12. Regularis kifejezesek

A **reguláris kifejezés** (rövidítve: **regexp** vagy **regex** az angol *regular expression* után) egy olyan, bizonyos szintaktikai szabályok szerint leírt string, amivel meghatározható stringek egy halmaza.

Az ilyen kifejezés valamilyen minta szerinti szöveg keresésére, cseréjére, illetve a szöveges adatok ellenrzésére használható.

Például egy érvényes (nem feltétlenül még él személyt jelöl) személyi szám biztosan a következ elemekbl áll:

- egy 1 és 8 közötti számjegy;
- egy szóköz;
- 19 vagy 20 vagy 21 és még két számjegy (év);
- utána
  - egy 0 és egy 1-9 közötti számjegy vagy
  - egy 1 és egy 0-2 közötti számjegy (hónap);
- utána
  - egy 0-2 közötti számjegy és egy 0-9 közötti számjegy vagy
  - egy 3, amit 0 vagy 1 követ (nap)
- egy szóköz
- és még négy számjegy.

(Esetleg a szóközök helyén kötjel is állhat.)

A programnak megmondhatjuk, hogy keresse meg az összes ilyen minta szerinti karaktersorozatot, majd pedig a kötjeleket cserélje szóközre, hogy egységes legyen (vagy a kötjeleket és a szóközöket is nem törhet szóközre, hogy egy weblapon egyben maradjanak). Egy másik példa: egy érvényes, dr. és más eltagok nélküli magyar személynév a magyar ábécé betibl, szóközökbl és kötjelekbl állhat a következ megszorításokkal: kötjel és szóköz nem állhat az elején és a végén, sem másik kötjel vagy szóköz után, csak két bet között; az els bet, valamint a szóközök és kötjelek utáni betk nagyok, a többi pedig kicsi; végül legalább egy szóköznek mindenképpen lennie kell. Ez a két példa nem garantálja, hogy amit találtunk, az biztosan érvényes személyi szám vagy magyar név, csak azt, hogy ily módon az összeset megtaláltuk. De ha olyan karaktersorozatot keresünk, amelyik 1 vagy 2 vagy 3 I (nagy i bet) után egy pontot, egy szóközt és az András vagy az Endre szavakat tartalmazza, akkor biztosak lehetünk benne, hogy egy valódi magyar király nevét találtuk meg (a római I a keresés szempontjából betnek számít, és még arra is figyelnünk kell, hogy a sorozatot megelz karakter bármi más lehet, csak még egy I nem, különben a reguláris kifejezésünk a „IIII. András” stringet is megtalálná, ami hiba lenne). Ha ugyanezt elfogadjuk nulla vagy egynél több szóközzel is, de ezeket egyre cseréljük, akkor hibás alakokat is találhatunk, és egy gyakori gépelési hibát javíthatunk ki. Ha ragozott alakokat is keresni akarunk, akkor figyelnünk kell rá, hogy az Endr**é**bl é bets alak lehet, míg az András szót változatlan marad.

A reguláris kifejezéseket sok szövegszerkeszt, illetve segédprogram használja, fleg szövegek keresésekor vagy szövegek bizonyos minták szerinti kezelésekor.

A reguláris kifejezéseket a jelsorozatokkal, stringekkel való mveleteknél több programozási nyelv is használja, illetve támogatja. Például a Perl, a Python és a Tcl is rendelkezik direkt reguláris kifejezések elemzésére szolgáló szintaktikai elemzvel. A különböz Unix-disztribúciókban lév segédprogramok jelents része (beleértve a sed szövegszerkeszt és a grep szrt) támogatta elször a reguláris kifejezések használatát.

A gyakran *mintának* nevezett reguláris kifejezés a jelsorozatok, stringek egy halmazát határozza meg. A minták használatával tömören megadhatók halmazok leírásai anélkül, hogy az összes elemüket fel kellene sorolni. Tegyük fel például, hogy egy halmaz a következ jelsorozatokat tartalmazza: *Handel*, *Händel*, és *Haendel*. Leírhatók-e a halmaz elemei a `H(ä|ae?)ndel` mintával (más szavakkal: mondhatjuk-e, hogy a mintához mindhárom string illeszkedik)? Amint az a késbbiekbl kiderül, általában azonos halmazok különböz mintákkal is leírhatóak.

