Säännöllisistä lausekkeista

Panu A. Kalliokoski

Date: 2004/02/05 20:18:37

Sisältö

1 Yleistä

Säännölliset lausekkeet (regular expressions) ovat voimakas työkalu tekstin etsintään eli poimintaan, muokkaukseen ja erityisesti yksinkertaisten muunnosten tekemiseen. Niihin törmää Unixissa joka puolella, ja koska niiden peruskäyttö on verrattain yksinkertaista, opettelun hyöty-vaivasuhde on melko hyvä. :)

Säännöllisistä lausekkeista puhutaan sen verran paljon erilaisissa ohjeissa ja muissa, että niille on tullut paljon lempinimiä, kuten *regexp*, *regex* tai vielä lyhyempi *RE*. Käytän tässä johdannossa lyhyyden vuoksi nimitystä *SL*.

SL:ista on olemassa kaksi (tai kolme) eri varianttia, "perinteiset" SL:t, "laajennetut" SL:t (vaikka oikeasti niiden toiminnallisuus on jotakuinkin sama ja lähinnä merkinnät ovat toiset), sekä uutena ryhmänä julmasti laajennetut "Perl-yhteensopivat" SL:t. Tässä tutustumme perinteisiin SL:siin, muun muassa siksi, että `grep` ja `sed` tukevat niitä sellaisinaan (`grep` myös laajennettuja `-e`-valitsimella) ja koska niissä on vähiten "maagisia" merkintöjä.

Linuxeissa on yleensä SL:ita kuvaava `man`-sivu jonka saapi näkyviin komennolla `man 7 regex`.

1.1 Mitä SL:t sitten ovat ja mihin niitä käytetään?

SL:t ovat tekstinpätkiä kuvaavia hahmoja, joita sovitetaan syötetekstiin. Esimerkiksi SL `jotain` on hyvin yksinkertainen SL, joka täsmää vain tekstinpätkään "jotain". Toisaalta SL `foo.*bar` on monimutkaisempi SL, joka täsmää mihin tahansa tekstiin, jossa on ensin pätkä "foo", sitten välissä mitä tahansa kirjaimia kuinka monta tahansa kappaletta (ml. 0), ja sen jälkeen "bar", eli esimerkiksi tekstinpätkiin "foonizbar", "fooabar", "foobar" ja "foomettofoo a mezin bar".

Mitä sille täsmäävälle tekstille sitten tehdään? Se riippuu siitä, missä yhteydessä SL:tta käytetään. Tutustumme kahteen ohjelmaan, joissa voi käyttää SL:ita:

`grep` tekee tekstistä poimintoja SL:den perusteella. Oletuksena (siis ilman mitään valitsimia) tulostaa syötteestä rivit, joiden sisällöstä jokin kohta täsmää kyseiseen SL:seen. Valitsimella `-o` tulostaa syötteestä vain SL:seen täsmänneet pätkät (ei koko riviä).

Huomaamme siis, että tähänastinen `grep`:n käyttömme on ollut erityistapauksia: olemme käyttäneet vain sellaisia säännöllisiä lausekkeita, jotka täsmäävät vain yhdenlaiseen tekstiin, kuten `jotain` yllä.

Esimerkiksi komento `grep 'foo.*bar' tiedosto.txt` näyttää `tiedosto.txt`:sta ne linjat, joilla on ensin jossain kohtaa "foo" ja sen perässä jossain kohtaa "bar".

`sed` Muuntaa tekstiä SL:den perusteella. (Itse asiassa `sed` on pieni ohjelmointikieli, jolla voi tehdä kaikenlaista muutakin, mutta tämä on sen ylivoimaisesti yleisin käyttö.) Tämä tapahtuu `sed`:n alakäskyn `s` avulla. `s` korvaa syötteestä SL:seen täsmänneet osat jollain toisella (annetulla) tekstillä, mutta päästää syötetekstin muuten läpi muuttumattomana.

Esimerkiksi komento `sed 's/taivas/maa/g' tiedosto.txt` näyttää tiedosto.txt:n sisällön siten, että jokainen "taivas"-sana (tai sanan osa) on korvattu "maa"-sanalla.

