Regulæruttrykk

Regulæruttrykk – for søking i tekst

- Aka. regex(p) / regular expressions
- Beskriver et *søkemønster* :
  - Kan angi teksten vi søker etter *eksplisitt* : `Jan`
  - Kan bruke *metategn** til å angi *variasjon*: `[Jj]an`
- Flere varianter av regulæruttrykk, bl.a.:
  - BRE: Basic Regular Expressions
  - ERE: **Extended Regular Expressions**
  - PCRE: Perl Compatible Regular Expressions

*: Merk at noen av metategnene i regex er de *samme* tegnene som brukes i "file name generation" i Bash

Vanlige anvendelser av regulærutrykk

- Spam- / junkmail-filtre
- Google og andre søkemotorer
- Databaseverktøy
- Syntaks-sjekking i kompilatorer og interpretere
- Tolkere av naturlig språk
- Emacs og andre avanserte editorer
- Mange programmeringsspråk har innebygd regex-støtte
- Linux-verktøy som f.eks. **grep**, **sed** og **awk**

Eksempel: Spam-filtrering

- Vil finne e-poster som inneholder reklame for "Viagra"
- Spammere feilstaver *bevisst* slike ord som trigger spam-filtre
- Spammere utnytter også at den menneskelige hjernen er assosiativ – når vi leser en e-post forstår vi både feilstavinger og ord som ligner

Eksempel: Spam-filtrering med regex

- Noen «kreative» omskrivinger:

```
v.i.a.g.r.a     v1aGra     vi-ag-ra
vi@gr@          ViA6ra     ViagR
```

- Regulæruttrykk kan brukes til å spesifisere alle slike kjente «feilstavinger» kompakt og presist

- En regex som dekker alle variantene ovenfor kan f.eks. se slik ut:

```
[Vv][.-]?[iI1][.-]?[aA@][.-]?[gG6][.-]?
[rR][.-]?[aA@]?
```

Match mellom regex og tekst

- Et regex prosesseres av en *regex-maskin*:
  - Programvare som sammenligner det angitte tegnmønsteret med en gitt tekst
- Oftest en og en *linje* om gangen som sjekkes
- Vi får en *match* på en tekstlinje hvis linjen *inneholder* tegnmønsteret som regulær-uttrykket beskriver

Noen metategn i regulæruttrykk

`*`	Null eller flere forekomster av forrige tegn
`+`	Én eller flere forekomster av forrige tegn
`?`	Null eller én forekomst av forrige tegn
`.`	Matcher alle tegn unntatt `NEWLINE`
`[ ]`	*Ett* av tegnene angitt mellom `[` og `]`
`[ ^ ]`	*Ett* av tegnene *ikke* angitt mellom `[ ^` og `]`
`\`	Ikke behandle neste tegn som metategn

Kontroll av repetisjon: `*` `+` `?`

- Angir *antall ganger* tegnet *foran* metategn kan forekomme:

`*`	Null eller flere forekomster
`+`	Én eller flere forekomster
`?`	Null eller én forekomst

- Merk:
  - En regex med `*` eller `+` vil matche et *uendelig* antall strenger

Eksempler: `*` `+` `?`

`0*1` Matcher null eller flere `0` etterfulgt av `1`:

`000001` `0001` `01` `1`

`0+1` Én eller flere `0` etterfulgt av `1`:

`000001` `0001` `01` (men ikke bare `1`)

`0?1` Null eller én `0` etterfulgt av `1`

`01` `1` (ingen andre matcher)

`.` (punktum): Wildcard/jokernotasjon

- Metategnet `.` matcher et hvilket som helst *enkelt* tegn, med unntak av tegnet `NEWLINE`
- Eksempel: `b.t`
  - Matcher en `b`, etterfulgt av ett vilkårlig tegn, etterfulgt av en `t`
  - Matcher f.eks.: `bat`, `bet`, `bit`, `bot`, `but`, `byt`, `b2t`, `b t` (`b`, `space`, `t`) `og bbt`
  - Matcher ikke f.eks. `bt og beat`

`* + ?` kan *endre* betydningen av `.`

`b.*t` Matcher en `b` fulgt av *0 eller flere* forekomster av *hva som helst*, etterfulgt av en `t`:

`bat`, `bet`, `bit`, `bot`, `bt`, `beat`, `bbbbbt`, `b2317612783 6t`, `b!# ¤%&/t`

`b.+t` Matcher det samme som `b.*t` , med unntak av at det *ikke* matcher `bt`

`b.?t` Som `b.t` , men vil også matche `bt`

Hvor i en tekstlinje matcher en regex?

