Filtres i expressions regulars

# Índex

[Índex](#_qfwfpag7ijt0)

[Filtres generals](#_ybjv1oz5qxot)

[Expressions regulars](#_5kgdb4gjlgk1)

[Expressions regulars POSIX (grep, sed i awk)](#_dp2g9liumlix)

[Avaluació cobdiciosa (“greedy”)](#_zchgib6z46u9)

[Escapament de caràcters i expressions regulars](#_m8hw4teiz968)

[grep](#_khf0zbjy4nu6)

[Exemples](#_xsxbknrh5l4a)

[Opcions més importants de grep](#_ij4a9iz8siaw)

[Programes alternatius de grep](#_oke2el6cnqey)

[Extensions disponibles en les avaluacions fent servir grep -P](#_f7891ihp5wo0)

[Quantificadors “lazy” (mandrosos), en comptes de “greedy” (cobdiciosos)](#_73nr289153ka)

[Lookahead en l’avaluació de grep -P](#_a6skj3yw775p)

[sed (Stream Editor)](#_mqwe5e7pt0nm)

[Introducció a sed i limitacions de les expressions regulars](#_cix5l0kyh8ct)

[Opcions de sed](#_ogbd3nfokkx0)

[Sintaxi de sed](#_w5u148txayx3)

[Instruccions més freqüents](#_ap9wrqooio8s)

[Exemple ordre d](#_3wz0x4kur98)

[Exemple multiples ordres i ordre n](#_20bejmtu982b)

[Exemple ordre q](#_mo8178wrn59a)

[Primers exemples, imitant el comportament de grep amb sed -n i el modificador p](#_qikomomd8ocq)

[Extracció d’informació: grep -o](#_mhcj5gug1wwa)

[Exemple: extracció d’adreces de correu](#_l7et6jsa0gkn)

[Exemple: Extreure UID\_MIN de /etc/login.defs](#_m56omiccnvrn)

[Exemple: Seleccionar un usuari aleatòriament.](#_g018d1v39zga)

[Tractament numèric](#_gj8w9w1x790)

[Exemple: Seleccionar un usuari amb UID >= 1000](#_2c8eqqdqpqpe)

[Modificació de les dades](#_s6pgy2t32qqf)

[Exemple: comparar directoris d’usuari declarats i directoris existents en /home](#_tfitujbokf5e)

[Altres exemples senzills](#_yugt3es05rh2)

[Exemple: Cas senzill d’extreure text d’una pàgina html](#_3gv2ucmkdxnf)

[Exemple “frases còmiques” en castellà](#_4zrf5i2q8htq)

[Exemple](#_sxsn1onzuj18)

[Exemple: extreure frases celebres d’arxiu html](#_rn5ehr900ttl)

[Casos complexos amb sed](#_1luh1wluuhp0)

[Etiqueta i instrucció de bifurcació en sed](#_aic7ut9qk79q)

[Materials complementaris](#_wo36pn5vddj0)

[Altres ordres relacionades amb els filtres](#_ccaebtbahcll)

[Especials per la gestió redireccions](#_8sewlyw7lr1r)

[Visualització](#_irm4gbytqsos)

[Comparació d’arxius](#_n5jd9ygu64js)

[Altres ordres](#_vjnmn8ep23ac)

# Filtres generals

* **grep** ( egrep / fgrep ): Fonamentalment farem servir grep per ***obtenir les línies que compleixen una expressió*** regular.
  + També es pot fer servir ***amb l’opció -q com a operador lògic***, perquè no produirà sortida, però retornarà zero (true) quan existeixi alguna línia que compleixi la condició; si cap línia la compleix, retorna un codi d’error diferent de zero.
  + Cridat amb l’opció ***-F*** és equivalent a l’ordre ***fgrep***. No s’ha de fer servir ***fgrep***, que és obsolet. Permet buscar literals, en comptes de fer servir expressions literals. Combinada amb l‘opció ***-f***, es pot fer servir un fitxer amb les paraules o textos que busquem, un per línia, i, donat un, o múltiples fitxers, retornarà totes les línies on hagi trobat una de les paraules o textos que es buscaven.
  + Cridat amb l’opció ***-E*** és equivalent a l’ordre ***egrep***. No s’ha de fer servir ***egrep***, que és obsolet. Ens permet escriure les expressions regulars amb comoditat, sense haver d’escapar els caràcters que formen l’expressió regular.
    - *En realitat l’opció indica que es fan servir expressions regulars extendides, però la versió GNU sempre té les expressions regulars extendides disponibles.*
  + Amb l’opció ***-o*** obtindrem només el text que correspon al patró, cada coincidència en una línia de sortida; les línies de sortida són coincidències.
  + Amb l’opció ***-c*** compta el número d’aparicions del patró.
  + **Tutorial de grep i introducció a sed**:
    - <https://blog.desdelinux.net/con-el-terminal-uso-de-expresiones-regulares/>
    - <https://blog.desdelinux.net/con-el-terminal-uso-de-expresiones-regulares-ii-reemplazos/>
* **sed** (***S****tream* ***ED****itor*): Permet la manipulació de l’entrada fent servir expressions regulars.
  + Ho cridem amb l’opció ***-E***, que facilita l’escriptura de les expressions regulars sense fer servir escapaments.
    - *En realitat l’opció indica que es fan servir expressions regulars extendides, però la versió GNU sempre té les expressions regulars extendides disponibles.*
  + L’opció ***-n*** evita que les líneas que es processen es copiïn en ***stdout***.
  + L’opció ***-f*** *filename* permet fer servir un fitxer (*filename*) amb les instruccions de sed (***programa***), en comptes d’indicar-les en la mateixa línia de comandes; llavors no ens caldrà fer servir cometes per les expressions de sed. Opció útil quan el *programa* de sed fa servir diverses línies.
  + Per saber-ne més: Tutorial de sed en profunditat (millor que el manual):
    - <https://www.grymoire.com/Unix/Sed.html>
* **cut**: Considera les línies formades per camps, i els camps separats amb un caràcter delimitador ***-d*** (*delimiter*), que per defecte és el tabulador. Permet extreure només els camps que indiquem amb ***-f*** (*fields*).
* **sort**: De la mateixa manera que cut, permet indicar el caràcter de separació dels camps, amb ***-t***, i els camps que constitueixen la clau per l’ordenació amb ***-k*** (*key*). És important diferenciar entre ordenacions numèriques i alfabètiques; l’opció ***-n*** permet activar l’ordenació numèrica, però potser s’ha d’indicar per a algun camp de la clau, en comptes d’activar-lo per a tots els camps. En l’opció ***-k*** cal indicar tots els camps que participen en la clau per l’ordenació. L’opció ***-r*** genera una ordenació inversa, i l’opció -u elimina línies ordenades. També existeix l’opció ***-V***, que permet l’ordenació per un camp que inclou un número al final.
  + Opcions *especials*, que el fan actuar com a un programa diferent:
    - Amb l’opció ***sort -c*** només es verifica si el fitxer està ordenat.
    - Amb l’opció ***sort -m*** es fa un “*merge*” de fitxers ordenats, generant un d’únic que ajunta els arxius d’entrada.
* **uniq**: Per tractar les línies repetides:
  + Comportament per defecte: Elimina línies repetides, però les línies han d’estar ordenades, i per tant, les línies iguals, consecutives. L’opció ***sort -u*** també elimina les línies repetides.
  + Amb l’opció ***uniq -c*** afegeix el nombre de repeticions de cada línia.
  + Amb l’opció ***uniq -d*** només mostra les línies repetides.
  + Amb l’opció ***uniq -u*** només mostra les línies úniques.
* **wc**: *word count*; compta paraules, línies i caràcters.
  + Amb l’opció ***wc -l*** compta les línies.
  + Amb l’opció ***wc -m*** compta els caràcters.
  + Amb l’opció ***wc -w*** compta les paraules.
  + Amb l’opció ***wc -c*** compta els bytes; compte! només en arxius ***ASCII*** serà igual a ***wc -m***; avuí dia els fitxers de text tenen caràcters unicode.
* **tr**: substitueix caràcters. S’indica el conjunt de caràcters a canviar i el conjunt de nous valors; tots dos conjunts han de tenir la mateixa mida.
  + Amb l’opció ***tr -d*** només s’indica un conjunt: caràcters que s’eliminen.
  + Amb l’opció ***tr -s*** només s’indica un conjunt: caràcters pels que s’eliminen les repeticions consecutives.
  + Fent servir totes dues opcions ***tr -ds*** s’indiquen dos conjunts: el primer conjunt són caràcters que s’eliminen, i el segón són caràcters pels que s’eliminen les repeticions consecutives.
* **expand / unexpand**: convertir tabuladors en espais, i a l’inrevés.

