# Modul 4

Sortera och filtrera

Sebastian Bengtegård, Johan Holmberg

22 Januari 2025

## Innehållsförteckning

# 1 Förord

I denna modul kommer vi att introduceras till sex nya kommandon:

- `head` (*output the first part of files*)
- `tail` (*output the last part of files*)
- `sort` (*sort lines of text files*)
- `nl` (*number lines of files*)
- `wc` (*print newline, word, and byte counts for each file*)
- `uniq` (*report or omit repeated lines*)

Genom dessa nya kommandon kommer vi att kunna sortera och filtrera på innehållet av textfiler. På egen hand utför dessa kommandon relativt enkla uppgifter. Men, genom att kombinera dessa via *pipes*, så kommer vi att få se den verkliga nyttan av att ha små enkla kommandon som tillsammans kan utföra komplexa uppgifter.

Modulen avslutas med en övning, men eftersom att denna modul handlar om att sortera och filtrera på innehållet av textfiler, så kan du redan nu ladda ner övningsmappen, för att ha något att experimentera med.

# 2 `head`: Tjuvkika på en fils innehåll

Kommandot `head` används för att skriva ut de 10 **första** raderna av en eller flera filer till *standard output*.

```
sebastian@mindator:~$ head file...
```

*Kom ihåg att notationen ... innebär att vi kan skriva ett eller flera filnamn efter varandra, separerade med ett mellanslag.*

För att ändra antalet rader som skrivs ut, antingen fler eller färre än 10, använder vi tillvalet `-n` (*lines*).

Skulle vi vilja skriva ut de första 5 raderna från filen `names.txt` hade vi skrivit så här:

```
sebastian@mindator:~$ head -n 5 names.txt
Julia
Adrian
Lisa
Pelle
Kerstin
```

## 3 `tail`: Tjuvkika på en fils innehåll, bakifrån

Kommandot `tail` används på samma vis som `head`, men det skriver istället ut de 10 **sista** raderna av en eller flera filer till *standard output*.

```
sebastian@mindator:~$ tail file...
```

Precis som med kommandot `head` kan vi styra antalet rader vi vill skriva ut via tillvalet `-n` (*lines*).

```
sebastian@mindator:~$ tail -n 5 names.txt
Erik
Johan
Elisabeth
Agnes
Stina
```

### 3.1 Tillvalet: follow

Det finns ett tillval som gör kommandot `tail` extra nyttigt, vilket är `-f`, kort för *follow*.

Detta tillval innebär att kommandot `tail` kommer att fortsätta skriva ut alla nya rader som läggs till i en fil. På grund av detta måste vi därför själva välja när vi vill avsluta kommandot. Detta gör vi genom kortkommandot `^C` (`CTRL-c`), som dessutom brukar fungera på de flesta kommandon och program där vi manuellt behöver avsluta det.

Om du vill se hur detta fungerar behöver du först skapa en textfil med hjälp av en textredigare, så som `nano` eller någon annan. Fyll sedan filen med en rad text, till exempel "Hello!", och spara den som `demo.txt`.

Därefter startar du kommandot `tail` med tillvalet `-f` på den textfil som du precis skapade.

```
sebastian@mindator:~$ tail -f demo.txt
Hello!
```

Om du nu, i din textredigerare (t.ex. nano eller notepad), lägger till raden "Goodbye." och sparar filen, så kommer kommandot `tail` att skriva ut den nya raden direkt i terminalen.

```
sebastian@mindator:~$ tail -f demo.txt
Hello!
Goodbye.
```

Ett vanligt användningsområde för detta är att följa (*follow*) utskriften till så kallade

"loggfiler". Dessa filer kan ses som en slags loggbok där ett program skriver ner saker och ting som sker, till exempel vilka som besöker en hemsida, vilket vi då hade kunnat se i realtid genom att följa utskriften till hemsidans loggfil.

## 4 `sort`: Sortera filinnehåll

Kommandot `sort` används för att sortera innehållet från en eller flera filer, för att sedan skriva ut resultatet till *standard output* (det vill säga att filen i sig kommer inte att redigeras).

```
sebastian@mindator:~$ sort file...
```

Till exempel, låt oss säga att vi har filen `names.txt` med 5 stycken namn.

```
sebastian@mindator:~$ cat names.txt
Lisa
Johan
Sebastian
Anita
Kerstin
```

För att sortera dessa använder vi oss av `sort`.

```
sebastian@mindator:~$ sort names.txt
Anita
Johan
Kerstin
Lisa
Sebastian
```

Skulle vi vilja spara detta i en separat fil, till exempel `sorted_names.txt`, använder vi tillvalet `-o` (*output*) följt av ett filnamn.

```
sebastian@mindator:~$ sort -o sorted_names.txt names.txt
```

Vi kan göra samma sak med en fil som innehåller siffror.

```
sebastian@mindator:~$ cat numbers.txt
10
1
120
2
```

```
30
29
sebastian@mindator:~$ sort numbers.txt
1
10
120
2
29
30
```

Ta en extra titt på de "sorterade" siffrorna, det är något som inte riktigt stämmer...

Kommandot `sort` kommer att sortera siffror på samma vis som för bokstäver, det vill säga en bokstav i taget från vänster till höger. Om vi istället vill sortera på det faktiska siffervärdet (1, 10, 100, osv.) behöver vi lägga till tillvalet `-n` (*numeric sort*).

```
sebastian@mindator:~$ sort -n numbers.txt
1
2
10
29
30
120
```

Skulle vi vilja ha utskriften i omvänd ordning lägger vi till tillvalet `-r` (*reverse*).

```
sebastian@mindator:~$ sort -nr numbers.txt
120
30
29
10
2
1
```

## 5 `nl`: Visa radnummer i en fil

Kommandot `nl` (för *newlines*) används för att skriva ut en eller flera filer, med radnummer, till *standard output*.

```
sebastian@mindator:~$ nl names.txt
   1  Lisa
   2  Johan
   3  Sebastian
   4  Anita
   5  Kerstin
```

Vilket är händigt om vi snabbt vill se vad som står på en viss rad.