1.2 SL:ista teoreettisemmin

Jokainen SL määrittää tietyn (mahdollisesti äärettömän) joukon merkkijonoja, eli ne merkkijonot, jotka "täsmäävät" kyseiseen SL:seen. Tällaista merkkijonojen joukkoa kutsutaan perinteisesti *kieleksi*. SL:den avulla pystyy määrittämään kieliä, jotka kuuluvat *säännöllisten kielten* luokkaan (siksi SL:ita sanotaan SL:iksi). Kieliä, joiden järkevien ilmausten joukkoa ei pysty määrittämään SL:lla, ovat esimerkiksi useimmat ohjelmointikielet (jotka enimmäkseen kuuluvat *kontekstittomien kielten* luokkaan) ja luonnolliset kielet (kuuluvat *kontekstillisten kielten* luokkaan). Käytännössä tämä tarkoittaa, että on paljon kieliä, joita ei pysty kuvaamaan SL:illa ollenkaan, mutta silti niillä pystyy kuvaamaan hyvin monia asioita.

SL:t ovat hyvin tiivis merkintätapa säännöllisille kielille. Niiden intuitiivista ymmärtämistä helpottaa se, että ne muistuttavat täsmäämiään merkkijonoja: esimerkiksi `kai.aa` vastaa kaikkia merkkijonoja, joissa on ensin "kai", sitten mikä tahansa yksi kirjain, sitten "aa", esimerkiksi "kaivaa" ja "kaitaa". SL:den

perustavanlaatuinen ominaisuus onkin, että niissä *suurin osa kirjaimista vastaa itseään*. Esimerkiksi `a` täsmää tekstiin "a".

SL:den tiiviys on myös haitta. Lausekkeiden monimutkaistuessa niihin alkaa tulla paljon näitä maagisia merkkejä jotka eivät esitä itseään vaan jotain yleisempää (kuten piste `.` tai tähti `*`) ja niistä tulee nopeasti hyvin kryptisiä ja vaikeita lukea. SL:ita onkin kutsuttu silloin tällöin välimerkkitehtaan räjähdykseksi, koska suurin osa maagisista merkeistä on välimerkkejä.

Mutta joo. Edetkäämme katsomaan, mitä rakenteita niissä säännöllisissä lausekkeissa on käytettävissä.

2 Perus-SL:t

2.1 Itsestäänselvyydet

Nämä asiat useimmat ihmiset tajuavat luonnostaan, mutta niiden sanominen erikseen on silti hyvästä.

SL:den perusrakennuspala on *kirjain*. Jokainen tavallinen (siis ei-maaginen) kirjain on SL, joka täsmää itseensä. SL:den perusrakennustapa on *katenaatio* eli peräkkäin yhdistäminen. SL, jossa on peräkkäin SL s_1 ja SL s_2 täsmää tekstiin, jossa on ensin jotain, joka täsmää SL:seen s_1 ja heti sen perässä jotain, joka täsmää SL:seen s_2.

Esimerkiksi: SL `ka` täsmää tekstiin "ka", koska sen ensimmäinen osa `k` on SL, joka täsmää kirjaimeen k ja toinen osa `a` on SL, joka täsmää kirjaimeen a. Edelleen `kan` täsmää tekstiin "kan" koska sen ensimmäinen osa `ka` on SL, joka täsmää tekstiin "ka" ja toinen osa `n` on SL, joka täsmää kirjaimeen n.

Oliko tarpeeksi perusteellista? Tämä on siis "syy" siihen, miksi kaikki maagisia merkkejä sisältämättömät SL:t täsmäävät samannäköiseen tekstiin.

2.2 Ensimmäiset maagiset merkit

Esimerkkimme `foo.*bar` yllä sisältää kaksi maagista merkkiä: pisteen ja tähden. Molemmat ovat käytettävissä erikseen, vaikka usein ne näkeekin tällä tavoin yhdessä. Mitä ne siis tarkoittavat?

`.` Tämä on oikeastaan jo selitetty. Se täsmää mihin tahansa kirjaimeen. Syötetekstissä voi siis pistettä vastaavalla kohdalla olla mikä tahansa merkki. Tämä on yksinkertaisin *merkkiluokka* (siihen kuuluvat kaikki merkit).