## Exemples de tractament amb filtres

Es mostra només la indicació de com fer-los, però l’execució queda com a exercici:

* Obtenir les vint paraules més freqüents en un text:
  + Dividim les paraules, una per línia, amb **grep**.
  + Ordenem el resultat amb **sort**.
  + Comptem les repeticions amb **uniq**.
  + Ordenem una altra vegada, per obtenir una ordenació per nombre d’aparicions de la paraula, fent servir **sort**.
  + Retornem només les vint primeres files, amb **head**.
* Obtenir els usuaris del sistema que tenen bash com a shell:
  + Obtenir els usuaris amb **getent**.
  + Obtenir amb **grep** els usuaris que tenen bash.
  + Obtenir amb **cut** el nom de l’usuari.
* Obtenir N línies des de la línia M:
  + Obtenim les N+M primeres línies amb **head**.
  + Obtenim les N últimes línies amb **tail**.
* Obtenir els últims 3 fitxers modificats del directori:
  + Obtenim els fitxers del directori ordenats per data de modificació.
  + Agafem els 3 primers amb **head**.
* Obtenir les NIC amb adreces IP començant per 192:
  + Fem servir **ip address** per obtenir les adreces IP.
  + Filtrem amb **grep** les línies que tenen el valor 192.
* Obtenir les NIC que són actives:
  + Fem servir **ip address** per obtenir les interfícies.
  + Fem servir **grep** per trobar les línies amb la paraula UP.
* Saber per a quins directoris té root permís d’escriptura:
  + Llistar els directoris amb **sudo ls -l**.
  + Obtenir quins directoris té permís d’escriptura root fent servir **grep** per buscar ‘drwx’.
* Saber el nombre de fitxers d’un directori:
  + Obtenim el llistat de fitxers fent servir **ls -1**.
  + Comptem les línies amb **wc -l**.
* Obtenir els PIDs (proces identifier) de tots els processos relacionats amb systemd:
  + Amb **ps -ef** obtenim la llista de processos.
  + Amb **grep** deixem només les línies on apareix systemd.
  + Agafem només el PID fent servir **awk '{ print $2 }'**.
  + En realitat també surt el nostre procés de búsqueda, però amb **grep -v grep** podem excloure les línies on aparegui grep.

## Ús de l’ordre sort amb múltiples camps i valors numèrics

L’ordre sort, quan volem indicar diverses claus d’ordenació, requereix indicar correctament cada clau; cal **indicar el final de la clau**, perquè en cas contrari, sort considera tots els camps que segueixen el camp indicat, política correcte quan només volem ordenar per un camp, però problemàtica si volem una segona clau; per l'última clau generalment no serà necessari indicar el final, però convenient si és una clau numèrica.

També és molt important tenir present la diferència entre l’**ordenació alfabètica** i l’**ordenació numèrica**; el valor 10 és anterior a 2 i segueix a 1 alfabèticament (pensa que AB és posterior a A, però anterior a B), però tots sabem que numèricament l’ordenació correcta és el 2 darrera de 1, i el 10 després del 9.

Existeixen dues ordenacions numèriques en l’ordre sort:

* ***-g*** ofereix una ordenació numèrica general, que inclou números de punt flotant en notació científica
* ***-n***, només permet números, signe negatiu si escau, i possibles decimals, però no exponent ni signe positiu.

Es recomana (en el manual) fer servir l’opció ***-n*** sempre que sigui possible, tant per ser molt més ràpida, com pel fet que la conversió a punt flotant podria incorporar pèrdues d'informació, tot i que no detalla exàctament la problemàtica associada (potser per casos d’overflow, underflow i errors de conversió?):

<http://www.gnu.org/software/coreutils/manual/html_node/sort-invocation.html>

### Exemple de l’ordre sort amb un camp no numèric i un altre numèric.

Ordenar l’arxiu nameslist.txt:

| [cesc@Skye ~]$ cat nameslist.txt Emily 0 Emily 1 Emily 5 Joe 0 Joe 10 Joe 5 Joe 6 |
| --- |

El problema és que si l’ordenació que fem és alfabètica, el resultat que obtenim no és el que voldríem:

| [cesc@Skye ~]$ sort -t' ' -k1,1 -k2,2 nameslist.txt Emily 0 Emily 1 Emily 5 Joe 0 Joe 10 Joe 5 Joe 6 |
| --- |

Però podem indicar els camps numèrics per obtenir una ordenació correcta; no oblidem, a més, que cal indicar des de quin camp fins a quin camp formen la clau, per a cada clau que indiquem:

| [cesc@Skye ~]$ sort -t' ' -k1,1 -k2,2n nameslist.txt Emily 0 Emily 1 Emily 5 Joe 0 Joe 5 Joe 6 Joe 10 |
| --- |

# Expressions regulars

Una **expressió regular** (col·loquialment anomenades *regexp*, acrònim de l'anglès *regular expression*) és una representació d'una porció de text genèric a buscar dins d'un altre text, com per exemple uns caràcters, paraules o patrons de text concrets.

Les expressions regulars s’escriuen fent servir unes regles, que formen un llenguatge formal. Les expressions regulars són un component important del shell de Linux, molt orientat al processament d’arxius de text, però les expressions regulars són també una important eina de programació en qualsevol llenguatge de programació, per la seva capacitat per trobar patrons dins dels texts; la sintaxi de les expressions pot canviar quan l’escrivim en un o altre programa, però en general les característiques, i fins i tot la sintaxi, són molt semblants, però sempre haurem de consultar el manual si escrivim en un altre llenguatge.

