Linux-programmer som bruker regulæruttrykk:

`grep`, `sed`, `awk`

`grep`* – søking i tekstlige data

`grep [`*`OPTIONS`*`] `*`REGEX`*` [`*`FILE`*`...]`

- Verktøy som finnes i alle Unix-lignende OS
- Leser en eller flere tekstfiler, eller standard input hvis ingen filer er angitt, linje for linje
- Hver linje sammenlignes med et søkemønster som angis som et *regulæruttrykk*
- Hvis en linje *matcher* skrives den ut til standard output
- Linjer som *ikke* matcher, skrives ikke ut

*: Navnet kommer fra tidlig Unix-historie

Bruk av `grep`

- Brukes typisk til å «redusere en stor mengde informasjon til en liten mengde nyttig informasjon»
- Ofte som del av «pipelines» på kommandolinjen
- Bruk av `grep` er så vanlig at det har blitt til et eget verb* i det engelske (og norske?) språket – "to grep"

*: På samme måte som "å google"

Varianter av `grep`

- `grep`
  - Basic Regular Expressions
- `egrep` *
  - Extended Regular Expressions (`grep -E`)
- `fgrep`
  - Fixed, bare konstante mønstre (`grep -F`)
- `rgrep`
  - Recursive, søker i et helt katalogtre (`grep -r`)

*: Vanligst(?) å bruke

«Quoting» av regulæruttrykk i `grep`

- Skriver ofte regulæruttrykk i `''` (single quotes)
- Unngår da at shellet *selv* håndterer metategn som er del av regulæruttrykket:
  - F.eks. kan et regulæruttrykk med `*` få shellet til å gjøre «file name expansion» og erstatte `*` med en liste av filer
- Kan i stedet bruke `""` (double quotes) hvis vi f.eks. trenger å hente ut verdier fra shellvariabler med `$` inne i regulæruttrykket

`egrep` – eksempler (1)

- Søk i alle tekstfiler etter "beløp i amerikanske dollar":

```
egrep '\$[0-9]+\.[0-9]{2}' *.txt
```

- Linjer som starter med ordene "Vi", "We" eller "Wir":

```
egrep '^Vi |^We |^Wir ' *.txt
```

- Alle "gamle" Linux / Unix-kommandoer med bare to tegn:

```
ls /bin | egrep '^[a-z][a-z]$'
```

`egrep` – eksempler (2)

- Linjer med tekst inne i firkant-paranteser (brackets):

```
cat brackets.txt | egrep '\[[A-Za-z]*\]'
```

- Alle som er logget på systemet, unntatt meg selv:

```
who | grep -v "^$USER" | cut -f1 -d' '
```

Noen nyttige opsjoner til `egrep`

`-w`	Let bare etter hele *ord*
`-c`	Skriv bare ut *antall linjer* i input som matcher
`-o`	Skriv bare ut de *delene* av en linjene som matcher
`-i`	Ikke forskjell på store og små bokstaver i regex
`-v`	Bare linjer som *ikke* matcher (ofte bedre enn `[^]`)
`-l`	Skriv bare ut *navn på file(r)* med minst én match
`-L`	Skriv bare ut navn på file(r) som *ikke* matcher
`-n`	Skriv også ut linjenummere i filen(e)
`-m`	Stopp søk i en fil etter et gitt antall (typisk 1) matcher
`-q`	Ingen output, stopp (med *exit-kode* 0) hvis match

Søking i ordlisten i /usr/share/dict/words (1)

- Ord med 20 eller flere bokstaver:

```
egrep '[[:alpha:]]{20,}'
```

- Antall ord med 15 eller flere bokstaver:

```
egrep -c '[[:alpha:]]{15,}'
```

- Ord på tre bokstaver med bare vokaler:

```
egrep -i '^[aeiouy]{3}$'
```

- Ord på tre bokstaver med bare konsonanter:

```
egrep '^[[:alpha:]]{3}$' | egrep -iv '[aeiouy]'
```

Søking i ordlisten i /usr/share/dict/words (2)

- Ord som begynner med `qu` og slutter med `y`:

```
egrep -i '^qu[a-z]*y$'
```

- Ord som inneholder alle vokalene i alfabetisk rekkefølge:

```
egrep -i 'a[a-z]*e[a-z]*i[a-z]*o[a-z]*u[a-z]*y'
```

`sed` – stream editor

- `sed` utfører redigering av en *tekststrøm*
- Redigeringen angis med enkle kommandoer
- Argumenter til kommandoene er ofte regulæruttrykk
- `sed` leser tekst linje for linje, ett tegn om gangen
- Teksten leses fra standard input eller fra en angitt fil
- Etter at en linje er lest inn og redigert, skriver `sed` den redigerte linjen ut til standard output

Hva brukes `sed` til?