A legtöbb formalizálásnál a következ **operátorok** használatával konstruálhatók meg a megfelel reguláris kifejezések.

választásA függleges vonal (|) a lehetséges alternatívákat választja el. Például a „kap|kép” minta alternatívákhoz illeszkedik a kap vagy a kép jelsorozat is.csoportosításA zárójelek az operátorok hatásköre elsbbségének a meghatározására szolgálnak. Például, a kap|kép és k(a|é)p minták különböznek, de ugyanazok a jelsorozatok illeszkednek hozzájuk (kap és kép).mennyiségjelzésA mennyiségjelz egy karakter vagy csoport után azt határozza meg, hogy hányszor fordulhat el a megelz kifejezés. A leggyakoribb mennyiségjelzk a ?, a * és a +:?A kérdjel jelzi, hogy a megelz kifejezés csak 0 vagy 1 esetben fordulhat el. Például, a colou?r minta illeszkedik a color és a colour jelsorozatok közül bármelyikre.*A csillag jelzi, hogy a megelz kifejezés akárhány esetben fordulhat el (beleértve a nullát is). Például, go*gle minta illeszkedik a ggle-e, a gogle-re, a google-re stb. Nagyon fontos, hogy a * értelmezése alapveten eltér a Windows-beli megszokott értelmezéstl!+A plusz karakter jelzi, hogy a megelz kifejezés legalább 1 esetben fordulhat el. Például a go+gle mintához illeszkedik a gogle, google stb. (de a ggle nem!).

A fenti konstrukciók egymással kombinálva a különféle formák komplex ellenrzését teszik lehetvé.

Tehát a `H(ae?|ä)ndel` és a `H(a|ae|ä)ndel` érvényes, szabályos minták, és ezen túlmenen, mindkett illeszkedik a elz példaként megadott jelsorozatokra.

Másik példa a elzekben leírt operátorok kínálta lehetségek kihasználásra:

Legyen a minta a következ:
`(nagy ?)*(apa|anya)`

A minta illeszkedik a következ stringek közül bármelyikre: *apa*, *anya*, *nagy apa*, *nagy anya*, *nagy nagy apa*, *nagy nagy anya*, *nagyapa*, *nagyanya*, *nagy nagyapa*, *nagy nagyanya*, *nagy nagy nagyapa*, *nagy nagy nagyanya* és így tovább.

A reguláris kifejezések pontos szintaxisa a változó eszközök és alkalmazások miatt egységesen nem adható meg; a további részleteket lásd a *Szintaxis* résznél.

„Tradicionális” Unix reguláris kifejezések[szerkesztés]

Az „alap” Unix reguláris kifejezéseinek definíciói a POSIX megjelenésével ugyan elavultak, ennek ellenére széles körben használatosak a visszafelé kompatibilitás miatt. A legtöbb reguláris kifejezést felismer Unix (segéd)program, például a *grep* és a *sed* alapértelmezésként használja még ezeket.

Ebben a szintaxisban a legtöbb karaktert literalként kezelik – saját magukra illeszkednek csak („a” illeszkedik „a”-ra, „(bc” illeszkedik „(bc”-re stb.). A kivételek az **illeszt-karaktereknek** nevezett metakarakterek:

.	Illeszkedik bármilyen egyedülálló karakterre
[ ]	Illeszkedik arra az egyedülálló karakterre, ami a zárójelek között fel van sorolva. Például, [abc]-re illeszkedik „a”, „b”, vagy „c”. [a-z]-re illeszkedik bármelyik kisbet. A jelölések keverhetk: [abcq-z]-re illeszkedik a, b, c, q, r, s, t, u, v, w, x, y, z, de így is írható: [a-cq-z]. A '-' karakter literálként viselkedik, ha a zárójelen belül els vagy utolsó helyen szerepel: [abc-] vagy [-abc]. A'[' vagy ']' karakterek illesztése nagyon egyszer, csak a záró szögletes zárójelet elbb kell írni a nyitó szögletes zárójelnél: [][ab]-re illeszkedik ']', '[', 'a' vagy 'b'.
[^]	Illeszkedik egy egyedülálló karakterre, amely nem szerepel a zárójelben felsoroltak között. Például, [^abc] illeszkedik bármilyen karakterre, amelyik nem „a”, „b”, vagy „c”. [^a-z] illeszkedik bármilyen egyedülálló karakterre, ami nem kisbet. Amint azt már említettük, ezek a jelölések is keverhetk.
^	Illeszkedik a sor kezdetére (vagy bármelyik sorra, többsoros mód alkalmazása esetén)
$	Illeszkedik a sor végére (vagy bármelyik sorra, többsoros mód alkalmazása esetén)
\(\)	Egy „megjelölt alkifejezést” definiál. Az alkifejezés illeszkedés késbb is ellenrizhet. Lásd a következ, *n* pontot. Megjegyezzük, hogy a „megjelölt alkifejezés” helyett gyakran a „blokk” kifejezés használatos.
\ *n*	Ahol *n* egy 1 és 9 közötti számjegy; az *n*-edik megjelölt alkifejezést illeszti. Ez a konstrukció elméletileg **szabálytalan** és nem engedi meg minden reguláris kifejezés kiértékel szintaxisa.
*	• Egy egyszer karakter kifejezést követ „*” a kifejezés nulla vagy több másolatát illeszti. Például, „[xyz]*” illeszkedik a következkre „”, „x”, „y”, „zx”, „zyx”, és így tovább. • *n**, ahol *n* egy 1 és 9 közötti számjegy, nulla vagy több iterációval illeszti az *n*-edik megjelölt alkifejezést. Például, „\(a.\)c\1*” illeszkedik „abcab”-re és „abcaba”-re, de nem illeszkedik „abcac”-re. • A „\(” és „\)” közé zárt kifejezést követ „*” érvénytelennek tekintend. Néhány esetben (például /usr/bin/xpg4/grep a SunOS 5.8 esetén), nulla vagy több iterációval illeszti a zárójelek közötti kifejezést. Másik esetben(például /usr/bin/grep a SunOS 5.8 esetén), illeszti a „*” eltti zárójelben lév kifejezést.
{ *x* , *y* }	Az utolsó „blokkot” legalább *x*-szer, de legfeljebb *y*-szor illeszti. Például, „a{3,5}”-re illeszkedik „aaa”, „aaaa” vagy „aaaaa”. megjegyezzük, hogy ez sem található meg minden regex megvalósításban.

Meg kell jegyezni, hogy bizonyos reguláris kifejezés elemz megvalósítások a \ metakaraktert másként értelmezik, mint azt az elzekben mutattuk.

A *grep* régi változatai nem támogatják a választási operátort jelöl „|” karakter használatát. Példák:

„.ép” illszkedik bármilyen háromkarakteres stringre, mint például kép, lép vagy tép és nép„[kl]ép” illeszkedik kép-re és lép-re„[^t]ép” illeszkedik minden, az elzekben leírt „.ép” kifejezésnek megfelel stringre, kivéve a tép-et„^[kl]ép” illeszkedik kép-re és lép-re, de csak akkor, ha ezek a stringek a sor elején állnak„[kl]ép$” illeszkedik kép-re és lép-re, de csak ha a sor végén állnak

A különféle beállításoktól, értelmezésektl függen változhat a karakterek sorrend szerinti csoportosítása (például bizonyos beállítások szerint a karakterek sorrendje az abc..yzABC..YZ szerinti, míg más elv szerint az aAbBcC..yYzZ a sorrend) a – nem mindenki által elfogadott – POSIX szabvány meghatározza a karakterek osztályokba vagy csoportokba sorolást a következ tábla alapján:

POSIX osztály	Megfelel	Jelentése
[:upper:]	[A-Z]	Nagybetk
[:lower:]	[a-z]	Kisbetk
[:alpha:]	[A-Za-z]	Nagy- és kisbetk
[:alnum:]	[A-Za-z0-9]	Számjegyek, nagy- és kisbetk
[:digit:]	[0-9]	számjegyek
[:xdigit:]	[0-9A-Fa-f]	hexadecimális számjegyek

[:punct:]	[.,!?:…]	elválasztó karakterek
[:blank:]	[\t]	szóköz és TAB
[:space:]	[\t\n\r\f\v]	üres karakterek
[:cntrl:]		vezérl karakterek
[:graph:]	[^ \t\n\r\f\v]	nyomtatásvezérl karakterek
[:print:]	[^\t\n\r\f\v]	nyomtatásvezérl karakterek és szóköz

Például: [[:upper:]ab] meghatározza az összes nagybett és a kisbets 'a'-t és 'b'-t.

(Egy ASCII kódtáblán színesben mutatja a különböz POSIX osztályokat a következ link http://www.billposer.org/Software/RedetManual/builtincharclass.html.)