## Expressions regulars POSIX (grep, sed i awk)

*En gris son abreviacions:*

| **.** | Un caràcter qualsevol  *Generalment treballem per línies, i el final de línea no el reconeix amb el punt.* |
| --- | --- |
| **[ ]** | Un caràcter d’entre els indicats dins dels claudàtors.  *els caràcters que escrivim entre els claudàtors -> sera un d’aquests. Podem escriure rangs. No tindrem ni la ç ni ñ (caràcters opcionals)*  grep -E "[A-Z]" ficher.txt |
| **[^ ]** | Un caràcter que no és un d’entre els indicats dins dels claudàtors.  *Seria el contrari de l’anterior* |
| **^** | Àncora a començament del text.  *Principi de la linea* |
| **$** | Àncora acabament del text.  *Final de la linea* |
| **( )** | Subexpressio.  *Per utilitzar quantificadors.*   * Les subexpressions permeten que les hi apliquem quantificadors de repetició. * Les subexpressions permeten agrupar una part del text que ha coincidit, i després referenciar-la.   veure: [Back-references and subexpressions](https://www.gnu.org/software/sed/manual/html_node/Back_002dreferences-and-Subexpressions.html) |
| **+** | Quantificador: L’anterior element es pot repetir.  *Al menys 1 o n* |
| **?** | Quantificador: L’anterior element és opcional, pot no ser-hi.  *pot estar o no estar. Ex: (a)? (hola)? → ho entenc com una expressió, un caràcter darrere d’altre.* |
| **\*** | Quantificador: L’anterior element pot no ser-hi o repetir-se. *Està 0 o n* |
| **{***m****,*** *n***}** | Quantificador: L’element anterior es repeteix: mínim ***m*** vegades, màxim ***n*** vegades.  *Si volem que es repeteixen un número exacte. {mínim,màxim}* |
| **{**m**}** | Quantificador: L’element anterior es repeteix ***m*** vegades. |
| **{**m**, }** | Quantificador: L’element anterior es repeteix com a mínim ***m*** vegades. |
| **{,** n**}** | Quantificador: L’element anterior es repeteix com a màxim ***n*** vegades. |
| **|** | Dues subexpressions alternatives, una a l’esquerra i una altra a la dreta. Només es requereix que es compleixi una.  *Dues accions, o aquesta expressió o aquesta altra. (OR)* |
| **\<** | Indicador començament de paraula. Compte amb la definició de “paraula” (veure **\w**)  *Començament de paraula* |
| **\>** | Indicador final de paraula. Compte amb la definició de “paraula” (veure **\w**)  *Final de paraula* |
| **\b** | Indicador final o començament de paraula. Preferible fer servir explícitament els indicadors de començament o final de paraula, segons el cas. |
| **\B** | Indicador que no es troba a final o començament de paraula. |
| **[:***classe***:]** | un caràcter que pertany a *classe*, on *classe* és una de les categories definide per POSIX. Es pot fer servir a qualsevol lloc on es pot fer servir un caràcter en l’expressió regular. S’admeten les categories POSIX:  *Es a dir, si volem algú d’això s’haura d’escriure: [[:\_\_\_:]]*   | **[:alnum:]** | alphanumeric. Inclou [:alpha:] i [:digit:] | | --- | --- | | **[:alpha:]** | alphabetic characters. | | **[:lower:]** | caràcters alfabètics en minúscula | | **[:upper:]** | caràcters alfabètics en majúscula. | | **[:blank:]** | space or tab.  *Aquest es l’habitual d’utilitzar* | | **[:cntrl:]** | control characters. Codi ASCII des de zero fins al 037 (que en decimal és 31), i també inclou el 177 (en decimal 127) | | **[:digit:]** | digits decimals, des de zero fins a 9.  *Que seria lo mateix [0-9] que és més ràpid* | | **[:xdigit:]** | digits hexadecimals, incloent-hi [A-F] | | **[:graph:]** | caràcters representables; inclou [:alnum:] i [:punct:] | | **[:print:]** | caràcters imprimibles; inclou [:graph:] i l’espai. | | **[:punct:]** | caràcters de puntuació;  compte, perquè també inclou caràcters tals com ‘@’, ‘$’ i ‘#’ | | **[:space:]** | caràcters separadors: tab, newline, vertical tab, form feed, carriage return, and space.  *Això inclou final de línea, però també altres caràcters com separadors.* | |
| **\w** | Sinònim de **[\_[:alnum:]]**. Defineix caràcters que formen “paraules”; però compte, perquè les “paraules” no es refereixen al nostre concepte de paraules, sinó que poden incloure dígits i el caràcter de subratllat.  *O el número O el subratllat.* |
| **\W** | Sinònim de **[^\_[:alnum:]]**. Caràcters que no pertanyen al conjunt de caràcters **\w**. |
| **\s** | Sinònim de **[[:space:]]**. És a dir: qualsevol caràcter separador: espai, tabulador, final de línia... |
| **\S** | Sinònim de **[^[:space:]]**. |

| grep -E '[[:upper:][[:lower:]]\*' ficher.txt |
| --- |

*Això marca les paraules que comencen amb Majúscula, i-o de minuscula.*

| grep -E 'l{2}' fitxer.txt |
| --- |

*Això marcará les paraules que tenen dos ll seguides.*

| *grep -E ' {2, } fitxer.txt* |
| --- |

*Això marcará si he escrit dos espais o més*

| *grep -E '\.{3}' fitxer.txt* |
| --- |

*Si volem agafar tres punts suspensius. haurem de ficar la contrabarra al punt.*

| *grep -E '\<w[[:alpha:]]\*'* |
| --- |

*Agafa la línia que tingui paraules que comencen amb la lletra w, seguit de lletres. Recordar que es case senstivie, si volem majúscules o minuscules*

| *grep -E '\<[wW}[[:alpha:]]\*'* |
| --- |

### Avaluació cobdiciosa (“greedy”)

L’avaluació dels quantificadors (\*,+ i ?) fa servir un algorisme “cobdiciós” (greedy). Això significa que retornarà el text més llarg possible que compleixi l’expressió. Cal anar amb compte, perquè sovint no hi pensem, i creiem que el model coincidirà amb la primera ocurrència.

**Exemple: Trobar el text entre cometes:**

| [cesc@Skye ~]$ grep -Eo '".+"' <<< 'A "witch" and her "broom" is one.' "witch" and her "broom" |
| --- |

*‘“.+”’ Algun caràcter dins de cometes dobles i acaba amb cometes dobles.*

*Agafa el més llarg:* "witch" and her "broom"

*No ha fet el que nosaltres volíem.*

El correcte hauria estat:

| [cesc@Skye ~]$ grep -Eo '"[^"]+"' <<< 'A "witch" and her "broom" is one.' "witch" "broom" |
| --- |

*Ho hauriem de fer: Cometes dobles, seguit de un caràcter que no sigui la cometa doble. Com ha mínim una vegada i la cometa doble trencant.*

*Llavors aquí sí que retorna:* "witch" "broom"

Exemple: Obtenir tags HTML:

| [cesc@Skye ~]$ grep -Eo '<.\*>' <<< '<title>My webpage title</title>' <title>My webpage title</title> |
| --- |

*Passa el mateix: agafa el més llarg:* <title>My webpage title</title>

El correcte hauria estat:

| [cesc@Skye ~]$ grep -Eo '<[^>]\*>' <<< '<title>My webpage title</title>' <title> </title> |
| --- |

*Hauria de ser de la mateixa manera, dir que sempre que no trobi un >*

### Escapament de caràcters i expressions regulars

Les expressions regulars en GNU són idèntiques, tant si es fan servir les expressions bàsiques com la variant extesa; és per això que es recomana fer servir l’opció **-E**, que només té influència en la manera d’escriure les expressions regulars.

Les expressions regulars es poden fer servir tant en ***grep*** com en ***sed***, indicant l’opció **-E**. Quan no s’indica l’opció **-E** alguns caràcters s’han d’escapar per indicar que tenen significat especial; aquest comportament és per motius de compatibilitat enrere, i es recomana fer servir l’opció ***-E***, excepte quan no és necessari cap dels caràcters que requereixen escapament (expressions molt senzilles):

| **caràcters que requereixen escapament si no es fa servir l’opció *-E*** |
| --- |
| **+** |
| **?** |
| **(** |
| **)** |
| **{** |
| **}** |
| **|** |

## grep: TROBAR LINIES

La sintaxi és:

| grep -E 'expressió regular' fitxer |
| --- |

Exemple:

| [cesc@Skye pràctiques]$ # Troba línies de /etc/passwd que comencen per c [cesc@Skye pràctiques]$ grep -E '^c' /etc/passwd cups:x:209:209:cups helper user:/:/usr/bin/nologin colord:x:973:973:Color management daemon:/var/lib/colord:/usr/bin/nologin cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash convidat:x:1001:1001::/home/convidat:/bin/bash |
| --- |

Exemple:

Troba **línies** de /etc/passwd que **acaben** per bash

| [cesc@Skye pràctiques]$ grep -E 'bash$' /etc/passwd root:x:0:0::/root:/bin/bash cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash convidat:x:1001:1001::/home/convidat:/bin/bash |
| --- |

Exemple:

Troba **línies** de /etc/passwd que continguin la seqüència ‘:1’

| [cesc@Skye pràctiques]$ grep -E ':1' /etc/passwd bin:x:1:1::/:/usr/bin/nologin mail:x:8:12::/var/spool/mail:/usr/bin/nologin ftp:x:14:11::/srv/ftp:/usr/bin/nologin polkitd:x:102:102:PolicyKit daemon:/:/usr/bin/nologin rtkit:x:133:133:RealtimeKit:/proc:/usr/bin/nologin usbmux:x:140:140:usbmux user:/:/usr/bin/nologin cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash convidat:x:1001:1001::/home/convidat:/bin/bash nvidia-persistenced:x:143:143:NVIDIA Persistence Daemon:/:/usr/bin/nologin |
| --- |

#### Exemples

Fem servir un fitxer per fer les proves; li diem proves.txt i el copiem aquí en fons verd:

- Lista de páginas wiki:

ArchLinux: https://wiki.archlinux.org/

Gentoo: https://wiki.gentoo.org/wiki/Main\_Page

CentOS: http://wiki.centos.org/

Debian: https://wiki.debian.org/

Ubuntu: https://wiki.ubuntu.com/

- Fechas de lanzamiento:

Arch Linux: 11-03-2002

Gentoo: 31/03/2002

CentOs: 14-05-2004 03:32:38

Debian: 16/08/1993

Ubuntu: 20/10/2004

Desde Linux Rulez.

