Robert Meolic
meolic@uni-mb.si

Regularni izrazi in orodje SED

interno gradivo za predmet VSO, 2006/07

1. LITERATURA

Pri sestavljanju gradiva o regularnih izrazih in orodju SED sem uporabljal internet in naslednji dve knjigi:

- EricFoster-Johnson, John C. Welch, Micah Anderson: Beginning Shell Scripting, Wiley Publishing Inc., 2005
- M. Tim Jones: GNU/Linux Application Programming, Charles River Media, Inc., 2005

2. REGULARNI IZRAZI

Pri iskanju datotek na računalniku smo navajeni posebnih znakov, s katerimi tvorimo vzorec imena, ki mu naj iskana datoteka ustreza. Npr. vzorcu `Data*.txt` ustrezajo datoteke z imeni `Data.txt`, `Data1.txt`, `DataNedelja.txt` itd. To vrsto vzorcev v angleščini imenujemo „file globs“. Za tvorjenje bolj kompleksnih vzorcev so vpeljali regularne izraze (v angleščini „regular expressions“), ki pa jih ne podpirajo vsi ukazi. V lupini Windows command npr. podpira regularne izraze ukaz `FINDSTR`, v lupini Bash pa ukaza `find` in `grep`. Po drugi strani ukaz `dir` v nobeni od teh lupin ne podpira regularnih izrazov.

Če ukaz podpira oba načina tvorjenja vzorcev, tistega preprostega in tistega z regularnimi izrazi, moramo biti pazljivi pri tem, kateri način se bo uporabil. Zelo pogosto uporabljani znak zvezdica in tudi drugi znaki imajo namreč pri teh dveh pristopih popolnoma različen pomen.

Na kratko lahko regularne izraze opišemo kot vzorce, v katerih so dovoljeni naslednji posebni znaki:

- `.` (pika pomeni katerikoli znak)
- `*` (zvezdica pomeni nič ali več ponovitev prejšnjega znaka oz. element
- `^` (strešica pomeni začetek vrstice, vsaka vrstica ima natanko en začetek)
- `$` (dolar pomeni konec vrstice, vsaka vrstica ima natanko en konec)
- `[razred]` (pomeni natanko en znak, dovoljen je katerikoli znak iz danega razreda)
- `[^razred]` (pomeni natanko en znak, dovoljen je katerikoli znak, ki ni iz danega razreda)
- `\<` (kombinacija \< označuje začetek besede)
- `\>` (kombinacija \> označuje konec besede)
- `|` (navpična črta pomeni logični or med dvema regularnima izrazoma)
- `(izraz)` (z okroglimi oklepaji ločimo podizraze med seboj)
- `\x` (povratna poševnica izniči poseben pomen posebnih znakov)

Beseda je zaporedje alfanumeričnih znakov (črke in številke), torej vejica, narekovaj, zvezdica in drugi posebni znaki niso sestavni del besede.

Razred je pri regularnih izrazih množica znakov. Podamo ga tako, da naštejemo vse znake enega za drugim (brez presledkov ali vejic!) ali pa da uporabimo območja. Pri podajanju razreda tudi vse posebne zanke vnesemo na običajen način, torej zapis `[.*]` pomeni pika ali zvezdica in nič drugega. Posebnost sta znaka strešica in minus. Če ju želimo imeti v razredu, strešice enostavno ne smemo podati na prvem mestu, medtem ko minus moramo podati na prvem mestu. Tukaj je nekaj primerov za podajanje območij:

- `[0-3]` pomeni enako kot `[0123]`
- `[a-k]` pomeni enako kot `[abcdefghijk]`
- `[A-C]` pomeni enako kot `[ABC]`
- `[A-Ca-k]` pomeni enako kot `[ABCabcdefghijk]`
- `[[:alpha:]]` pomeni enako kot `[a-zA-Z]`
- `[[:upper:]]` pomeni enako kot `[A-Z]`
- `[[:lower:]]` pomeni enako kot `[a-z]`
- `[[:digit:]]` pomeni enako `[0-9]`
- `[[:alnum:]]` pomeni enako kot `[0-9a-zA-Z]`
- `[[:space:]]` pomeni prazen prostor, torej presledke in tabulatorje

V tako imenovanih razširjenih regularanih izrazih (v angleščini "extended regular expressions") poznamo še dodatne posebne znake, med katerimi sta najpomembnejša naslednja:

- + (zvezdica pomeni ena ali več ponovitev prejšnjega znaka oz. elementa)
- ? (vprašaj pomeni nič ali ena ponovitev prejšnjega znaka oz. elementa)

Primeri:

- Iščemo datum 28.02.2006, ki je zapisan v navedeni obliki, ali v obliki 28/02/2006 ali pa v obliki 28-02-2006.