## 6 `wc`: Räkna ord i filer

Kommandot `wc` (för *word count*) används för att skriva ut antal radbryt, ord och *bytes* (storleken) från en eller flera filer.

```
sebastian@mindator:~$ wc file...
```

Vanligtvis vill vi endast skriva ut antalet av en sak, till exempel antal ord. Här är några exempel för att demonstrera detta:

Table 1: Några exempel på kommandot `wc`

| Kommando | Resultat |
|---|---|
| `wc -l file1` | Skriver endast ut antal radbryt från filen *file1*. |
| `wc -w file1` | Skriver endast ut antal ord från filen *file1*. |
| `wc -c file1` | Skriver endast ut antal *bytes* från filen *file1*. |

Om vi skulle vilja ta reda på hur många ord (förnamn i vårat fall) som finns i filen `names.txt` hade vi skrivit följande:

```
sebastian@mindator:~$ wc -w names.txt
5 names.txt
```

I exemplet ovan förutsätter vi att vi endast har ett namn (dvs. ett ord) per rad. I andra fall kunde tillvalet `-l` (antal radbryt) vara ett lämpligare förslag.

## 7 `uniq`: Dölj dubbletter

Kommandot `uniq` används för att dölja rader som, direkt efter varandra, upprepar sig. Det kan även visa hur många gånger dessa rader upprepade sig.

```
sebastian@mindator:~$ uniq file
```

Till exempel, låt oss säga att vi har filen `names.txt` som innehåller följande:

```
sebastian@mindator:~$ cat names.txt
Johan
Sebastian
Sebastian
Anita
Kerstin
Sebastian
```

För att dölja upprepande rader skriver vi följande:

```
sebastian@mindator:~$ uniq names.txt
Johan
Sebastian
Anita
Kerstin
Sebastian
```

Observera att den tillåter flera av samma rad, **men inte** direkt efter varandra.

Skulle vi vilja veta hur många gånger en rad upprepar sig använder vi tillvalet `-c` (*count*).

```
sebastian@mindator:~$ uniq -c names.txt
   1 Johan
   2 Sebastian
   1 Anita
   1 Kerstin
   1 Sebastian
```

Här kan vi se att "Sebastian" upprepar sig två gånger i början av filen.

### 7.1 `uniq` + `sort`

På egen hand kan kommandot `uniq` kännas en aning begränsat. Om vi däremot kombinerar det med kommandot `sort` så öppnas flera nya möjligheter upp.

Till exempel, om vi endast skulle vilja skriva ut unika namn från filen `names.txt`, så hade vi skrivit följande:

```
sebastian@mindator:~$ sort names.txt | uniq
Anita
Johan
Kerstin
Sebastian
```

Notera att i ovanstående exempel skriver vi bara ut "Sebastian" en gång.

Skulle vi vilja ta reda på hur många gånger respektive namn upprepas i samma fil, så lägger vi till tillvalet `-c` (*count*):

```
sebastian@mindator:~$ sort names.txt | uniq -c
   1 Anita
   1 Johan
   1 Kerstin
   3 Sebastian
```

## 8 Praktiska exempel

De kommandon vi nu gått igenom kan fungera bra på egen hand, men fungerar desto bättre i kombination med varandra. Nedan finner du några exempel.

Skriv ut antal unika namn i filen `names.txt`.

```
sebastian@mindator:~$ sort names.txt | uniq | wc -w
```

Skriv ut de 10 första raderna av filen `names.txt` med radnummer.

```
sebastian@mindator:~$ head names.txt | nl
```

Skriv ut antalet rader från filerna `a.txt` och `b.txt` (sammanslaget).

```
sebastian@mindator:~$ cat a.txt b.txt | wc -l
```

Bläddra igenom alla unika namn från filerna `a.txt` och `b.txt` i `less`.

```
sebastian@mindator:~$ cat a.txt b.txt | sort | uniq | less
```

Det är bara fantasin som sätter stopp för vad vi kan åstadkomma genom att kombinera kommandon.

Skriv ut de 5 mest förekommande namnen och hur många gånger respektive namn upprepas, i omvänd ordning (så att det mest förekommande namnet är överst).

```
sebastian@mindator:~$ sort names.txt | uniq -c | sort -r | head -n 5
```

Exemplet ovan är relativt avancerat, pröva med ett kommando i taget för att se vad som sker steg för steg.

# 9 Övningar

Börja med att öppna en ny terminal och navigera till den plats där du placerat övningsmappen (*exercise-module-04*). Tänk på att vissa av dessa övningar kan lösas på flera vis och genom olika kommandon, försök därför så gott det går att förhålla dig till de kommandon vi presenterat i denna modul.

- Vilket namn står på rad 7 i filen `names.txt`? Antecka ner namnet som **A1**.
- Hur många unika siffror finns det i filen `numbers.txt`? Anteckna ner antalet som **A2**.
- Hur många unika namn är det bland de sista 15 raderna i filen `names.txt`? Antecka ner antalet som **A3**.
- Hur många rader finns det sammanslaget från filerna `names_a.txt`, `names_b.txt` och `names_c.txt`? Anteckna ner antalet som **A4**.
- Hur många gånger upprepas siffran `1337` i filen `numbers.txt`? Anteckna ner antalet som **A5**.