Troba **línies** amb la paraula Linux

| [cesc@Skye pràctiques]$ grep -E 'Linux' proves.txt ArchLinux: https://wiki.archlinux.org/ Arch Linux: 11-03-2002 Desde Linux Rulez. |
| --- |

Troba **línies** amb el text CentO seguit d’un caràcter qualsevol

| [cesc@Skye pràctiques]$ grep -E 'CentO.' proves.txt CentOS: http://wiki.centos.org/ CentOs: 14-05-2004 03:32:38 |
| --- |

Troba **línies** amb els valors 2002 o 2004

| [cesc@Skye pràctiques]$ grep -E '200[24]' proves.txt Arch Linux: 11-03-2002 Gentoo: 31/03/2002 CentOs: 14-05-2004 03:32:38 Ubuntu: 20/10/2004 |
| --- |

Troba **línies** amb el caràcter de dos-punts, seguit d’un caràcter que no sigui la barra

| [cesc@Skye pràctiques]$ grep -E ':[^/]' proves.txt ArchLinux: https://wiki.archlinux.org/ Gentoo: https://wiki.gentoo.org/wiki/Main\_Page CentOS: http://wiki.centos.org/ Debian: https://wiki.debian.org/ Ubuntu: https://wiki.ubuntu.com/ Arch Linux: 11-03-2002 Gentoo: 31/03/2002 CentOs: 14-05-2004 03:32:38 Debian: 16/08/1993 Ubuntu: 20/10/2004 |
| --- |

Troba **línies** amb un dígit seguit d’un guió

| [cesc@Skye pràctiques]$ grep -E '[0-9]-' proves.txt Arch Linux: 11-03-2002 CentOs: 14-05-2004 03:32:38 |
| --- |

Troba **línies** amb el text http o https seguit de dos-punts. *El interrogant és només per la ‘s’*

| [cesc@Skye pràctiques]$ grep -E 'https?:' proves.txt ArchLinux: https://wiki.archlinux.org/ Gentoo: https://wiki.gentoo.org/wiki/Main\_Page CentOS: http://wiki.centos.org/ Debian: https://wiki.debian.org/ Ubuntu: https://wiki.ubuntu.com/ |
| --- |

Troba **línies** que continguin text que conté el text wiki, i després possiblement amb caràcters entremig, apareix el text org  *Wiki (alguna cosa) org*

| [cesc@Skye pràctiques]$ grep -E 'wiki.\*org' proves.txt ArchLinux: https://wiki.archlinux.org/ Gentoo: https://wiki.gentoo.org/wiki/Main\_Page CentOS: http://wiki.centos.org/ Debian: https://wiki.debian.org/ |
| --- |

Troba **línies** amb 5 o més lletres minúscules al final de la paraula

| [cesc@Skye pràctiques]$ grep -E '[[:lower:]]{5,}\>' proves.txt - Lista de páginas wiki: ArchLinux: https://wiki.archlinux.org/ Gentoo: https://wiki.gentoo.org/wiki/Main\_Page CentOS: http://wiki.centos.org/ Debian: https://wiki.debian.org/ Ubuntu: https://wiki.ubuntu.com/ - Fechas de lanzamiento: Gentoo: 31/03/2002 Debian: 16/08/1993 Ubuntu: 20/10/2004 |
| --- |

Troba **línies** amb paraules que comencen amb majúscules seguides de 5 o més lletres minúscules.

| [cesc@Skye pràctiques]$ grep -E '\<[[:upper:]][[:lower:]]{5,}' proves.txt Gentoo: https://wiki.gentoo.org/wiki/Main\_Page Debian: https://wiki.debian.org/ Ubuntu: https://wiki.ubuntu.com/ - Fechas de lanzamiento: Gentoo: 31/03/2002 Debian: 16/08/1993 Ubuntu: 20/10/2004 |
| --- |

#### Opcions més importants de grep

Generalment farem servir l’**opció -E que facilita l’escriptura d’expressions regulars**, tot i que l’opció -F pot ser pràctica si no busquem una expressió, sino un text concret. *Sinó hauriem d’utilitzar contrabarra per escapar cosas com /\ .. d’aquesta manera possem contrabarra només quan vulguem utilitzar les classes.*

L’opció -P permet certes extensions. Cal anar amb compte, perquè no sempre faciliten les expressions, sinó que, de vegades, si no tenim experiència, podem triar opcions poc adequades i complexes, i descuidar-nos de les opcions més evidents (i eficients).

| **Format de les expressions regulars** | |
| --- | --- |
| **-E** | Expressions regulars (sense haver d’escapar-les) |
| **-F** | Strings, en comptes d’expressions regulars. |
| **-P** | Expressions regulars encara amb més quantificadors i expressions; fa servir el motor d’expressions regulars de ***Perl***. Preferible no fer-lo servir. |
| **Opcions de sortida** | |
| **-q** | No genera sortida, només codi d’error zero si troba línies, diferent de zero si no les troba. |
| **-c** | Retorna només el nombre de resultats (generalment línies). |
| **-o** | Només retorna els resultats que s’han trobat, escrivint un per línia. |

* L’opció de sortida ***-q*** és de molta utilitat, per incorporar condicions en els scripts (generalment en una ordre if, però també podria donar-se el cas en un bucle).
* L’opció de sortida ***-c*** també és força usual, per comptar les aparicions d’un text o d’un patró.  *Compadors. no retorna quines, sino quantes.*

| [cesc@Skye pràctiques]$ grep -c 'Marx' frases.txt 3 |
| --- |

… efectivament, en el fitxer frases.txt n’hi ha tres frases de Marx, tot i que dues són de Groucho Marx, i una de Karl Marx.

* L’opció ***-o*** permet obtenir com a resultats exclusivament el text que produeix una coincidència amb el patró que es buscava; fin i tot, quan es produeixen dues coincidències en una mateixa línia, es produeixen dues línies de sortida, una amb cadascuna de les coincidències. Aquest comportament ens acosta una mica a una altra eina: ***sed***.

### Programes alternatius de grep

El programa ***egrep*** és equivalent a ***grep -E***, i ***fgrep*** a ***grep -F***, i es mantenen per compatibilitat enrere; es recomana **no fer servir egrep ni fgrep**.

### Extensions disponibles en les avaluacions fent servir grep -P

#### Quantificadors “lazy” (mandrosos), en comptes de “greedy” (cobdiciosos)

Els quantificadors ***\**** i ***+***, i fins i tot ***?***, tenen un comportament cobdiciós o voraç, i intenten satisfer el màxim nombre de repeticions possibles. L’enfocament contrari és l’enfocament mandrós, que intenta satisfer el mínim nombre de repeticions. Es pot canviar el comportament greedy per un de lazy incorporant un interrogant al final del quantificador. Així tenim els quantificadors lazy ***\*?***, ***+?*** i ***??***.

