

Filtres

Els comandaments sort, tr, grep i cut són el que s'anomenen filtres, és a dir, obtenen una entrada, la alteren d'alguna forma, i extreuen una sortida. S'utilitzen, com el comandament grep vist a la pràctica anterior, amb combinació amb altres comandaments, que habitualment els ofereixen una entrada de dades, si bé també poden utilitzar-se independentment.

SORT

sort és un filtre que permet ordenar línia a línia allò que li passem. Com tots els filtres rep la seva entrada, processa i genera la seva sortida.

Un exemple adequat és el d'ordenar el fitxer /etc/passwd. Aquest fitxer conté tots els usuaris del nostre sistema, junt amb altra informació com una descripció, el tipus de terminal que emprà, el seu UID i GID (identificadors d'usuari i de grup respectivament), etc ... Cada camp d'aquest fitxer està separat per ":".

```
chicken:x:506:100:Heike Jurzik:/home/chicken:/bin/bash
```

Nom d'usuari (login)

Contrassenya: si és "x" vol dir que es troba a /etc/shadow)

UID (User ID), per a usuaris normal és >100

GID (ID de grup)

Informació addicional (nom, núm. telf, etc)

Directori home de l'usuari

Shell que emprà

Per comprovar si un fitxer està ordenat

```
$ sort -c /etc/passwd
```

```
sort: /etc/passwd:2: unsorted  
bin:x:1:1:bin:/bin:/bin/bash
```

Ens diu que el fitxer desordenat i que la primera línia desordenada és la 2.

La forma més simple d'ordenar un fitxer. Ordenarà pel primer camp (el nom d'usuari)

```
$ sort /etc/passwd >passwd_sort
```

Amb -r ens ordenarà a la inversa.

```
$ sort -r /etc/passwd >passwd_sort
```

I en el cas de que tinguem línies repetides, l'opció -u (unic) no les replica.

```
$ sort -u /etc/passwd >passwd_sort
```

Suposem que volem ordenar per un camp diferent de la primera lletra (en aquest cas la descripció, subratllat a l'exemple).

```
$ sort -t: -k5 /etc/passwd  
o  
$ sort -t"." -k5 /etc/passwd
```

```
murie:x:500:500:Manuel Muriel Cordero:/home/murie:/bin/bash  
root:x:0:0:root:/root:/bin/bash  
practica:x:501:501:Usuari de pràctiques per a Ksh:/home/practica:/bin/ksh wi-  
zard:x:502:502:Wizard per a nethack:/home/wizard:/bin/bash
```

-t és l'opció que indica quin separador s'utilitza (dos punts). **-k5** es refereix a la columna per la qual ordenarem (la cinquena).

Ara suposa que disposem d'un fitxer (deuen) en el qual s'arxiven les persones i els deutes en qüestió de préstecs que té amb tu. Un exemple:

Fitxer deuen

```
Joan Nopagamai:23450  
Maria Tacanya:4570  
Pep Gorron:356700  
Anna Notincmaidiners:700
```

Si vols saber a qui has d'enviar primer l'home del frac hauràs d'obtenir una sortida ordenada segons morosos.

Si es realitza aquesta prova es pot observar que:

```
$ sort -t: -k2 deuen
```

```
Joan Nopagamai:23450  
Pep Gorron:356700  
Maria Tacanya:4570  
Anna Notincmaidiners:700
```

Això no és precisament el que s'espera degut a que s'ha realitzat una cerca alfabètica amb un número diferent de dígit. La solució es troba a l'opció **-n**:

```
$ sort -n -t: -k2 deuen
```

Anna Notincmaidiners:700
Maria Tacanya:4570
Joan Nopagamai:23450
Pep Gorron:356700

+n.m se salta els **n** primers camps i els **m** següents caràcters abans de començar l'ordenació.

\$ sort -t: -k2.2 deuen

Anna Notincmaidiners:700
Joan Nopagamai:23450
Pep Gorron:356700
Maria Tacanya:4570

Observa el '\;' que hi ha al següent exemple. Per què creus que el -t no ve únicament acompanyat del ; ?

\$ sort -t\; -k2.2 deuen

Anna Notincmaidiners;700
Joan Nopagamai;23450
Pep Gorron;356700
Maria Tacanya;4570

Si volem ordenar per més d'un camp podem emprar més d'una opció -k per a indicar les diferents columnes; fins i tot podem ordenar per rangs de columnes.