`*` Tähti on *operaattori*, eli se muuttaa lähintä edeltävää SL:tta. Tähti sanoo, että sen sijaan, että edeltävään SL:seen täsmäävä merkkijono pitäisi löytyä syötteestä kerran, siitä saakin löytyä kuinka monta tahansa (mukaanlukien ei yhtään) peräkkäistä edeltävään SL:seen täsmäävää merkkijonoa.

Tähti toimii aivan hyvin myös tavallisten kirjainten muuntimena. Esimerkiksi `telefo*ni` täsmää sanoihin "telefoni", "telefooooooni" ja "telefni", muttei sanaan "telefuuni".

Lisää esimerkkejä:

SL	täsmää	ei täsmää
`j..urtt*i`	“jokurtti”, “jugurti”, “jYÅurttttttti”	“yogurtti”, “jukuri”
`Jarmo .*Puntti`	“Jarmo L. za-Puntti”, “Jarmo Puntti”	“Jarmo-san Puntti”, “Jarm Puntti”

2.3 Tarkemmat merkkiluokat

Jorkutthi-esimerkissämme yllä `.` oli tarpeettoman laaja luokka. Riittäisi, että kirjain voisi olla jokin muutamasta yleisestä variantista. Hakasulkeilla (`[` ja `]`) voi määrittää merkkiluokkia, joihin kuuluu tietyt kirjaimet. Esimerkiksi SL `[aieouyää]` täsmää mihin tahansa (suomen kielen) vokaaliin ja SL `t[ei][ae]` täsmää sanoihin “tie”, “tee”, “tea” ja “tia”. Edelleen SL `[Jj]oskus` täsmää sanaan “joskus” riippumatta, onko se kirjoitettu isolla vai pienellä alkukirjaimella.

Hakasulkeiden sisällä voi käyttää myös `tr`-mäisiä välejä, joista yleisimmät ovat `a-z` (kaikki pienet kirjaimet a:sta z:an), `A-Z` ja `0-9` (kaikki numerot). Siispä SL `[a-zääö]` täsmää mihin tahansa suomen kielen pieneen kirjaimeen ja SL `[a-zääö]*` minkä tahansa mittaiseen rimpsuun niitä (esim. kokonaan pienellä kirjoitettuun sanaan).

Hakasulkeilla voi muodostaa myös *käänteisluokkia* eli luokkia, johon kuuluvat kaikki *muut* kuin luetellut kirjaimet. Käänteisluokka merkitään `[^kirjaimia\ldots{}]` eli esimerkiksi SL `[^A-Za-z]` täsmää mihin tahansa merkkiin, joka ei ole englannin kielen sanakirjain.

Käänteisluokat ovat siitä merkittäviä, että niillä selviää tietystä hankalasta tilanteesta. Tähtirakenne `*` on nimittäin siitä hankala, että usein joudutaan tilanteeseen, jossa sillä on useampia laillisia täsmättäviä pituuksia. Esimerkiksi jos poimimme SL:tta `foo.*bar` ja syöte on “frobnize foo in bar for every bar”, laillisia poimintoja ovat “foo in bar” sekä “foo in bar for every bar”. Perinteisissä SL:issa useammista vaihtoehdoista valitaan aina pisin. Tämä ei kuitenkaan usein ole se, mitä halutaan: jos yritämme poimia sulkeissa olevia asioita SL:lla `(.*)`, syöte “joskus (muttei usein) menee pieleen (vahinko!)” tuottaa väärän tuloksen “(muttei usein) menee pieleen (vahinko!)”, kun olisi pitänyt tuottaa kaksi poimintoa “(muttei usein)” ja “(vahinko!)”.

Ratkaisu on korvata `.*` SL:sta lausekkeella, joka ei sovitu sulkeviin sulkeisiin. `([^)]*)` tuottaa oikean tuloksen. Yleensäkin tämä tapa kieltää erikseen `*`:n tuottamasta rimpsusta tietyt merkit on hyvin yleinen.

2.4 Rivin ja sanan alku ja loppu

Olen hiukan yrittänyt vastustaa kiusausta sanoa esimerkiksi, että SL `poika` täsmää *sanaan* “poika”. Tässä SL:ssahan ei ole mitään, mikä varsinaisesti pakottaisi täsmäävän tekstin olevan itsenäinen sana: myös sanasta “poikamies” täsmää siihen alkuosa ja merkkijonosta “hoopoikarus” keskiosa.