Els quantificadors *lazy* només són disponibles amb l’opció d’avaluació d’expressions regulars ***-P*** (*Perl*), que incorpora més extensions.

| **Quantificadors d’avaluació lazy** | |
| --- | --- |
| \*? | Quantificador \* en mode *lazy*. |
| +? | Quantificador + en mode *lazy*. |
| ?? | Quantificador ? en mode *lazy*. |
| {m, n}? | Quantificador {n, m} en mode *lazy*. |
| {m, }? | Quantificador {m, }? en mode *lazy*. |
| { , n}? | Quantificador { , n}? en mode *lazy*. |

Els quantificadors són “greedy”:

[cesc@Skye ~]$ grep -E 'c.\*s' <<< 'can cats eat plants'

**can cats eat plants**

Els quantificadors “lazy” no es reconeixen normalment:

[cesc@Skye ~]$ grep -E 'c.\*?s' <<< 'can cats eat plants'

**can cats eat plants**

Cal fer servir l’opció ***grep -P***:

[cesc@Skye ~]$ grep -P 'c.\*?s' <<< 'can cats eat plants'

**can cats** eat plants

Sense els quantificadors “lazy” el resultat és el mateix que amb ***grep -E***:

[cesc@Skye ~]$ grep -P 'c.\*s' <<< 'can cats eat plants'

**can cats eat plants**

* Trobar les paraules entre cometes:

[cesc@Skye ~]$ grep -Po '".+?"' <<< 'A "witch" and her "broom" is one.'

"witch"

"broom"

* Obtenir tags HTML:

[cesc@Skye ~]$ grep -Po '<.\*?>' <<< '<title>My webpage title</title>'

<title>

</title>

Només amb l’opció ***-P*** (extensió per expressions regulars de Perl) podem fer servir els quantificadors *lazy*. És convenient fer servir les expressions regulars “greedy”, perquè són més compatibles i podem fer-les servir en altres programes que no admeten les extensions “lazy”, com ara ***sed***.

#### Lookahead en l’avaluació de grep -P

També l’opció ***-P*** incorpora altres millores, com el “*lookahead*”, una expressió que no és part del resultat.

[cesc@cport-virtualbox b]$ grep -P '.\*(?=ando)' << end

> buscando a Nemo

> buscando a Dory

> buscando a Wally

> buscando a Susan desesperadamente

> educar jugando

> crecer jugando

> chupando limones

> pilotando un F1

> fumando espero...

> end

**busc**ando a Nemo

**busc**ando a Dory

**busc**ando a Wally

**busc**ando a Susan desesperadamente

**educar jug**ando

**crecer** **jug**ando

**chup**ando limones

**pilot**ando un F1

**fum**ando espero...

### grep -F per buscar strings, en comptes de fer servir expressions regulars

Amb l’opció ***grep -F*** ja no fem servir expressions regulars, sinó senzillament textos. Aquesta opció sovint es combina amb ***-f*** per indicar un fitxer amb una llista de textos a buscar, un per línia. Si, a més a més, els textos que busquem són paraules, podem activar també l’opciò ***-w***. Així, per buscar una llista de paraules, fent servir també l’opció ***-i*** per no diferenciar les majúscules, faríem servir ***grep -wiFf paraules.txt fitxer***.

#### Negació amb grep: l’opció -v

L’opció ***grep -v*** ens permet obtenir les línies que no compleixen el patró; aquesta opció està disponible per a totes les diferents variants ***grep -E***, ***grep -P*** i ***grep -F***.

Així, seguint amb ***grep -F***, potser d’una llista de paraules volem excloure les que ja hi són a una altra llista. Per exemple, si obtenim totes les paraules del text frases.txt, potser volem excloure els articles, que són paraules molt freqüents i no rellevants; en aquest cas farem servir ***grep -wiFvf articles.txt frases.txt***.

[cesc@Skye pràctiques]$ cat articles.txt

el

els

l

la

les

lo

los

en

n

na

un

una

unes

uns

[cesc@Skye pràctiques]$ grep -Eo '\<[[:alpha:]]+\>' frases.txt | grep -wiFvf articles.txt | wc -l

21089

## sed (Stream Editor)

#### Introducció a sed i limitacions de les expressions regulars

El programa sed permet la manipulació d’un fitxer, amb comandes que poden semblar les ordres d’un editor.

Les expressions regulars són molt potents, però també tenen les seves limitacions. Cal ser conscient quines coses poden fer-se i quines no.

Un exemple clar, que no es podria fer fent servir expressions regulars, seria comprovar que una seqüència de parèntesi tanqui correctament… es requereix comptar quants parèntesi obren i quants tanquen, i això no pot fer-ho una expressió regular (els programes que reconeixen les expressions regulars es denominen ***autòmats***, però les expressions més complexes ja s’anomenen gramàtiques).

En realitat ***sed*** és un llenguatge molt potent, i podem programar qualsevol cosa (*Turing complet*, és a dir, amb la potència d’un llenguatge de programació), però les característiques avançades de sed no són senzilles de programar. En canvi, per alguns problemes determinats, resulta una eina senzilla i flexible.

Només veurem algunes funcionalitats bàsiques de ***sed***, i sempre s’han de tenir presents les limitacions de les expressions regulars, perquè els problemes que són complexos de resoldre fent servir ***sed*** probablement és preferible resoldre’ls amb una altra eina, quedant sed com a una eina àgil i senzilla.

#### Opcions de sed

* **-E**: El mateix que en ***grep***, fer servir expressions regulars (sinó cal fer servir escapaments per molts caràcters de control).
* **-n**: No mostrar por ***stdout*** les líneas que es processen. Usual quan només es volen les línies que es copien explícitament amb ***p***.
* **-f** *filename*: Fer servir un fitxer (*filename*) amb les instruccions de sed (***programa***), en comptes d’indicar-les en la mateixa línia de comandes; llavors no ens caldrà fer servir cometes per les expressions de sed. En tant les expressions de sed es compliquen, l’ordre sed -f fitxer cobra sentit i es fa molt útil.
* **-s**: Considerar els diferents fitxers d’entrada per separat, en comptes d’adjuntar les entrades d’una única entrada.

#### Sintaxi de sed

La sintaxi és:

***# sed [-ns] '[direcció] instrucció arguments'***

* **[direcció]** es opcional, i indica l’àmbit d’actuació de les instruccions. Si no s’indica [*direcció*], s’actua en todas les líneas del flux.La direcció pot ser:
  + Un número de línea (N)
  + Un rang de números de línea (N,M)
  + Una búsqueda de regexp: ***/regexp/***
  + Un rang definit amb un número de línia i una expressió de búsqueda per indicar el final del rang.
    - És freqüent per a l’inici d’un fitxer, indicant el ***zero*** per a que totes les línies siguin considerades dins del rang, fins que es compleix la condició de búsqueda.
    - També és freqüent fer servir l’expressió de búsqueda ***$*** per a indicar el final del fitxer.
  + Un rang definit amb dues regexp, una d’inici i una altra de final separades amb una comma: ***/regexp\_ini/,/regexp\_fin/***
* La **instrucció** pot ser:

**d** = Esborrar línia actual.

**p** = Imprimir línia actual en stdout.

**s** = Substituir cadena en la línia actual.

**i** = Insertar línia abans de la línia actual.

**a** = Insertar línia després de la línia actual.

**c** = Canviar la línia actual.

**r** *fitxer* = Afegir contingut de "*fitxer*" a la línia actual.

**w** *fitxer* = Escriure sortida en “*fitxer*”.

**!** = Aplicar la instrucció a les línias no seleccionades per la condició.

**q** = Finalitzar procés del fitxer.

* ***{*** *instruccions* ***}*** = fent servir les claus es poden indicar diverses instruccions, una en cada línia, o bé fent servir punt i coma per separar-les.

#### Instruccions més freqüents

| **s**/*pattern*/*valor*/**g**  **s**/*pattern*/*valor*/**gp** | L’ordre ***s*** reemplaça *pattern* per *valor*.   * El modificador **g** fa que ho faci en totes les aparicions; si no es fa servir només ho farà en la primera aparició que trobi. * El modificador **p** copia la línia en la sortida, i generalment es fa servir en combinació amb l’opció ***-n*** de ***sed***. |
| --- | --- |
| **p** | Escriu la línia en la sortida. |
| **q** | Acaba l’execució, sense llegir més línies. |
| **n** | Salta la línia |
| **d** | Esborra la línia |

##### Exemple ordre d

L’ordre d elimina la línia

[cesc@Skye pràctiques]$ sed ‘/^[[:blank:]]\*$/d’ fitxer

Eliminació de línies en blanc. S’eliminen les línies de *fitxer* que compleixen el patró de línia buida o només amb espais en blanc: ^[[:blank:]]\*$

##### Exemple multiples ordres i ordre n

L’ordre n salta la línia.