Modifiquem el fitxer deuen

1 Joan Nopagamai 23450
2 Maria Tacanya 4570
3 Pep Gorron 356700
4 Anna Notincmaidiners 700
5 Joan Notincmaidiners 777
6 Pere Tacanya 666

Amb la següent ordre ordenem pel tercer camp (cognom) i després pel segon

\$ sort -k 3 -k 2,2 deuen2

Observa la notació -k 2,2 . Quan ordenem per un camp, en realitat ordenem per aquest camp i els que segueixen. Amb 2,2 limitem al segon camp (de camp 2 a camp 2), ignorant la resta dels camps següents.

Podríem restringir més l'ordenació anterior amb:

```
$ sort -k 3,3 -k 2,2 deuen2
```

Més exemples d'ordenació:

```
$ sort -t ":" -k 4n,4 -k 3nr,3 /etc/passwd  
$ sort -k 3.4,4.5 /etc/passwd
```

A continuació tens alguns dels modificadors dels paràmetres:

- b ignora espais en blanc inicials
- f ignora la distinció entre majúscules i minúscules.
- n ordre aritmètic
- r ordre invers

TR

L'ordre tr reemplaça caràcters en arxius de text. Per a això llegeix l'entrada estàndard i envia els resultats a la sortida estàndard. Per descomptat podem utilitzar els operadors tradicionals per a redirigir ambdós fluxos. De fet a tr se li treu tot el profit quan es combina amb altres comandaments. Passa el mateix amb altres filtres com sort o grep.

Reemplaçaments simples

El comandament tr rep dues cadenes com a argument i reemplaça totes les coincidències del primer argument amb el segon argument. Veiem-ho amb un exemple:

```
$ echo Madagascar | tr 'a' 'e'
```

```
Medegescer
```

Podem obtenir la cadena d'un fitxer de text:

```
$ tr '1' '2' <prova.txt
```

Llavors reemplaçarem qualsevol '1' que aparegui a prova.txt per '2' i enviarem el resultat cap a la sortida estàndard (pantalla). Hem de tenir en compte que qualsevol caràcter que passem com argument s'interpreta de manera independent, és a dir, cada caràcter es reemplaça per la seva contrapart en el segon argument. D'aquesta forma

```
$ tr 'abc' 'xyz'
```

Reemplaçarà les 'a' per 'x' , les 'b' per 'y' i les 'c' per 'z'. Si la segona cadena és més curta que la primera, tr substituirà els que no tinguin contrapart pel darrer caràcter del segon argument.

Per exemple, si escrivim

```
$ tr 'abc' 'z'
```

reemplaçarem cada 'a', cada 'b' i cada 'c' per 'z'. Malgrat tot, tr és incapaç de substituir correctament "ä" per "ae". Com que no són cadenes de la mateixa longitud, cada "ä" se substitueix només per "a". Per a aquest tipus de substitució es recomana utilitzar sed, que veurem més endavant.

Majúscules/minúscules

Imaginem un fitxer prova.txt amb el següent contingut:

```
hola mon de l'any 2007 del segle 21
```

tr pot ser molt útil si volem intercanviar majúscules amb minúscules:

```
$ tr 'a-z' 'A-Z' <prova.txt
```

```
HOLA MON DE L'ANY 2007 DEL SEGLE 21
```

o si es prefereix també pot fer-se de la següent forma

```
$ tr [:lower:] [:upper:] <prova.txt
```

```
HOLA MON DE L'ANY 2007 DEL SEGLE 21
```

Tot excepte...

El comandament tr té alguns paràmetres que ens permeten tenir un major control. Per exemple, podem utilitzar l'opció -d per a esborrar:

```
$ tr -d '0-9' < prova.txt
```

```
hola mon de l'any del segle
```

que provoca que tots els números passin a millor vida. Si combinem aquesta opció amb -c tenim una forma encara millor d'eliminar contingut superflu. Si volem suprimir qualsevol cosa excepte espais en blanc, lletres en majúscula i lletres en minúscula, utilitzem -c per a indicar-li a tr que és el que no ha d'esborrar:

```
$ tr -c -d 'A-Z a-z' <prova.txt
```

```
hola mon de l'any del segle
```

Diguem que l'opció -c fa que el -d es comporti al revés. Noti's que hem deixat un espai entre A-Z i a-z, sinó es menjarà també els espais i nota que també ha desaparegut la ' .