  Ustrezen regularni izraz je: `28[-./]02[-./]2006`

- V glavi elektronske pošte iščemo vrstice, ki se začnejo s poljem „From:“ in „To:“.

  Ustrezen regularni izraz je: `^(From|To):`

- Poišči vrstice, ki vsebujejo besedilo med parom dvojnih narekovajev.

  Ustrezen regularni izraz je: `“[^”]*“`

- Poišči vrstice, ki vsebujejo številko od 0 do 23. Številke od 0-9 so lahko zapisane z enim mestom ali pa kot dvomestno število z vodilno ničlo, torej 00, 01, 02 itd.

  Možni sta dve enako lepi rešitvi.

  - Prvi ustrezen regularni izraz je: `[01]?[0-9]|2[0-3]`
  - Drugi ustrezen regularni izraz je: `[01]?[4-9]|[012]?[0-3]`

V regularnih izrazih lahko uporabimo še mnoge druge operatorje. Za konec te kratke predstavitve bomo omenili le operatorje za sklicevanje na prejšnje podizraze. Besedilo se s pomočjo regularnih izrazov razpoznava od leve proti desni. Če uporabimo okrogle oklepaje s tem označimo, a je en del razpoznanega besedila zaključena celota, ki jo imenujemo podizraz. Z operatorjem `\1` zahtevamo, da se na nekem mestu pojavi enak podizraz, kot je bil nazadnje razpoznan. Obstajajo tudi operatorji `\2`, `\3`, itd. s katerimi se sklicujemo na predprejšnji, predpredprejšnji izraz itd.

Primer:

- V besedilu poišči vrstice, v katerih se zaporedoma pojavita dve enaki besedi.

  Ustrezen regularni izraz je: `\<([[:alpha:]])[:space:]]\1\>`

2. ORODJE SED

Sestavni del številnih skript je obdelava tekstovnih datotek tako, da v datoteki nekaj popravimo. Znan primer je npr. pretvorba med DOS in UNIX načinom zapisa nove vrstice. Če je tekstovna datoteka v formatu DOS, potem je konec vrstice označen z dvema znakoma CR in LF. Če je tekstovna datoteka v formatu UNIX, potem je konec vrstice označen le z znakom LF. Pretvorba iz DOS v UNIX torej pomeni, da je potrebno vse kombinacije CR/LF nadomestiti le z enim LF, pri pretvorbi iz UNIX v DOS pa je potrebno storiti obratno. Drug znan primer obdelave tekstovne datoteke je pretvorba kodne tabele za zapis šumnikov.

Skriptna jezika lupin Windows command in Bash vsebujeta nekaj ukazov a obdelavo teksta, vendar pa pri zahtevnejših obdelavah teksta dobili zelo dolge in nepregledne skripte. V takih primerih raje uporabimo katerega od namenskih orodij oz. skriptnih jezikov za obdelavo tekstovnih datotek. Najbolj popularno orodje za te namene je sed, med skriptnimi jeziki za obdelavo teksta pa je trenutno najbolj popularen perl. V nadaljevanju si bomo ogledali le orodje sed.

V operacijskem sistemu Microsoft Windows ni orodja sed, lahko pa ga namestimo. Dobimo ga na naslovu `http://gnuwin32.sourceforge.net/packages/sed.htm`. Pri uporabi orodja sed iz v lupini Windows command ukaze namesto v apostrofih napišemo v narekovajih.

Orodje sed si lahko predstavljamo kot urejevalnik, v katerega po vhodni cevi priteka tekst, sed ga obdela in ga po izhodni cevi spet pošlje naprej. Ta primerjava je dobra, ker ponazarja nekaj osnovnih lastnosti orodja sed:

- ko sed obdela tekst in ga odda naprej, ga ne more več spreminjati,
- ko sed obdeluje trenutni tekst ne more vedeti, kaj bo priteklo za njim, ne more torej pogledati naprej,
- si pa lahko sed zapomni, kaj je obdeloval prej in glede na shranjeno informacijo prilagodi trenutno obdelavo teksta.