- Enkle oppdateringer av mange tekstfiler:
  - Effektivt, sparer manuell editering
  - Lett å automatisere med shellprogrammer
  - Raskt, `sed` leser gjennom en tekstfil bare *en* gang
  - Men også begrenset funksjonalitet
- Alternativer til `sed` :
  - For enklere jobber: `grep`, `head`, `tail`, `tr`
  - For mer kompliserte jobber: `awk`, `perl`

Enkel bruk av `sed`*

- Bytte strengen `Jan` med `Harald` i standard input:

```
sed 's/Jan/Harald/'
```

- Flere kommandoer kan angis på samme linje:

```
sed -e 's/Christian/en raring/' -e 's/Robert/en
raring til/' innfil > utfil
```

- I eksemplet over leser og skriver `sed` tekst fra/til filer

- Kommandoer kan også leses fra fil – et *sedscript*:

```
sed -f kommandofil < inputfil > outputfil
```

*: Vi skal begrense oss til bare å se på redigeringskommandoen `s` – "search and replace"

`sed` og regulærutrykk

- Syntaks for kommandoen `s`:

  ```
  sed 's/pattern/replacement/'
  ```

- `pattern` kan angis som et regulæruttrykk
- *Første* tekststreng som matcher `pattern` på hver linje, vil bli byttet ut med `replacement`
- For å bytte *alle* forekomster av `pattern` på linjen:

  ```
  sed 's/pattern/replacement/g'
  ```

- Opsjonen `-r` for Extended Regular Expressions

Eksempel (fra læreboken)

- En fil inneholder linjer med adresser, der en adresse kan innholde en angivelse av leilighet, som:

```
Apt 5          apt 5        Number 5
number 5       #5
```

- Ønsker å bytte ut alle disse variantene med:

```
Apartment 5
```

- Løsning:

```
sed -r -e 's/[Aa]pt|[Nn]umber/Apartment/'
-e 's/#/Apartment /' innfil > utfil
```

Sletting av tekststrenger

- Kan gjøres enkelt ved å ved å angi en tom "replacement string" i kommandoen `s`
- Eksempel: Fjerne alle (norske) kontonummere:

```
1111.22.33333
```

- Kommando:

```
sed -r 's/[0-9]{4}\.[0-9]{2}\.[0-9]{5}//'
```

Redigering av strengen som matcher

- `sed` tilbyr også mulighet for å *redigere* strengen som er funnet som match til *pattern*, i stedet for bare å bytte den ut med en fast *replacement*
- Siste streng som er funnet som match ligger alltid lagret i en *placeholder*
- Vi kan referere til innholdet i placeholder med tegnet `&`
- Eksempel: Sette matchende tekststreng i anførselstegn:

```
sed -r 's/ekspert(er|en|ene)?/"&"/'
```

Innebygd editering av placeholder *

- Funksjoner for å endre teksten i placeholder:

  `\u` upper case the first letter of the string

  `\l` lower case the first letter of the string

  `\U` upper case the entire string

  `\L` lower case the entire string

- Gjør om alle ord til små bokstaver:

```
sed -r 's/[A-Za-z]+/\L&/g'
```

- Gjør om første tegn i alle ord til stor bokstav:

```
sed -r 's/\<[a-z]/\u&/g'
```

*: Placeholder kan også deles i «items» som redigeres separat, se lærebokens avsnitt 6.5.3

`awk` – et språk for å håndtere tekstdata

- Interpretert programmeringsspråk laget for tekstbehandling
- Brukes typisk for å ekstrahere data og lage rapporter
- Er standard verktøy på Linux- og Unix-systemer
- Virkemåte:
  - Leser linjer med tekst, oppdelt i *felt*
  - Utfører operasjoner for linjer som har *matchende* felt
  - Matching spesifiseres med *regulærutrykk*
  - Operasjonene som skal utføres kan programmeres med bl.a. if-tester, løkker og kall på funksjoner
- Se `man awk` og lærebokens avsnitt 6.6 for mer info.