[cesc@Skye pràctiques]$ sed ‘n;n;s/./x/g’ fitxer

Salta dues línies i en la tercera fa la substitució (ordre ‘**s**’) de tots els caràcters pel caràcter *x*; el modificador **g** indica que ho fa per a totes les aparicions.

Anirà repetint aquesta seqüència “tatxant” amb x una de cada tres línies del fitxer, fins el final.

##### Exemple ordre q

L’ordre q acaba l’execució.

[cesc@Skye pràctiques]$ sed ‘3q’ fitxer

Acaba en la tercera línia, sense llegir més línies del fitxer. En aquest cas mostraria les 3 primeres línies, perquè les línies d’entrada es copien en la sortida si no s’indica l’opció ***sed -n***

#### Primers exemples, imitant el comportament de grep amb sed -n i el modificador p

El programa ***sed*** és més potent que ***grep***, perquè fa més coses. Podem “*simular*” el comportament de ***grep***; per fer-ho, farem servir l’opció ***sed -n*** per evitar copiar l’entrada en la sortida, i amb la comanda ‘**p**’ (print) copiarem explícitament les línies que compleixen la condició.

En general ***sed*** pot imitar els comportaments de ***grep***, però ***sed*** està pensat per a la modificació de les dades d’entrada, no per la búsqueda de patrons, així que no es pot fer servir com a condició lògica (recordem que el codi de retorn zero en grep indica que ha trobat un patró, i el valor 1 quan no el troba), perquè els valors de retorn de ***sed*** no indiquen èxit, sinó veritables errors que s’han produït durant el processament.

En el cas de ***grep***, només les línies que compleixen el patró es copien en la sortida; en canvi, en el cas de ***sed***, es copien per defecte totes les línies en la sortida, línies que poden ser modificades, o evitar explícitament que siguin copiades en la sortida.

Fent servir l’opció ***sed* -n** no es copiarà l’entrada a la sortida, i només s’escriuran les línies que explícitament copiem en la sortida amb l’ordre ***p*** de ***sed***.

Exemple 1

[cesc@Skye pràctiques]$ # Línies de /etc/passwd que comencen per c

[cesc@Skye pràctiques]$ sed -En '/^c/p' /etc/passwd

cups:x:209:209:cups helper user:/:/usr/bin/nologin

colord:x:973:973:Color management daemon:/var/lib/colord:/usr/bin/nologin

cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash

convidat:x:1001:1001::/home/convidat:/bin/bash

Exemple 2

[cesc@Skye pràctiques]$ # Línies de /etc/passwd que acaben per bash

[cesc@Skye pràctiques]$ sed -En '/bash$/p' /etc/passwd

root:x:0:0::/root:/bin/bash

cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash

convidat:x:1001:1001::/home/convidat:/bin/bash

Exemple 3

[cesc@Skye pràctiques]$ # Línies de /etc/passwd que continguin la seqüència ‘:1’

[cesc@Skye pràctiques]$ sed -nE '/:1/p' /etc/passwd

bin:x:1:1::/:/usr/bin/nologin

mail:x:8:12::/var/spool/mail:/usr/bin/nologin

ftp:x:14:11::/srv/ftp:/usr/bin/nologin

polkitd:x:102:102:PolicyKit daemon:/:/usr/bin/nologin

rtkit:x:133:133:RealtimeKit:/proc:/usr/bin/nologin

usbmux:x:140:140:usbmux user:/:/usr/bin/nologin

cesc:x:1000:1000:Cesc Sasal:/home/cesc:/bin/bash

convidat:x:1001:1001::/home/convidat:/bin/bash

nvidia-persistenced:x:143:143:NVIDIA Persistence Daemon:/:/usr/bin/nologin

#### Extracció d’informació: grep -o

Quan només volem seleccionar línies ***grep*** és en realitat una eina més adequada, però quan no volem tota la línia:

* Quan les dades a extreure siguin camps que ja siguin ben diferenciats en el fitxer amb les dades, podem fer servir ***grep*** combinat amb ***cut***.
* Podem fer servir **grep -o** (*only-matching*), que retorna només el text que concorda amb l’expressió regular indicada.

Els exemples mostrats a continuació es poden fer fent servir una d’aquestes dues tècniques, sovint de manera més senzilla.

**Exemple**: Mostrar els noms de tots els usuaris de la màquina.

[cesc@Skye pràctiques]$ sed -E 's/(^[^:]\*).\*/\1/' /etc/passwd

El subgrup \1 es refereix a la primera subexpressió entre parèntesi, ^[^:]\*, que és la que reconeix el primer camp, el nom de l’usuari (a continuació del començament de línia, tots els caràcters que no són dos-punts).

**Exemple**: Mostrar els noms de tots els usuaris de la màquina i el seu UID, que no tenen shell. En aquest cas fem servir sed, en comptes de grep i un pipe amb cut, que potser resultaria més natural:

[cesc@Skye pràctiques]$ \

sed -E 's/(^[^:]\*):[^:]\*:([^:]\*).\*(\/false|\/nologin)$/\1:\2/pg' /etc/passwd

El subgrup \1 és la primera subexpressió entre parèntesi, i el subgrup \2 la segona, que recull el tercer camp del fitxer. Busquem que, després de qualsevol caràcter a continuació dels tres primers camps que hem diferenciat, la línia acabi en /false o bé en /nologin: *(\/false|\/nologin)$*. Fa servir el $ per indicar l’acabament de la línia; observar que s’ha d’escapar el caràcter / per incloure-ho com a caràcter dins del patró.

##### Exemple: extracció d’adreces de correu

Veiem una aplicació clara, en aquest cas fent servir ***grep***; extreure totes les adreces de email d’un fitxer, tot i que és cert que les adreces de correu avuí dia poden tenir formes més complexes que la presentada aquí, que és una variant simplificada:

[cesc@Skye pràctiques]$ \

grep -Eo '\<[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\>' *fitxer*

Amb una tècnica semblant els robots de propaganda roben adreces de les pàgines publicades en els llocs d’internet.

##### Exemple: Extreure UID\_MIN de /etc/login.defs

[cesc@Skye pràctiques]$ # Defineix la variable UID\_MIN obtinguda de /etc/login.defs

[cesc@Skye pràctiques]$ readonly UID\_MIN=$(sed -En 's/^UID\_MIN[[:blank:]]\*([0-9]+)$/\1/p' /etc/login.defs)

##### Exemple: Seleccionar un usuari aleatòriament.

Per extreure un usuari aleatòriament podem fer servir ***sed -n***, i escriure la línia amb l’ordre **p**:

[cesc@Skye pràctiques]$ cat /etc/passwd | cut -d: -f1 | sed -n $((RANDOM % $(wc -l < /etc/passwd) + 1 ))'p'

#### Tractament numèric

Una limitació important dels programes de tractament lèxic (grep, sed) és que ***no poden interpretar els caràcters que llegeixen***, així que 1000 no és més que el dígit ‘1’ seguit de tres ‘0’.

El tractament numèric de les dades es requereix la seva interpretació, i això ho fan possible llenguatges de programació més generals, com podria ser ***Python***, ***Perl*** o el més antic ***Awk***… en realitat n’hi ha molts llenguatges de programació.

Existeixen alguns filtres que incorporen la interpretació numèrica de les dades (escrites en text), que, a més, poden tenir diferents formats: diferent base (octal, hexadecimal…), positius i negatius, formats en punt flotant amb exponent... En concret recordem que **sort** i **uniq** permeten la interpretació de camps numèrics, però també tenen algunes limitacions en els formats que són capaços de considerar.

