UNIX/Linux
Filosofi og Syntaks

Flosofi omkring udvikling

Linux filosofi

- **Lav små programmer**, der kan kombineres
- Lav hvert program, så det er **eminent til én ting**
- Lav en **prototype** så hurtigt som muligt
- Prioritér **flytbarhed over effektivitet**
- Gem data i **flade ASCII-filer**
- **Genbrug kode** fremfor at skrive koden fra grunden
- **Brug shellscripts** af hensyn til effektivitet og flytbarhed

Linux/Unix kultur

Linux/UNIX kultur

- Giv brugere mulighed for at tilpasse deres arbejdsmiljø
- Hold UNIX/Linux kernen lille
- Brug små bogstaver og hold tingene korte
- Spar på træerne (data på papir er døde data)
- Tavshed er guld (Ingen unødig kommunikation)
- Tænk parallelt
- Summen af de enkelte dele er større end helheden

Linux kommandoer

- Meget af dit arbejde med Linux består af at indtaste kommandoer.
- Der findes flere hundreder forskellige Linux-kommandoer. Du kan ikke lære dem allesammen at kende.
- Linux-kommandoer skrives på *kommandolinjen*.
- Linux-kommandoer skal bruges korrekt for at give det ønskede resultat.
- Selvom kommandoer ser forskellige ud og gør forskellige ting, kan man alligevel godt angive en tommelfingerregel for, hvordan syntaksen ser ud:

  Eksempel:

  Kommandonavn [Evt. argumenter]

Eksempler på kommandoer

```
$ ls -l          #Kommando med et "flag"
$ cat fil3       #Kommando med er argument
```

Flere kommandoer på samme linje

```
$ ls ; cat fil3
```

En tekst der fylder mere end en linje

```
$ echo "Her er en tekstlinje,
> der fylder mere end 1 linje på skærmen"
```

Eksempler på kommandoer

En kommando kan skrives på flere linjer \

```
$ ls -l | sort -n \
> -k5
```

Lav “luft” i kommandoerne for at gøre linjen læsevenlig:

```
$ echo Her    er lidt        tekst
Her er lidt tekst

$ ls -l   |    sort -n -k5   |    more
```

Kommandonavnet

- Komandonavnet er typisk et kort ord på mellem 2 - 5 bogstaver
- For eks. ls , mv eller cp

Prøv at skrive følgende simple kommandoer:

```
bruger@superusers:~$ date
bruger@superusers:~$ who
bruger@superusers:~$ cal
bruger@superusers:~$ ls
bruger@superusers:~$ pwd
```

Argumenter

"flag" og *"parametre"*

Kommandonavn [evt. flag] [parametre]

Et ***flag*** er en ét-bogstavskode, der giver adgang til en speciel variant af kommandoen. Flag indledes i de allerfleste tilfælde med et minus (-).

Man kan ofte angive flere flag efter hinanden, så de deles om minustegnet - eksempel:

```
$ ls -li
$ ls -l -i
```

Argumenter

”flag” og *”parametre”*

Der findes hjælp at hente omkring kommandoernes flag i ”man” og –help

Prøv for eksempel at skrive:

```
$ man ls
eller
$ ls --help
```

Eksempler på brug af 'ls' kommandoen

```
$ ls -l fil3 #(Udskriver lang oplysningslinje om fil3)

$ ls -l a b #(Udskriver lange oplysningslinjer om
            de to filer a og b)

$ ls -l   #(Når der ikke angives nogen parametre,
          tages udgangspunkt i arbejdskataloget!
          Altså udskrives lange oplysningslinjer om
          filerne i arbejdskataloget - dog ikke
          filer, hvis navn begynder med punktum)

$ ls -l /usr #(Udskriver lange oplysningslinjer om
             filerne i kataloget /usr -- dog ikke
             filer, hvis navn begynder med punktum)
```

Eksempler på brug af 'cat' kommandoen:

```
$ cat brev   #(Udskriver lndholdet af filen brev)

$ cat        #(Udskriver linjer fra tastaturet)
```

- *Lav øvelser 3.1, 3. 2 og 3.3*

Interne og eksterne kommandoer

- Når ***vi skriver en kommando***, ønsker vi, at ***shellen udfører kommandoen***
- Mange ***kommandoer er indbyggede i shellen***
- ***Shellen*** kan uden videre selv udføre arbejdet.
- ***Shellen*** sørger også for at ***eksterne kommandoer*** kan udføres.
- Der er ***principielt ingen forskel i syntaks*** på ***interne*** og ***eksterne kommandoer*** – udover de forskelle, der er kommandoerne imellem.

PATH og udførelse af ekstern kommando

- ***Shellen leder i bestemte kataloger*** i en bestemt rækkefølge, når den skal udføre en ***ekstern kommando***
- Dette er defineret i PATH variablen
- Indholdet her kunne f.eks. være:
  `PATH=/usr/bin:/usr/ucb:/home/bruger1/bin`

  Hvilket gør at shellen benytter rækkefølgen:

  Kig *først* i `/usr/bin -kataloget`
  Kig *dernæst* i `/usr/ucb -kataloget`
  Kig *dernæst* i `/home/bruger1/bin -kataloget`
- Så snart shellen har fundet filen med kommandoen holder den op med at lede.
  Derefter bliver kommandoen læst ind og udført.

3 nyttige kommandoer om kommandoer

- Sådan kan du finde ud af hvad en given kommando er for noget:

```
$ whereis ls                    #(Hvor findes ls kommandoen)
```

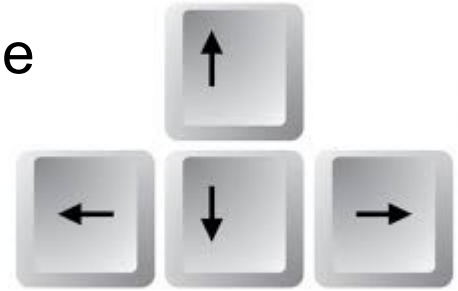
```
$ whatis ls                     #(Hvad er ls)
```

```
$ type ls                       #(Fortæller noget typen)
```

Genbrug af tidligere indtastede kommandoer

- Sådan kan du navigere tilbage i tidligere indtastede kommandoer i **Bash** (Mere om bash senere): `$ ls -l`

PIL OP fører dig tilbage



Brug venstre og højre piletast hvis du vil redigere i tidligere indtastet kommando

Udfør kommandoen ved at trykke retur

Visning af output på skærm

- Hvis ***output*** fra en kommando ***fylder mere*** end ***vinduets højde***, ruller det blot ud foroven.
- Man kan bremse outputtets flugt over skærmen med:
  | more (primært UNIX)
  | less (primært Linux)
- **Retur –** bladrer 1 linje frem
- **f -** bladrer en skærmvisning **f**rem
- **b -** bladrer en skærmvisning **b**aglæns
- **q -** afslutter

```
$ ls -l | more
$ ps -ef | less
```

Lav øvelse 3.4