Megjegyezzük, hogy a magyar ábécé ékezetes betinek kezelése esetenként eltér módon történik, ugyanis a különféle szabványok nem minden esetben térnek ki a különböz országokban használatos, az angol ábécé betitl eltér betk kódolására, kezelésére. Gyakori probléma például, hogy az ábécé szerinti rendezésnél az ékezetes betk rossz helyre kerülnek, mivel a karakterek kódjai – többnyire – a nem ékezetes karakterek kódjai után következnek, tehát rendezés szempontjából azok valóban „nincsenek sorban”.

POSIX modern (bvített) reguláris kifejezések[szerkesztés]

Több modern, „bvített” reguláris kifejezést használhatnak a modern Unix segédprogramok a parancs sorban az „-E” jelz hatására.

A POSIX bvített reguláris kifejezéseinek szintaxisa néhány kivételtl eltekintve megegyezik a „tradicionális” Unix reguláris kifejezéseivel. A következ metakaraktereket értelmezi még a program:

+	Az utolsó „blokk” egyszeri vagy többszöri illesztése – „ba+” illeszkedik a következkre: „ba”, „baa”, „baaa” és így tovább.
?	Az utolsó „blokk” illesztése (csak egyszer) vagy nem illesztése – „ba?” illeszkedik a következkre: „b” vagy „ba”
|	A választás (vagy unió képz) operátor: az operátort megelz vagy követ kifejezést illeszti – „abc|def”-re illeszkedik „abc” vagy „def”.

A „visszatörtjel” eltávolítása: \{…\} átalakul {…}-re és \(…\) átalakul (…)-re. Példák:

„[hc]+at” illeszkedik a következkre: „hat”, „cat”, „hhat”, „chat”, „hcat”, „ccchat” stb.„[hc]?at” illeszkedik „hat”-ra, „cat”-ra és „at”-ra„([cC]at)|([dD]og)” illeszkedik a következkre: „cat”, „Cat”, „dog” és „Dog”

A speciális jelentéssel bíró '(', ')', '[', ']', '.', '*', '?', '+', '^' és '$' karakterek az escape-zés használatával literálként lesznek értelmezve. Ez azt jelenti, hogy a karatert megelzi a '\' karakter, amely így „escape-ezve” már saját literálját jelenti csak. Példa:

„a\.(\(|\))” illeszkedik az „a.)” vagy az „a.(” stringekre

grep

Adott mintát tartalmazó sorok megjelenítése

REGULÁRIS KIFEJEZÉSEK

A Unix eszközök közül nagyon sok használ mintaillesztést, s a megadott mintára illeszkedõ adatokon további feldolgozást hajt végre. Vannak olyan parancsok, ahol a felhasználó adja meg a keresendõ mintát, ilyen a grep parancs, másokban rejtve dolgozik a mintakeresõ algoritmus. Reguláris kifejezések használatakor egy komplex mintát adunk meg (ez a reguláris kifejezés), és azt vizsgáljuk, hogy a feldolgozandó adatok melyik része illeszkedik a megadott mintára. A reguláris kifejezések karakterekbõl állnak, ezek közül néhány speciális jelentest hordoz, ezeket metakaraktereknek nevezzük.

A reguláris kifejezéseket meghatározó szabályok:

- Egy egyedülálló karakter, amely nem újsor karakter, és nem a . * [ ] \ ^ $ karakterek egyike, önmagára illeszkedik. Ez azt jelenti, hogy ha a reguláris kifejezés egy a betû, akkor ez a vizsgált szövegben csakis egy darab a karakterre fog illeszkedni.
- A \c karakterpáros, ahol c egy látható karakter, a c karakterre illeszkedik a karakter literális értelmében. Tehet a * illeszkedik a * -ra, a \\ pedig a \ -re.
- A . (pont) karakter egy olyan reguláris kifejezés, amelyik bármelyik (nem újsor) karakterre illeszkedik. Így pl. az ab. minta illeszkedik az a ba, abb, abc, ..., abz, ab0, ... karaktersorokra.
- Ha e egy reguláris kifejezés, akkor e* egy olyan reguláris kifejezés, amely az e reguláris kifejezés 0 vagy többszöri elõfordulását jelzi. Így az a* reguláris kifejezés illeszkedni fog az üres sztringre (hiszen abban zérószor szerepel az a karakter), valamint az a, aa, aaa, ... karakterláncokra.
- A [ ] zárójelbe tett karaktersorozat illeszkedik az abban a pozícióban levõ bármely, a zárójelben felsorolt karakterre. A karakterek