Perinteinen ratkaisu tähän on ottaa sanaa seuraava ja edeltävä merkki mukaan SL:seen, esimerkiksi jommallakummalla seuraavista tavoista:

- `[ ,.-]poika[ ;:,.-]`
- `[^-a-zääö]poika[^-a-zääö]` (viiva pitää panna paikkaan, jossa sitä ei voi tulkita välin merkinnäksi)

Tässä on se vika, että rivin alussa ja lopussa oleva sana jää poimimatta, koska sitä ei edellä / seuraa mikään merkki. Edelleen tähän perinteinen ratkaisu on ollut lisätä rivin alkuun ja loppuun esim. välilyönti ennen `grep`:lle antamista.

Tämä on sotkuista. Ongelmaan on keksitty erilaisia ratkaisuja, kuten maagisia merkintöjä, jotka täsmäävät sanojen alkuihin tai loppuihin, tai esim. `grep`:n valitsin `-w`, joka kertoo, että annetun SL:n tulee täsmätä itsenäiseen sanaan. Itse asiassa Linuxeissa sekä `grep` että `sed` ymmärtävät merkinnät `\<` ja `\>`, jotka täsmäävät sanan alkuun ja loppuun. Itse asiassa tarkempaa olisi sanoa, että ne täsmäävät tyhjään merkkijonoon sanan alussa tai lopussa, koska ne eivät siis täsmätyssä tekstissä vastaa yhtäkään kirjainta, edellyttävätpähän vain, että kyseisellä kohdalla on sanan alku / loppu. Mutta näiden merkintöjen tarjolla olemisesta ei ole takuita kaikissa SL-varianteissa.

Rivin alulle ja lopulle on sen sijaan olemassa jokikisessä SL-variantissa toimivat maagiset merkit:

`^` Täsmää rivin alkuun. Esimerkiksi `^-1)` täsmää merkkijonoon "-1)" rivin alussa ja komennolla `sed 's/^[ ]*//' tiedosto.txt` voi poistaa välilyönnit kaikkien rivien alusta (hakasulkeet eivät ole välttämättömät mutta saavat SL:n mielestäni helpommaksi lukea).

`$` Täsmää rivin loppuun. Esimerkiksi komennolla `sed 's/$/!/' tiedosto.txt` voi lisätä huutomerkin jokaisen rivin loppuun (sillä se korvaa tyhjän merkkijonon rivin lopusta huutomerkillä).

Tällä komennolla voi poimia rivit, joiden neljänneksi viimeinen kirjain on "a": `grep 'a...$' tiedosto.txt`

Vielä yksi esimerkki: seuraava komento poimii rivit, joilla ei ole mitään muuta kuin sana "turha":

```
grep '^turha$' tiedosto.txt
```

On muuten syytä huomata, että nämä erikoiset "paikkaehtomerkinnät" tekevät mahdolliseksi tehdä SL:ita, jotka eivät täsmää yhtään mihinkään merkkijonoon. Esimerkiksi `po\<ogs\>` ei voi täsmätä mihinkään, koska kahden o-kirjaimen väli ei koskaan ole sanan alku.

3 Vaativammat SL:t

Tähän mennessä käsitellyt ominaisuudet ovat sellaisia, että niitä tarvitsee vähän väliä ja ne ovat verrattain helppoja ymmärtää ja pähkäillä. Niillä saa yllättävän paljon aikaan ja monelle käyttäjälle ne riittävät aivan hyvin.

Tässä jaksossa käsittelemme rakenteita, jotka lisäävät SL:den kuvausvoimaa huomattavasti, mutta jotka ovat vaikeatajuisempia ja edellyttävät yhdistelykykyä.

3.1 Toisto-operaattorit

Olemme käsitelleet tähän mennessä vain yhden operaattorin (maagisen rakenteen, joka muuttaa jonkin SL:n merkitystä). `*` on yksinkertaisin toisto-operaattori: se määrittää, että edeltävä SL saa olla syötteessä peräkkäin kuinka monta kertaa

tahansa. Tämä on erikoistapaus toisto-operaattoreista, ja samoin kuin pisteen (`.`) tapauksessa, sen yleisyyttä voi rajoittaa.