- Regex: **`b?.t+`**

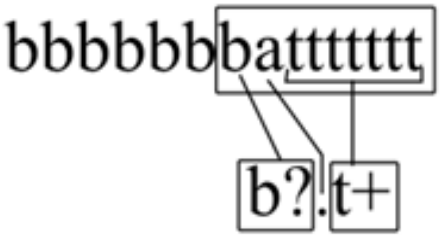
  0 eller 1 `b`, ett vilkårlig tegn og minst én `t`

- Linje: **`bbbbbbbatttttttt`**

  7 `b`, deretter 1 `a` og 7 `t`

- Er det match mellom regex og linje?

- Svaret er ’ja’, fordi:
  - Linjen/strengen inneholder en *substreng* som matcher
  - Prøver oftest å matche *alle* substrenger på en tekstlinje



Metategn for å kontrollere *hvor* i teksten en regexp matcher

^	Begynnelse på en linje
$	Slutten på en linje
\<	Begynnelse på et ord
\>	Slutten på et ord

Eksempler

regex	**matcher**
`^Jan`	Jan er en hyggelig fyr
`Jan$`	Linuxnerden heter Jan
`\<know`	I have knowledge about this
`known\>`	This is unknown to me

Matche hele linjer og hele ord

regex	**matcher**
`^Jan er grei$`	Jan er grei
`^$`	tom linje (bare `NEWLINE`)
`\<unknown\>`	This is unknown to me

Hva matcher denne?:

`^ab*c+$`

`[ ]` : Angi *eksakt* hvilke tegn som kan matche

- `.` (punktum) matcher *ett* vilkårlig tegn (hva som helst)
- `[`*`list`*`]` matcher også bare *ett* tegn, men *kun* ett av tegnene som er angitt eksplisitt i listen
- Eksempel:

  `J[abc]n` matcher `Jan Jbn Jcn`

Angivelse av tegnlister inne i `[ ]`

Kan angis på tre ulike måter:

`[`*`list`*`]`	F.eks. `[abcde]` eller `[0123456789]`
`[`*`range`*`]`	F.eks. `[a-e]` eller `[0-9]`
`[[:`*`class`*`:]]`	Bruker en tegnklasse, som definert i POSIX-standarden for klassifisering av tegn, f.eks. `[[:alpha:]]` (bokstaver)

POSIX-tegnklasser som kan brukes i regex

Class	Meaning
:alnum:	Any letter or digit
:alpha:	Any letter
:blank:	Space and tab
:cntrl:	Any control character
:digit:	Any digit
:graph:	Any non-whitespace
:lower:	Lower case letter
:print:	Any printable character
:punct:	Any punctuation mark including [and]
:space:	Any whitespace (space, tab, new line, form feed, carriage return)
:upper:	Any upper case letter
:xdigit:	Any digit or hexadecimal digit

Eksempler: Regex med tegnlister

`H[ABC][0-9]`	HA0 HB1 HC7 HA4 ...
`H[a-zæøå]iberg`	Høiberg Haiberg Hxiberg Hæiberg ...
`H[a-z]*`	H Haaaa Habcdefg Hzzzzaaa ...
`[0-9]+`	Matcher alle sekvenser med bare siffer
`^[0-9]+$`	Matcher alle *hele linjer* med bare siffer
`^[[:digit:]]*$`	Som forrige, matcher også tom streng
`[abc][abc][abc]`	aaa abc cba cbc bbc cab ccc ...

Eksempel: Fornavn Etternavn

```
^[[:upper:]][[:lower:]]+ [[:upper:]][[:lower:]]+$
```

- Matcher hele linjer med bare to ord
- Ordene må være skilt med én enkel `SPACE`
- Stor bokstav først i hvert ord, resten små bokstaver:
  - Robbie Robertson
  - Levon Helm
  - Cashmere Cat

En annen bruk av metategnet `^`

- Tegnet `^` (’caret’ / ’hat’) kan brukes *inne i* `[]` for å angi tegn som *ikke* skal forekomme
- Eksempel:

  `J[^abc]n` Jdn J3n Jzn JAn J n J@n ...

- Bruken av `^` for å utelukke tegn er ofte litt tricky*

*: Kommandoen `grep` tilbyr en enklere mekanisme for *ikke* å matche tegn/strenger

Eksempler: ^ utenfor og inne i [ ]

`^[^0-9]`

Match any line that does not start with a digit

`^[^0-9]+$`

Match any line that contains no digits

`^[^0-9].*[^0-9]$`

Match any line that does not start or end with a digit (but may have digits in between)

`^[^0-9].*[0-9].*[^0-9]$`

Match any line that contains at least one digit but not at the beginning or end

Matching av metategn

- Hvordan matche f.eks. et punktum?