felsorolására vonatkozó szabályok:
    - kódjukat tekintve az egymás után következõ karaktereket rovidíteni lehet a kötõjel használatával. Pl. 0-9a-z jelenti az összes számjegyet és az angol abc összes kisbetûjét.
    - a nyitó zárójelet követõ ^ jel a felsorolt karakterek tagadása. Pl. [^0-9] jelenti a bármely nem szám karaktert.
    - a] zárójel csak a felsorolás elsõ tagja lehet
    - a - karaktert a \- karakterpáros jelenti
- Két egymás után írt reguláris kifejezés szintén reguláris kifejezés. pl. [^0-9][0-9] kifejezés a nem szám karaktert követõ szám karaktersorra illeszkedik.
- Két egymástól | jellel elválasztott reguláris kifejezés illeszkedik akar az egyik akar a másik kifejezésre.
- A ^ jel a sor elejére, a $ jel a sor végére illeszti a mintát.

Speciális karakterek
Mivel a reguláris kifejezések mint interpolált füzérek kerülnek kiértékelésre, ezért a következk is használhatók:

\t	tabulátor (HT, TAB)
\n	újsor, soremelés (LF, NL)
\r	kocsivissza karakter (CR)
\f	lapdobás (FF)
\a	cseng (bell) (BEL)
\e	escape (ESC)
\033	oktális kód megadása (mint a PDP-11 nél)
\x1B	hexa karakter
\c[	kontrol karakter
\l	a következkarakter kisbetsre konvertálva
\u	a következ karakter nagybetsre konvertálva
\L	kisbetsek a következ \E-ig
\U	nagybetsek a következ \E-ig
\E	kisbet, nagybet módosítás vége
\Q	reguláris kifejezések meta karakterei elé fordított törtvonalat ra a következ \E-ig

Ezeken kívül a Perl még a következket is definiálja

\w	egy szóban használható karakter (alfanumerikus karakterek és aláhúzás)
\W	minden ami nem \w
\s	szóköz fajta karakter, szóköz, tabulátor, újsor stb.
\S	minden ami nem \s
\d	számjeg
\D	nem számjegy

Ezek a jelölések karakterosztályok megadásában is használhatók, de **nem** intervallum egyik végén.

A következk is használhatók még reguláris kifejezésekben:

\b	szó határának felel meg
\B	nem szó határnak felel meg
\A	csak a füzér elején
\Z	csak a füzér végén
\G	ott folytatja ahol az elz m//g abbahagyta

A \b karakterosztályban használva visszalépés karakternek felel meg. Egyébként mint szóhatár olyan helyet jelöl, ahol az egyik karakter \w és a mellette lev \W. Ennek eldöntésére a füzér elején az els karakter elé és a füzér végén az utolsó karakter utánra egy-egy \W karaktert képzel a rendszer.

A \A és \Z ugyanaz, mint a ^ és a $ azzal a különbséggel, hogy ezek még a m opció használata esetén sem felelnek meg a füzér belsejében lev soremelés karakternek.

Szintaxisa:

```
grep  [ kapcsolók ]  minta    [ fájl ... ]
```

Adott az alábbi belépési napló:

belepesinaplo.txt

2005.10.25 18:30 alex 196.145.43.3

2005.10.26 18:30 joe 216.45.3.2

2005.10.26 18:44 richard 196.145.43.3

2005.10.28 22:31 joe 196.145.43.3

2005.10.29 18:00 alex 196.145.43.3

A futtatás eredménye:

```
grep alex belepesinaplo.txt

2005.10.25 18:30 alex 196.145.43.3
2005.10.29 18:00 alex 196.145.43.3
```

A grep -v azokat jeleníti meg, amelyek nem tartalmazzák a sorokat.

```
grep -v alex belepesinaplo.txt
2005.10.26 18:30 joe 216.45.3.2
2005.10.26 18:44 richard 196.145.43.3
2005.10.28 22:31 joe 196.145.43.3
```

Az alábbi naplóállomány a következ mezket tartalmazza:

dátum	id	felhasználónév	ipcím	letöltött adatmennyiség

naplo.log

2005.12.25 18:30 alex 196.145.43.3 1960 bytes

2005.12.26 18:30 joe 216.45.3.2 1512 bytes

2005.12.26 18:44 richard 196.145.43.3 2005 bytes

2005.12.28 22:31 joe 196.145.43.3 2006 bytes

2005.12.29 18:00 alex 196.145.43.3 2050 bytes

2006.01.02 08:25 joe 195.166.29.5 2008 bytes

2006.01.02 12:20 alex 195.165.1.1 2005 bytes

Feladatunk, hogy listázzuk a 2006-os eseményeket. Ha most a fentebb tárgyalt módon csak ennyit írunk:

```
grep 2006 naplo.log
```