##### Exemple: Seleccionar un usuari amb UID >= 1000

Bé, considerem aquí que UID\_MIN és 1000, un valor habitual en els sistemes actuals. Com tractarem el valor 1000 ***lèxicament***, és a dir, com al dígit ‘1’ seguit de tres ‘0’, no generalitzem el problema a un valor numèric qualsevol

, problemàtica que ja s’ha comentat que excedeix les funcionalitats bàsiques de ***grep*** i ***sed***.

Si volem seleccionar un usuari amb UID >= 1000, podem fer una funció, on primer seleccionem els usuaris, i després treiem un aleatòriament. Aquesta funció fa servir un fitxer temporal, creat amb l’ordre ***mktemp***.

#!/bin/bash

#

# rndusr.sh

#

# funció per obtenir un nom d'usuari aleatòriament de /etc/passwd

function rndusr() {

local tmpusr=$(mktemp /tmp/rndusrXXX)

cat /etc/passwd | grep -E ':1[0-9]{3}:' | cut -d: -f1 > $tmpusr

sed -n $((RANDOM % $(wc -l < $tmpusr) + 1))'p' $tmpusr

rm $tmpusr &> /dev/null

return 0

}

function rnduid() {

local tmpusr=$(mktemp /tmp/rndusrXXX)

cat /etc/passwd | grep -E ':1[0-9]{3}:' | cut -d: -f3 > $tmpusr

sed -n $((RANDOM % $(wc -l < $tmpusr) + 1))'p' $tmpusr

rm $tmpusr &> /dev/null

return 0

}

#### Modificació de les dades

Els exemples anteriors només demostren que ***sed*** pot fer les tasques que fa ***grep***, però és més senzill fer servir ***grep***.

Però la funcionalitat de ***sed*** no es limita a trobar línies que compleixin una condició, on ja podria ser suficient amb ***grep***, ni tan sols a trobar continguts, que també podria fer grep. Amb ***sed*** podem modificar les dades d’entrada per produir unes dades diferents en la sortida.

##### Exemple: comparar directoris d’usuari declarats i directoris existents en /home

[cesc@Skye pràctiques]$ comm -3 <(cat /etc/passwd | grep -E ":1[0-9]{3}:" | cut -f6 -d: | sort) <(ls -1 /home | sort | sed 's/^/\/home\//g')

/home/lost+found

Observem que es fa una crida a ***comm***, on no es mostra la tercera columna, que correspon a les línies que coincideixen a tots dos arxius. Es fan servir dos paràmetres, on cadascú s’ha indicat fent servir **<( … )**, que permet la substitució d’ordres, fent-les servir com a fitxers.

* El segon argument mostra el contingut de **/home** en una línia per a cada resultat, ordena les línies, i, fent servir ***sed***, afegeix ***/home*** al començament de cada línia, per construir la ruta absoluta de cada resultat.

[cesc@Skye pràctiques]$ ls -1 /home | sort | sed 's/^/\/home\//'

/home/cesc

/home/convidat

/home/lost+found

* En el primer argument selecciona amb ***grep*** les línies que compleixen l’expressió regular ":1[0-9]{3}:": *dos punts* (:) seguit del número 1 i després un caràcter entre el zero i el nou 3 vegades, acabat en *dos punts* (:), és a dir, un número entre 1000 i 1999 delimitat per davant i per darrera amb el caràcter de *dos punts*. De les línies resultants, ***cut*** agafa només la 6a columna, que és el directori d’inici de l’usuari, definint les columnes pel caràcter delimitador *dos punts*. Finalment l’ordena amb ***sort***.

[cesc@Skye pràctiques]$ cat /etc/passwd | grep -E ":1[0-9]{3}:" | cut -f6 -d: | sort

/home/cesc

/home/convidat

En la comparació, ignorant les línies que són iguals, en la segona columna obtenim la línia que diferencia el segon del primer.

[cesc@Skye pràctiques]$ comm -3 <(cat /etc/passwd | grep -E ":1[0-9]{3}:" | cut -f6 -d: | sort) <(ls -1 /home | sort | sed 's/^/\/home\//')

/home/lost+found

##### Altres exemples senzills

* Exemple: Eliminar comentaris (pe. del fitxer /etc/fstab)

[cesc@Skye pràctiques]$ sed 's/#.\*//g' /etc/fstab

* Exemple: Comentar totes les línies (pe. del fitxer /etc/fstab)

[cesc@Skye pràctiques]$ sed 's/^/# /g' /etc/fstab

* Exemple: Escriure entre parèntesi les lletres majúscules (pe del fitxer frases)

[cesc@Skye pràctiques]$ sed -E 's/([A-Z])/(\1)/g' *frases*

Observar que els parèntesi en l’expressió regular indiquen la subexpressió que en el segon terme de la instrucció **s** (substitute) es referència amb **\1**. Els parèntesi en el segon terme són els parèntesi que s’escriuen.

* Exemple: Escriure totes les línies entre cometes

[cesc@Skye pràctiques]$ sed 's/.\*/"&"/g' frases

En aquest cas no es referència cap subexpressió, sinó que amb la referència **&** es referència tot el contingut reconegut per l’expressió regular.

##### Exemple: Cas senzill d’extreure text d’una pàgina html

S’ha comentat que les expressions regulars tenen certes limitacions, com el cas de la verificació del tancament de parèntesi; aquests casos que excedeixen les possibilitats de les expressions regulars s’anomenen *llenguatges recursius*, i els *llenguatges de marques* com *html* generalment cauen en la categoria dels llenguatges recursius. Per tant, ***grep*** no és suficient per fer processaments automàtics d’aquests llenguatges, i el cas de ***sed***, sovint trobarem que el tractament seguint patrons d’expressions regulars no és el llenguatge més adequat per tractar llenguatges de marques estructurats en gramàtiques. Naturalment existeixen eines més potents i adequades per fer tractaments complexos en gramàtiques, com ara, el procediment *TagReader*, de *Perl*, o *html.parser* de *Python*.

Però en casos concrets, sí podem fer un tractament senzill, per exemple, eliminar tots els tags html d’un arxiu, trobant fins i tot que ***sed*** és l’eina més adequada per algunes tasques que, podriem qualificar de ***lexicogràfiques***.

###### Exemple “frases còmiques” en castellà

En aquest cas anem a extreure frases contingudes en un fitxer html, una darrera l’altra, cadascuna en una única línia

* 1a part: extreure un contingut d’una pàgina web, netejant les etiquetes.

Podem obtenir un llistat de frases còmiques en castellà, llegint-ho d’un lloc web <http://artigoo.com/lista-de-frases-comparativas-comicas>. Baixarem l’arxiu fent servir ***curl*** (*client url*) i el filtrarem amb ***sed*** per quedar-nos només amb les frases.

Observem que:

* Cal ignorar totes les línies fins trobar el títol:

***<h3>Frases comparativas cómicas:</h3>***

Ignorem des del començament del fitxer fins trobar aquest contingut:

0,/<h3>Frases comparativas cómicas:<\/h3>/d

* Acaba el text que ens interesa arribant al text:

***<p>FRASES COMPARATIVAS***

En trobar aquest text podem acabar:

/<p>FRASES COMPARATIVAS/q

* El contingut que volem de cada línia es troba entre les etiquetes **<p>** i **</p>**; la subexpressió que obtenim (entre parèntesi) és la part que escrivim referenciant-la amb **\1**., i és qualsevol cosa que trobem: **.\***. L’indicador ***g*** indica que l’operació s’ha de fer per totes les instàncies que es trobin, tot i que no trobarem més que una en cada línia. El resultat l’escrivim en la sortida amb el modificador ***p***, així que farem servir ***sed -n*** per no repetir la sortida amb allò que escrivim.

s/<p>(.\*)<\/p>/\1/gp

[cesc@Skye p2]$ curl http://artigoo.com/lista-de-frases-comparativas-comicas | sed -nE '0,/<h3>Frases comparativas cómicas:<\/h3>/d;/<p>FRASES COMPARATIVAS/q;s/<p>(.\*)<\/p>/\1/gp' > frases 2> /dev/null

* 2a part; Hem obtingut un fitxer amb frases; ara podem fer un programa semblant al conegut programa ***fortune*** de Linux, obtenint una línia aleatòria del fitxer. L’operació ***p*** ens permet escriure el resultat, i sed -n no mostrarà altres sortides més que allò que nosatres explícitament escrivim.

