prog1 144 2007-04-12 15:10 zh2
adamkoa@it:~$
```

sort

Szövegfájl sorainak rendezése.

Szintaktika:

`sort [kapcsolók] [be_fájlnév] [ki_fájlnév]`

Kapcsolók:

- -o : az eredményt a kimeneti állományba írja az alapértelmezés szerinti kimenet helyett
- -r : visszafele (csökkenő sorrendbe) rendez
- -n : Szöveg elején álló számok numerikus összehasonlítása. A számok előjelből és 0 vagy több számjegyből, és tizedespont utáni további számjegyekből állhatnak
- -u : A bemenet ismétlődő sorai a kimeneten pontosan csak egyszer fognak szerepelni
- -k : a rendezés alapjául szolgáló kulcs oszlop megadása, ha elhagyjuk, akkor az egész sort tekinti egybe.

```
[adamkoa@kkk proba]$ ls -l
total 60
-rw-rw-r--  1 adamkoa adamkoa    0 May  9 15:32 core
-rw-rw-r--  1 adamkoa adamkoa    0 Mar  9  2010 link2.txt
-rw-r-----  1 adamkoa adamkoa    0 Mar 23  2010 link.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file2.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file.txt
-r--rw-r--  1 adamkoa adamkoa   25 Mar  9  2010 p2.txt
-rw-rw-r--  1 adamkoa adamkoa   26 Apr 20  2010 sed.txt
lrwxrwxrwx  1 adamkoa adamkoa    8 Apr 27  2010 szimbolikus.txt -> link.txt
drwxrwxr-x  2 adamkoa adamkoa 4096 Apr 27  2010 test
lrwxrwxrwx  1 adamkoa adamkoa    5 Apr 27  2010 tmp -> /tmp/
[adamkoa@kkk proba]$ ls -l |sort -k 5
total 60
-rw-r-----  1 adamkoa adamkoa    0 Mar 23  2010 link.txt
-rw-rw-r--  1 adamkoa adamkoa    0 Mar  9  2010 link2.txt
-rw-rw-r--  1 adamkoa adamkoa    0 May  9 15:32 core
-r--rw-r--  1 adamkoa adamkoa   25 Mar  9  2010 p2.txt
-rw-rw-r--  1 adamkoa adamkoa   26 Apr 20  2010 sed.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file2.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file.txt
drwxrwxr-x  2 adamkoa adamkoa 4096 Apr 27  2010 test
lrwxrwxrwx  1 adamkoa adamkoa    5 Apr 27  2010 tmp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    8 Apr 27  2010 szimbolikus.txt -> link.txt
[adamkoa@kkk proba]$ ls -l |sort -k 5 -n

-rw-r-----  1 adamkoa adamkoa    0 Mar 23  2010 link.txt
-rw-rw-r--  1 adamkoa adamkoa    0 Mar  9  2010 link2.txt
-rw-rw-r--  1 adamkoa adamkoa    0 May  9 15:32 core
total 60
lrwxrwxrwx  1 adamkoa adamkoa    5 Apr 27  2010 tmp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    8 Apr 27  2010 szimbolikus.txt -> link.txt
-r--rw-r--  1 adamkoa adamkoa   25 Mar  9  2010 p2.txt
-rw-rw-r--  1 adamkoa adamkoa   26 Apr 20  2010 sed.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file2.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file.txt
drwxrwxr-x  2 adamkoa adamkoa 4096 Apr 27  2010 test
[adamkoa@kkk proba]$
```

Több oslzop szerint is lehet rendezni:

```
[adamkoa@kkk proba]$ ls -l |sort -k5n -k9
total 64
-rw-rw-r--  1 adamkoa adamkoa    0 May  9 15:32 core
-rw-rw-r--  1 adamkoa adamkoa    0 Mar  9  2010 link2.txt
-rw-r-----  1 adamkoa adamkoa    0 Mar 23  2010 link.txt
lrwxrwxrwx  1 adamkoa adamkoa    5 Jun  5 15:21 temp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    5 Apr 27  2010 tmp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    8 Apr 27  2010 szimbolikus.txt -> link.txt
-r--rw-r--  1 adamkoa adamkoa   25 Mar  9  2010 p2.txt
-rw-rw-r--  1 adamkoa adamkoa   26 Apr 20  2010 sed.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file2.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file.txt
drwxrwxr-x  2 adamkoa adamkoa 4096 Apr 27  2010 test
[adamkoa@kkk proba]$ ls -l |sort -k5n -k9r
-rw-r-----  1 adamkoa adamkoa    0 Mar 23  2010 link.txt
-rw-rw-r--  1 adamkoa adamkoa    0 Mar  9  2010 link2.txt
-rw-rw-r--  1 adamkoa adamkoa    0 May  9 15:32 core
total 64
lrwxrwxrwx  1 adamkoa adamkoa    5 Apr 27  2010 tmp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    5 Jun  5 15:21 temp -> /tmp/
lrwxrwxrwx  1 adamkoa adamkoa    8 Apr 27  2010 szimbolikus.txt -> link.txt
-r--rw-r--  1 adamkoa adamkoa   25 Mar  9  2010 p2.txt
-rw-rw-r--  1 adamkoa adamkoa   26 Apr 20  2010 sed.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file.txt
-rw-rw-r--  1 adamkoa adamkoa   30 Apr 14  2010 new_file2.txt
drwxrwxr-x  2 adamkoa adamkoa 4096 Apr 27  2010 test
[adamkoa@kkk proba]$
```

uniq

Az egymást követő ismétlésekkel dolgozik. Az inputon érkező rendezett adathalmazból alapértelmezésben eltünteti a duplikált sorokat.

Szintaktika:

`uniq [kapcsolók] [fájlnév]`

Kapcsolók:

- -u : csak a nem azonos sorokat írja ki
- -d : csak a duplikált sorokat írja ki
- -c : kiírja a sor elé, hogy az adott sor hányszor fordult elő

wc

A Word Count bájtok (karakterek), szavak (whitespace karakterekkel tagolva), sorok megszámlálására alkalmas.

Szintaktika:

`wc [kapcsolók]`

Kapcsolók:

- -c : karaktereket számol
- -w : szavak számol
- -l : sorokat számol

Példa a wc parancs használatára:

```
adamkoa@it:~$ wc nevek.txt
10 11 66 nevek.txt
adamkoa@it:~$
```

tail

Kiírja az állomány utolsó néhány sorának tartalmát a standard outputra.

Szintaktika:

`tail [kapcsolók] [filenév]`

Kapcsolók:

- -f : hozzákapcsolódik az állományhoz és mikor az állomány tartalma megváltozik akkor kiírja az utolsó sorait. Ez igen hasznos funció aktív rendszereken az aktív folyamatok log állományit nyomonkövetni, úgy hogy közben más folyamatot is futtathatunk feltéve ha a tail-t háttérfolyamatként indítottuk el.

tr

A tr karakterek lecserélésére, vagy törlésére használható.

Szintaktika:

`tr [mit] [mire]`

Példa a tr parancs használatára:

```
adamkoa@it:~$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
Megyek, iszom valamit.
MEGYEK, ISZOM VALAMIT.
adamkoa@it:~$ tr abcd AZ
ab
AZ
abc
AZZ
abcd
AZZZ
adamkoa@it:~$
```

tee

Szintaktika:

`tee [kapcsoló] [fájlnév]`

Az adatokat a standard inputról veszi. Paramétere egyetlen fájl amibe a beérkező adatot beleírja, majd a standard outputon az adatot továbbdobja. A -a kapcsoló használata esetén a fájlt folytatja (append), ellenkező esetben a tartalmát törli.

Példa a tee parancs használatára:

```
adamkoa@it:~$ date | tee ido.txt | ...
adamkoa@it:~$
```

Az aktuális idő az "ido.txt" állományba kerül majd a pipe fut tovább a dátummal.