  `b.t` – vil matche `b.t` men også `bat b2t bXT` ...

- For å matche et punktum *eksakt*, er det to muligheter:

  `\.` "Backslash escape": Neste tegn behandles *ikke* som metategn

  `[.]` De fleste metategn "virker ikke" inne i `[ ]` *

- Eksempel: Hva matcher `J\\\*n` ?

* Noen tegn må "escapes" med backslash også inne i `[ ]`, f.eks. `[\[\]]` matcher enten `[` eller `]`

Eksempel med metategn: Ligninger

- Enkel ligning med tre tall: `2 + 12 = 14` , `3 * 3 = 9`

  Generell form: `Heltall operator Heltall = Heltall`

  Antar at det er en enkel `SPACE` mellom tall og tegn i ligningen

  Operatorene som kan brukes er: `*` `/` `+` `-` `%` (mod)

- Merk at:

  `*` `+` `-` er metategn i regulæruttrykk

  `-` (minustegn) er metategn også inne i `[]` (angir range)

- To løsninger som begge virker:

```
[0-9]+ [*/+\-%] [0-9]+ = [0-9]+
[0-9]+ [*/+%-] [0-9]+ = [0-9]+
```

Kontroll på antall repeterte tegn

- Metategnene `*` og `+` angir et vilkårlig antall gjentagelser av foregående tegn
- Metategn for å kontrollere antall gjentagelser:

`{n}`	Nøyaktig `n` forekomster av forrige tegn
`{n,m}`	Minst `n` og høyst `m` forekomster av forrige
`{n,}`	`n` eller flere forekomster av forrige tegn

- `n` og `m` må være ikke-negative heltall og $n \leq m$

Eksempler: Antall repetisjoner

- `ab{2}` Matcher `abb`
- `ab{3,}` Matcher `abbb abbbb abbbbbb` ...
- Norske registreringsnumre for biler:

  `[A-Z]{2}[1-9][0-9]{4}`

- Norsk 11-sifret bankkontonummer:

  `[0-9]{4}\.[0-9]{2}\.[0-9]{5}`

Angivelse av tegn*sekvenser* som metategn skal virke på

- Metategnene for repetisjon virker på enkle tegn
- Vi kan få dem til å virke på tegnsekvenser, f.eks. hele ord, ved å bruke metategnene `(` `)` – venstre og høyre parantes – til å *gruppere* tegn sammen
- `(` `)` kan også brukes til å *neste* flere regulæruttrykk inne i hverandre

Eksempler: Bruk av `( )`

- `(Jan){2,3}` matcher `JanJan` og `JanJanJan`
- Linjer med minst 6 "ord", skilt fra hverandre med `SPACE`:

```
^([A-Za-z][a-z]* ){5,}[A-Za-z][a-z]*$
```

- Amerikanske telefonnummer – (123) 456-7890 :

```
(\([0-9]{3}\) )?[0-9]{3}-[0-9]{4}
```

- Noen nestede regulæruttrykk (repetisjon inne i repetisjon):

```
(A+ ){4,}
((ABC){2})+
((AB){2}C){2}
```

Valg mellom flere regulærutrykk

- Regulæruttrykk tilbyr en enkel mekanisme for å kunne matche en streng med *flere* ulike regulæruttrykk samtidig
- Angis med metategnet `|` (vertikal strek, "or")
- Kan kombineres med `()` for å sikre riktig rekkefølge i evaluering av regularuttrykkene
- Kan også bruke `()` til å neste flere "or" inne i hverandre

Eksempler på bruk av |

- Matche `AB` eller `CD` eller `EF`:
  - Virker ikke (hvorfor?): `[ACE][BDF]`
  - Virker: `AB|CD|EF`
- Matche begge stavemåtene av "grå" på engelsk:
  - `gr(a|e)y`
- Amerikanske postnumre (ZIP-codes):
  - Enten 5 siffer (12345) eller 9 siffer (12345-6789)
  - `([0-9]{5})|([0-9]{5}-[0-9]{4})`


A-1


E-6813
N


Z-48


E-443


UN 27231


24-04-75

Eksempel: Bilskilter i Norge

- Norske sivile registreringsnumre for bil etter 1950:

  AA12345

  A-1 A-12 A-1234 A-12345 A-123456

  12-34-56

- Regulæruttrykk som matcher alle skilt-typene:

```
[A-Z]{2}[1-9][0-9]{4}|[A-Z]-[1-9][0-9]{0,5}|
([0-9]{2}-){2}[0-9]{2}
```