Kot prvi primer vzemimo ukaz `sed 's/originalni/spremenjeni/g'`. S tem ukazom zahtevamo, da naj se beseda „originalni“ vsepovsod spremeni v besedo „spremenjeni“. Če bomo na vhodu imeli besedilo „To je originalni stavek.“ se bo na izhodu pojavilo besedilo „To je spremenjeni stavek.“. Če izpustimo črko g na koncu, se bo zamenjal le prvi najdeni niz. Če namesto črke g navedemo neko številko, npr. 3, se bo zamenjal le tretji najdeni niz. Besedilo v orodje sed najlažje spravimo s pomočjo preusmeritev. Globalno zamenjavo preizkusimo takole:

```
echo "To je originalni stavek." | sed 's/originalni/spremenjeni/g'
```

Drug način je, da stavek shranimo v datoteko `Test.txt` in izvedemo naslednji ukaz:

```
sed 's/originalni/spremenjeni/g' Test.txt
```

V tem primeri se bo tekstovna datoteka brala vrstico po vrstico, torej bo sed naenkrat dobil eno vrstico datoteke. Vsaka vrstica bo torej obdelana posebej, saj prejšnje vrstice sed več ne more spreminjati, ker jo je že oddal, naslednje vrstice pa še ni dobil. Ta lastnost predstavlja osnovno omejitev obdelave tekstovnih datotek z orodjem sed. Če namreč iščemo lastnosti, ki segajo čez več vrstic (npr. odstrani vse komentarje v programu v C-ju), potem običajno uporaba orodja sed ni smiselna oz. morda z njim sploh ne moremo rešiti naloge. V takih primerih uporabimo awk, perl ali kakšen drug skriptni jezik.

Če je ukaz za sed zelo dolg in ga nočemo vedno znova pisati, ga lahko posname v datoteko Ukaz.sed (samo tisti del, ki je bil prej napisan v apostrofih, vendar brez apostrofov!) in potem uporabimo:

```
sed -f Ukaz.sed Test.txt
```

Tukaj je še en primer:

```
echo "Doma imam štiri mačke in tri pse." | sed -e 's/pse/mačke/g' -e 's/mačke/slone/g'
```

Na izhodu bomo dobili: „Doma imam štiri slone in tri slone.“ V tem primeru smo orodju sed podali dva ukaza in zahtevali, da naj sed nad vrstico najprej v celoti izvrši prvi ukaz, nato pa nad dobljenim rezultatom še drugega. Če imamo več kot en ukaz, moramo pred vsak ukaz napisati kretnico `-e`.

Če v ukazu za zamenjavo izpustimo drugi niz znakov, se bodo podani nizi znakov v besedilu zbrisali. Na primer:

```
echo "Doma imam štiri mačke in tri pse." | sed -e 's/štiri//g'
```

Na izhodu dobimo „Doma imam mačke in tri pse.“ Med besedama imam in mačke sta dva presledka! Čeprav smo podali le en ukaz ni nič narobe, če uporabimo kretnico `-e`.

Če želimo, da se pri iskanju besed ne razlikuje med malimi in velikimi črkami uporabimo „`Ig`“ namesto „`g`“ na koncu ukaza.

Prava moč orodja sed se pokaže, ko začnemo uporabljati regularne izraze. Naslednji ukaz vse številke v besedilu nadomesti z zvezdico.

```
sed 's/[[:digit:]]/*/g'
```

Če ukaz preoblikujemo tako, da se bo glasil ‚`sed '/[[:digit:]]/d'` se zbrišejo vse tiste vrstice, ki vsebujejo niz znakov, ki ustreza podanemu regularnemu izrazu. V danem primeru bi se zbrisale vse vrstice, ki vsebujejo kakšno številko. V zvezi z brisanjem vrstic je uporaben naslednji ukaz, ki zbriše vse prazne vrstice:

```
sed '/^$/d'
```

Tukaj je še nekaj preprostih uporabnih ukazov.

Izpiši tiste vrstice, v katerih se nahaja niz znakov, ki ustreza danemu izrazu:
```
sed -n "/ izraz /p"
```

Zbriši tiste vrstice, v katerih se nahaja niz znakov, ki ustreza danemu izrazu:
```
sed "/ izraz /d"
```

Prvi niz znakov v vrstici, ki ustreza danemu izrazu, daj v okrogli oklepaj:
```
sed "s/ izraz /(&)/g"
```

Vse nize znakov, ki ustrezajo danemu izrazu, daj v okrogli oklepaj:
```
sed "s/ izraz /(&)/g"
```

Prvi niz znakov v vrstici, ki ustreza danemu izrazu, zamenjaj z besedo „NASEL“:
```
sed "s/ izraz /NASEL/"
```

Drugi niz znakov v vrstici, ki ustreza danemu izrazu, zamenjaj z besedo „NASEL“:
```
sed "s/ izraz /NASEL/2"
```

Vse nize znakov, ki ustrezajo danemu izrazu, zamenjaj z besedo „NASEL“:
```
sed "s/ izraz /NASEL/g"
```

Vse nize znakov, ki ustrezajo danemu izrazu, zamenjaj z znakom za novo vrstico:
```
sed "s/ izraz /\n/g"
```