úgy azok a sorok is megjelennek ahol a letöltött byte-ok száma 2006. Ezért ez nekünk nem jó. Jeleznünk kell, hogy csak azokat a sorokat szeretnénk megjeleníteni, ahol a 2006 a sor elején szerepel. Ehhez a ^ karaktert használjuk:

```
grep ^2006 naplo.log
```

Fájlnevek megjelenítése tartalom alapján: Például keressük az aktuális könyvtárban, az összes fájl között, azokat a fájlokat, amelyek tartalmazzák a VGA szót, megjeleníteni azonban csak a fájlneveket szeretnénk:

```
grep -l VGA *
```

A karakterosztályok használata

naplo.log

2005.12.25 18:30 alex 196.145.43.3 1960 bytes

2005.12.26 18:30 joe 216.45.3.2 1512 bytes

2005.12.26 18:44 richard 196.145.43.3 2005 bytes

2005.12.28 22:31 joe 196.145.43.3 2006 bytes

2005.12.29 18:00 alex 196.145.43.3 2050 bytes

2006.01.02 08:25 joe 195.166.29.5 2008 bytes

2006.01.02 12:20 alex 195.165.1.1 2005 bytes

2006.01.03 07:02 mari 195.165.2.45 2007 bytes

A fenti állományban azokat a sorokat keressük, ahol a letöltés 2005, 2006, 2007 vagy 2008 van.

```
grep "200[5678] bytes" naplo.log
```

A negyedik karakter csak a szögletes zárójelben ”[ ]„ megadott négy karakter egyikére illeszkedik. Az 5,6,7 vagy 8-as karakterek valamelyikére. A szám után kötelezen egy szóköznek, majd utána bytes karakterek megköveteltek.

A szögletes zárójel lehetvé teszi intervallumok megadását [5-8]:

```
grep "200[5-8] bytes" naplo.log
```

Az intervallumok megadására használhatók betk is:

```
[a-z]
```

Az elbbi minta például az angol ábécé összes kisbetjére illeszkedik. A nagybetk:

```
[A-Z]
```

Természetesen megadhatók kisebb intervallumok is:

```
[c-k]
```

Üres sorok törlése

```
grep -v "^$" filename
```

```
grep . filename
```

```
grep -v '^[[:space:]]*$' filename
```

Hashmárk jeles sorok kihagyása

```
grep -v '\#' filename
```

Fájlok keresése tartalom alapján

```
grep -H -r “Rewrite” /etc/apache2/
```

```
man -L en grep
```

grep gyakorlat

Adott az alábbi naplóállomány részlet (/var/log/syslog):

```
Feb 21 18:32:12 server postfix: Connection,
Feb 21 18:32:12 server postfix: LOGIN, user=test@server.hu,
Feb 21 18:32:46 server postix: LOGOUT, user=test@server.hu
Feb 21 18:32:12 server pop3d: Connection,
Feb 21 18:32:12 server pop3d: LOGIN, user=test@server.hu,
Feb 21 18:32:46 server pop3d: LOGOUT, user=test@server.hu
```

Csak a pop3d-t tartalmazhó sorokat szeretnék megjeleníteni.

```
grep pop3d /var/log/syslog
```

Adott egy konfigurációs állomány (dspam.conf) ami tele van megjegyzésekkel. Szeretném kiszrni a megjegyzés sorokat és csak a beállításokat szeretném:

```
#
# DSPAM Home: Specifies the base directory to be used for DSPAM storage
#
Home /var/spool/dspam
```

Ekkor:

```
grep '^[^#]' dspam.conf
```

Fájl keresése tartalom alapján

```
grep -lir "keresett szöveg" /utvonal/konyvtar
```

egrep

Nem egyezik meg a grep -E használatával, mert annál több reguláris kifejezést ismer ha ezt használjuk.

Ehhez hasonló reguláris kifejezések is használhatók:

```
+,?, | és ()
```

fgrep

Fix vagy fast grep rövidítése. Megegyezik a grep -F használatával

On-line tanulas:

https://github.com/zeeshanu/learn-regex

On-line gyakorlas

https://regexone.com/