Rakenne `\{n,m\}` (jossa n ja m ovat joitain numeroita) merkitsee, että edeltävän SL:n pitää täsmätä vähintään n ja enintään m peräkkäistä kertaa. Jos m puuttuu, ylärajaa ei ole. Esimerkiksi:

SL	merkitys
`a\{3,6\}`	3 - 6 a-kirjainta peräkkäin
`[aeiou]\{4,\}`	vähintään 4 englannin kielen vokaalia peräkkäin
`roc\{0,1\}k`	"rock" tai "rok"
`^[ ]*[0-9]\{3,\}`	vähintään 3 numeroa (sisennettynä) rivin alussa

Huomaa myös, että seuraavat SL:t täsmäävät samoihin merkkijonoihin eli määrittävät saman kielen:

1. `[abc]\{2,\}`
2. `[abc][abc][abc]*`

Laajennetuissa SL:issa on monille tyypillisille toistomäärille lyhenteitä. Niistä saapi tietoa `man 7 regex`-sivulta.

3.2 Sitovuusjärjestys (presedenssi)

Operaattorit muuntavat aina edeltävän SL:n merkitystä. Tarvitaan keino, jolla voi kertoa, kuinka suuri osa SL:sta kuuluu operaattorin muunnettavaksi. `\(` ja `\)` toimivat kuten sulkeet matematiikassa ja kertovat, missä järjestyksessä SL tulkitaan.

Jos halutaan pidempi kuin yhden merkin toistuva osuus, pitää sulkeet merkitä erikseen:

SL	täsmää
`bar*`	esim. ba, bar, barrrr
`\(bar\)*`	esim. bar, barbarbarbar ja tyhjä merkkijono
`mus\(ta\)\{1,2\}`	musta, mustata
`mus\(t[aeiou]\)*`	esim. mus, muste, mustuti, mustotata
`\(ha[ .h]*\)\{1,\}`	esim. hahaha, hah. hah., hahha.haha .hah.

Usein, kun alkaa tekemään jotain "hienoa" eli monimutkaista säännöllisillä lausekkeilla, pitää aloittaa yksinkertaisesta ja tarkentaa SL:tta pikku hiljaa saadakseen tarkempia tuloksia. Tällöin usein pitää muuttaa merkkiluokka joksikin suljelausekkeeksi. Esimerkkinä virkkeeseen täsmäävä SL (`[.]` tarkoittaa nimenomaan pistettä (eikä mitä tahansa merkkiä), sillä hakasulkeissa piste on tavallinen kirjain eikä maaginen):

SL	kommentti
`[A-ZÅÄÖ].*[.]`	liian laaja, .* voi mennä yli virkerajasta
`[A-ZÅÄÖ][^.]*[.]`	periaatteessa hyvä, mutta pisteloppuiset lyhenteet katkaisevat virkkeen
`[A-ZÅÄÖ]\([^.]\([.] [a-zääö]\)*\)*[.]`	vielä parempi

Tässä siis toistettava SL monimutkaistettiin siten, että virkerimpsussa saa olla (ei-pisteiden lisäksi) kohtia, joissa on piste, välilyönti ja sitten pieni kirjain. Ei toimi vieläkään esim. syötteelle "toht. Nurmi" mutta välttää.

Toinen esimerkki: rivi, jolla on kolmesta viiteen a-kirjainta:

SL	kommentti
`^a\{3,5\}$`	nyt rivillä ei voi olla mitään muuta kuin a-kirjaimia
`^\(a.*\)\{3,5\}$`	ei toimi, piste täsmää myös a-kirjaimeen
`^\(a[^a]*\)\{3,5\}$`	edellyttää ensimmäisen a:n rivin alkuun
`^[^a]*\(a[^a]*\)\{3,5\}$`	toimii

3.3 Takaisinviittaukset

Tämä hämmästyttävän monipuolinen ominaisuus puuttuu laajennetuista SL:ista (joten oikeasti ne ovat myös kavennettuja). Syyt tähän ovat hyvin teknisiä ja liittyvät SL:den toteuttamiseen ohjelmissa. Vaikka periaatteessa takaisinviittauksille on helppo keksiä käyttökohteita, yleensä niitä ei tarvitse, ja esittelenkin ne sen takia, että ne helpottavat `sed`:n kehittyneiden tekstikorvausten ymmärtämistä.