[cesc@Skye pràctiques]$ sed -n $((RANDOM % $(wc -l < ./frases) + 1 ))'p' < ./frases

Fent servir aquest petit programa, podem mostrar una d’aquestes frases, per exemple, cada vegada que obrim un pseudoterminal.

##### Exemple

Separar les dates de FITXER en día mes i any:

[cesc@Skye pràctiques]$ sed -r 's/([0-9]{2})[/-]([0-9]{2})[/-]([0-9]{4})/Dia: \1, Mes: \2, Any: \3/g' FITXER

##### Exemple: extreure frases cèl·lebres d’arxiu html

Una altra vegada fem l’operació d’extreure frases d’un lloc web. Aquesta vegada ho farem en dues parts.

En la primera part eliminem tota la primera part del fitxer html, i la part final

0,/<h2>.\*Frases.\*Cèl·lebres<\/h2>$/d

/<\/div><!-- .entry-content -->/,/^<\/htlm>$/d

/^[[:blank:]]\*$/d

s/&#8217;/'/g

s/&#8211;/-/g

s/&#183;/·/g

s/&#8220;/"/g

s/&#8221;/"/g

s/&#8230;/... /g

s/&nbsp;/ /g

s/&mdash;/-/g

s/&ndash;/-/g

s/<cite[^>]\*>/-/g

s/<\/cite>/-/g

s/Anònim//g

s/--/-/g

s/- -/-/g

s/>-</></g

s/<\/blockquote>/<\/blockquote>\n/g

s/\.-</-</g

### Casos complexos amb sed

El funcionament de sed ja l’haurem vist:

* Per a cada línia, es repeteixen totes les ordres per ordre una a una fins haver-les aplicat totes, i es copia la línia en la sortida (excepte si s’ha eliminat, o si cridem a ***sed -n***).
* Per a cada ordre s’ha de verificar que l’ordre és aplicable comprovant, si hi és, la part d’adreça, que funciona com a *condició*.
* L’ordre **d** elimina la línia, i per tant no es fan més accions per la línia.
* Les substitucions de l’ordre **s** modifiquen la línia.
* L’ordre **p** escriu la línia amb les modificacions que hi té.
* L’ordre **q** finalitza i acaba.

Considerem que volem eliminar d’un text els parèntesi, claus, claudàtors i cometes buits. Podem provar amb:

[cesc@Skye ~]$ sed -E 's/\(\)//;s/\{\}//;s/\[\]//;s/""//'<<< '() hola "" {} []'

hola

Però trobarem que tenim problemes per considerar casos imbrincats:

[cesc@Skye ~]$ sed -E 's/\(\)//;s/\{\}//;s/\[\]//;s/""//'<<< '([]) hola {""} (())'

() hola {} ()

Ens caldria tornar a executar-ho per eliminar tots els parèntesi. Per aquest motiu sed inclou la possibilitat d’executar bucles.

#### Etiqueta i instrucció de bifurcació en sed

| **instruccions sed de flux de control** | |
| --- | --- |
| **:** *etiqueta* | Defineix *etiqueta* |
| **b** *etiqueta* | Canvia el flux de control, saltant (branch) a etiqueta |
| **t** *etiqueta* | Canvia el flux de control, saltant a etiqueta si ha hagut substitució en l’ordre anterior. La lletra ***t*** és deguda a la paraula ***test***. |

**parèntesi.sed**

:inici

s/[[:blank:]][[:blank:]]+/ /

t inici

s/\(\)//

t inici

s/\{\}//

t inici

s/\[\]//

t inici

s/""//

t inici

s/^[[:blank:]]\*//

s/[[:blank:]]\*$//

[cesc@Skye ~]$ sed -Ef parèntesi.sed<<< '([]) hola {""} (())'

hola

#### Materials complementaris

* Manual de grep:

<https://www.gnu.org/software/grep/manual/grep.html>

* Manual oficial de sed: “*sed, a stream editor*”

<https://www.gnu.org/software/sed/manual/sed.html>

* Manual de sed, versió recomanada per qui vulgui entrar en profunditat:

<https://www.grymoire.com/Unix/Sed.html>

* Conjunt d’articles sobre sed, amb exemples interessants, per Irene Burn:
  + <https://es.ccm.net/faq/3052-sed-trucos-y-tips>
  + <https://es.ccm.net/faq/3026-sed-anadir-texto>
  + <https://es.ccm.net/faq/3031-sed-eliminar-una-o-varias-lineas-de-un-fichero>
  + <https://es.ccm.net/faq/3060-sed-introduccion-a-sed-parte-ii>
* Per saber-ne més: Introducció a sed; des de comandes senzilles, avançant mica en mica fins casos més complexes:
  + <https://www.americati.com/doc/sed/sed.html>
* Per saber-ne més: Exemples classificats (alguns no són senzills)
  + <http://www.sromero.org/wiki/linux/aplicaciones/uso_de_sed>

# Altres ordres relacionades amb els filtres

## Especials per la gestió redireccions

* **xargs**: converteix la sortida en arguments de línia.
  + <https://shapeshed.com/unix-xargs/>
  + <https://www.tecmint.com/xargs-command-examples/>
* **tee**: copia l’entrada en la sortida i en un arxiu.
* **pv**: indica el progrés d’una operació en un pipe.
  + <https://www.solvetic.com/tutoriales/article/7685-comando-pv-y-progress-linux-para-ver-el-progreso-comandos/>

## Visualització

* **cat**: visualitza un fitxer; és a dir, el còpia en la sortida estàndard.
* **tac**: el mateix que cat, però en ordre invers; comença per l’última línia.
* **nl** (*equivalent a cat -n*) : el mateix que cat, però afegeix el número de línia.
* **head**: el mateix que cat, però només les primeres línies, per defecte 10.
* **tail**: el mateix que cat, però només les primeres línies, per defecte 10. Per visualitzar els canvis en un fitxer (les línies que s’afegeixen), existeix l’opció -f que queda en execució i actualitza la visualització quan el fitxer és modificat.
* **hexdump**: mostra en hexadecimal (i altres formats) el contingut d’un fitxer; el formata per facilitar la visualització binària.
* **od**: mostra en octal (i altres formats) el contingut d’un fitxer; el formata per facilitar la visualització binària.
* **fold**: Divideix les línies a una mida, per defecte 80 caràcters. Es pot indicar que només divideixi on hagi un espai.
* **more**:
* **less**:
* **banner**: Mostra el text del paràmetre en lletres grans, dibuixades fent servir caracteres ASCII.
* **cowsay**: Mostra un dibuix ASCII d’una vaca dient allò que indiquem com a paràmetre. Fent servir ***cowsay -f tux*** serà la mascota tux, en comptes d’una vaca.

## Comparació d’arxius

* **diff**: compara dos fitxers per línies, indicant les linies on s’han de fer modificacions, o línies que s’han d’afegir, o línies que s’han d’eliminar, perquè els fitxers siguin iguals.
* **comm**: compara fitxers per línies, però han d’estar ordenats.
* **cmp**: compara fitxers a nivell de caràcter, fins trobar una diferència.

## Altres ordres

* **paste**: Combina les línies de dos o més fitxers, separant-les amb un separador. Normalment els fitxers tindran el mateix número de línies.
* **split**: Divideix un fitxer. Es pot indicar en quants trossos i que no divideixi cap línia:
  + Exemple: Dividir en 10 trossos sense dividir línies: ***split -nl/10***
  + Pot ser útil per ordenar un arxiu molt gran:
    1. Dividir en parts
    2. Ordenar les parts
    3. Merge de les parts ordenades amb ***sort -m***
* **sleep**: (moure a apartat de processos)
* **exec**: (manca incloure-la en l’apartat de scripts, junt amb source)