En combinació amb -s, tr ens permet reduir el tamany d'un arxiu. És útil, per exemple si tenim un arxiu .log ple d'espais en blanc. L'opció -s espera el pas d'un o dos arguments. Per exemple,

(suposem que prova2.txt conté una línia: "hola mon de l'any 2007 del segle 21" amb espais repetits)

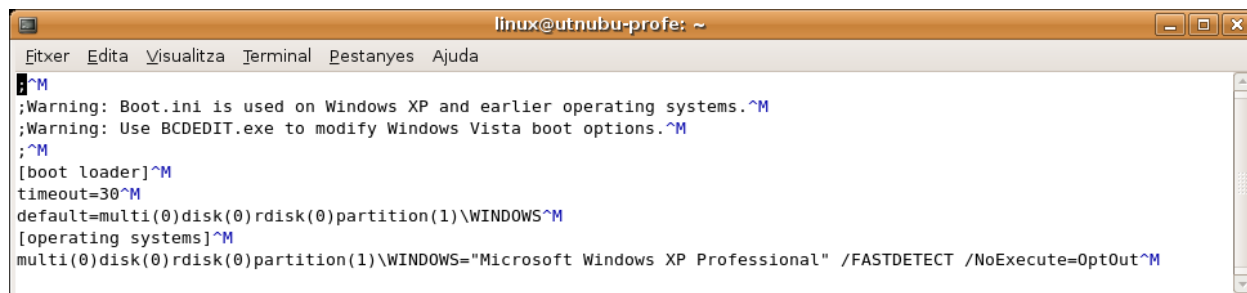
```
$ tr -s ' ' <prova2.txt
```

```
hola mon de l'any 2007 del segle 21
```

Entre dos mons

(obsolet)

Si hem tingut ocasió d'intercanviar fitxers entre sistemes Windows i Linux, haurem comprovat que solen aparèixer caràcters estranys al final de les línies. L'editor vi per exemple els mostra amb el caràcter ^M al final:



```
linux@utnubu-profe: ~  
Fitxer Edita Visualitza Terminal Pestanyes Ajuda  
^M  
;Warning: Boot.ini is used on Windows XP and earlier operating systems.^M  
;Warning: Use BCDEDIT.exe to modify Windows Vista boot options.^M  
^M  
[boot loader]^M  
timeout=30^M  
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS^M  
[operating systems]^M  
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional" /FASTDETECT /NoExecute=OptOut^M
```

La raó de tot això radica en que windows i linux utilitzen símbols diferents per al caràcter de nova línia. Windows utilitza un \r\n i linux únicament utilitza \n. Això no passa amb editors tipus gedit, ja que fan la traducció internament.

Per fer la traducció:

```
$ tr -d '\r' <fitxerwin >fitxerlinux
```

Amb això eliminarem el caràcter extra del final de cada línia.

CUT

El filtre cut es fa servir per a tallar i passar a la sortida estàndard les columnes o camps de l'entrada estàndard o de l'arxiu especificat. L'opció -c és per a tallar columnes i l'opció -f per a tallar camps. En aquest darrer cas existeix una opció -d per a indicar quin és el separador de camps, que per defecte és el tabulador.

Per a especificar els camps o columnes que volem tallar s'utilitza una llista:

A-B camps o coulmes des d'A fins a B, inclosos

A- Des d'A fins al final

A,B A i B

Exemples:

```
$ cat persones
```

```
SSP : 908732124
```

```
ASF : 456789212
```

```
MBV : 432765433
```

```
ASH : 423562563
```

```
JPA : 798452367
```

```
$ cut -c 1-3 persones
```

```
SSP
```

```
ASF
```

```
MBV
```

```
ASH
```

```
JPA
```

```
$ cut -f 2 -d: persones
```

```
908732124
```

```
456789212
```

```
432765433
```

```
423562563
```

```
798452367
```



\$ cut -f 1 -d: persones

SSP

ASF

MBV

ASH

JPA