Toisto-operaattorit ovat edelleen hyvin sallivia. Ne eivät oikeasti edellytä toistoja *syötetekstissä* vaan pelkästään toistuvia täsmäyksiä edeltävään lausekkeeseen. Esimerkiksi `[klo]*` täsmää, paitsi merkkijonoihin "kkkk", "lll", "ooooo" ja tyhjä, myös merkkijonoon "olkkolokoo". Yleensä tämä on juuri se, mitä haluammekin. Mutta mitä, jos haluamme löytää oikeita toistoja tekstistä?

Tätä varten takaisinviittaukset ovat. Rakenteella `\n` (jossa n on jokin luku) voi viitata aiempien suljeilmausten (joita ympäröi `\(` ja `\)`) sisältöön. Esimerkiksi `\4` viittaa tekstiin, joka syötteessä täsmäsi neljänsien sulkeiden (laskien avaavia sulkeita vasemmalta lähtien) sisältämään SL:seen.

SL	täsmää
`\([klo]\)\1`	kk, ll, oo
`\([klo]\)\1*`	esim. k, llllll, oooo
`\(.\)\1\1`	esim. ooo, aaa,)))
`\([^a-z]\)[a-z]*\1`	sanaan, jota ennen ja jonka jälkeen on sama merkki
`\([a-zääö]\{1,\}\) \1`	kahdesti peräkkäin tulevaan sanaan

4 sed:n käyttö

4.1 Yksinkertaiset tekstikorvaukset

Yksinkertaiset tekstikorvaukset on helppo tehdä `sed`:lla. `sed`:lle annetaan argumentiksi (hipsuissa, yleensä) komento muotoa `s/mitä/mihin/valitsimia`, jossa *mitä* on SL, johon täsmäävä teksti korvataan, *mihin* on teksti, jolla se korvataan, ja *valitsimet* ovat `sed`:n `s`-komennon toimintaa muuttavia kirjaimia. Yleisin näistä on `g`, joka pyytää tekemään kaikki mahdolliset muutokset joka rivillä (muuten `sed` tekee vain ensimmäisen).

`s`-komento	syöte	tulos
`s/foo/bar/g`	kafoo on foo	kabar on bar
`s/aa*/A/g`	laakso ja jakso	lAkso jA jAkso
`s/a/e/`	paalu	pealu
`s/([^)]*)//g`	(Jos) on tärkeää (niin)	on tärkeää
`s/[yj][ou][gk]u/jugu/`	yokurtti	jugurtti

Itse asiassa `s`-komennon osia ei tarvitse erottaa toisistaan juuri kauttaviivoilla — mikä tahansa merkki käy. Kauttaviiva on perinteinen valinta, mutta jos SL tai korvaava teksti sisältää kauttaviivoja, kannattaa valita toinen merkki.

4.2 Monimutkaisemmat tekstikorvaukset

`s`-komennon *mihin*-osa voi sisältää takaisinviittauksia. Näin SL:n täsmäämästä tekstistä voi poimia osia ja uudelleenjärjestää ne tai lisäillä niihin jotain. Lisäksi koko täsmänneen tekstin saa `&`-merkistä.

`s`-komento	syöte	tulos
`s/^\(...\)\(.\)/\2\1/`	Paavo Kauppi	vPaao Kauppi
`s/\([A-Z][a-z]*\) \(.*\)/\2, \1/`	Paavo Kauppi	Kauppi, Paavo
`s/[a-zääö]\{1,\}/'&'/g`	juu ei siis	'juu' 'ei' 'siis'
`s/\([a-z]\)\([a-z]*\)@.*/\1. \2/`	pkalliok@ling	p. kalliok

Tällä tavalla saa usein esim. olennaisesti vähennetyksi työtä, jos pitää muuntaa jokin epämääräisessä muodossa oleva yhteystietoluettelo järkevään muotoon. Tai vaihtaa sanoja tiedostoista.