Laborationer i kursmomentet Datoranvändning E1

http://www.etek.chalmers.se/~hallgren/Eda/

Laboration nr 6: Mer om UNIX

1996, 1997 Magnus Bondesson
1998 och 99-10-26 Thomas Hallgren

1 Introduktion

Vi stötte redan i början av läsperiod 1 på några de vanligaste UNIX-kommandona. I denna laboration presenteras några fler användbara kommandon, tillsammans med sätt att kombinera kommandon för att utföra uppgifter som det inte finns något färdigt kommando för.

Kommandon som dyker upp är: `finger`, `rwho`, `v`, `w`, `uptime`, `xload`, `which`, `echo`, `set`, `wc`, `grep`, `sort`, `cut`, `chmod`.

Begrepp som dyker upp: variabler, omdirigering, rör (pipes), kommandofiler.

1.1 Förberedelser

Gå på föreläsningen! Skumma igenom häftet och läs de avsnitt i Gula Boken som det finns hänvisningar till!

1.2 Redovisning

Skriv svar på frågorna i häftet. Spara filer som skapas under arbetet. Visa allt för handledaren för att bli godkänd.

2 Uppgifter

Som tidigare markerar symbolen ☐ i uppgifter med flera delar inledningen på ett stycke med en konkret arbetsuppgift. Dessförinnan kan finnas motiverande eller förklarande text.

2.1 Diverse användbara UNIX-kommandon

Uppgift 1. Vem använder datorerna? Finns det någon ledig dator?

☐ Pröva följande kommandon. Se manualblad eller Gula Boken för mer information.

- `finger` *användarnamn* eller *förnamn* eller *efternamn*
  Ger information om alla användare med aktuellt namn.

- `finger` *användarnamn@datornamn*.
  Ger information om en användare på en annan dator. Detta brukar fungera om den andra datorn är en Unix-dator och finger-servern inte är avstängd av säkerhetsskäl. Prova t ex `finger hallgren@butthead.cs.chalmers.se`.
- `rwho`
  Ger information om alla påloggade användare vid E-systemet.
- `w`
  Visar bland annat vilka program användare på samma dator kör för tillfället.
- `v`
  Lokalt UNIX-kommando, som ger information om lediga/upptagna datorer. `v -h` ger mer info om vad kommandot kan göra.
- `uptime`
  Visar medelvärden över hur hårt belastad datorn har varit den senaste tiden: 0.00 = datorn har inget att göra, 0.50 = datorn arbetar med något 50% av tiden, 1.00 = datorn är upptagen med något hela tiden, 2.00 = datorn har i genomsnitt två olika saker att göra och växlar mellan dem, o s v. Programmet `xload` (som normalt startas automatiskt och visas i övre högra hörnet av skärmen) visar samma sak i ett diagram.

Uppgift 2. Var finns alla kommandon?

Kommandon i UNIX är program, som finns i filer. Med kommandot `which` kan man ta reda på var ett kommando finns. Kommandot `which ls` ger svaret `/usr/bin/ls`, t ex.

☐ a. Var finns kommandot `date`? _____

☐ b. Var finns kommandot `v`? _____

☐ c. Var finns `netscape`? _____

Hur kommer det sig att det räcker att skriva `ls` för att kommandotolken ska hitta `/usr/bin/ls`? Hittar kommandotolken rätt program var det än finns? Det rätta svaret är att kommandotolken letar i de kataloger som anges av variabeln `$path`. Man kan använda kommandot `echo $path` för att ta reda på hur den är definierad.

☐ d. Hur många kataloger letar kommandotolken i? _____

Man kan lätt själv ändra vilka kataloger kommandotolken ska leta i. Antag t ex att vi vill lägga till `~hallgren/Intro/` i listan. Vi kan då skriva

```
set path = ( $path ~hallgren/Intro/ )
```

vilket betyder att det nya värdet på variabeln blir det gamla utökat med `~hallgren/Intro/`.

Anm. Om det finns två kommandon med samma namn får man det som finns i den katalog som står först i listan.

Anm. I hemkatalogen finns en dold kommandofil som heter `.cshrc`. Kommandona i den utförs varje gång en ny kommandotolk startas (t ex när man startar ett terminal-

fönster). Man kan stoppa in egna kommandon här, t ex för att lägga till egna kommandokataloger till `$path`-variablen.

Anm. `$path` är bara en av många variabler som styr hur kommandotolken fungerar. Se avsnitt 1.16.4 Gula Boken för mer information. Det finns även en uppsättning variabler som påverkar andra kommandon, t ex `$PRINTER` för val av skrivare, som vi stötte på i Laboration nr 4. Dessa beskrivs i avsnitt 1.16.5.

2.2 UNIX-kommandon, omdirigering och rörledningar

Uppgift 3. Omdirigering

Normalt hamnar resultaten av ett UNIX-kommando i det fönster i vilket kommandot givits, vilket brukar kallas "standard-ut". Men man kan också dirigera resultaten så att de i stället hamnar i en fil, med (> kan utläsas "från kommandot till")

`kommando >`*filnamn*

Läs mera om omdirigering i avsnitt 1.16.1 i Gula Boken.

☐ Se till att resultatet av `ls -l` hamnar som filen `LISTA`. Kontrollera med t ex `more` att `LISTA` innehåller det den skall.

På motsvarande sätt kan ett kommando som normalt hämtar information från tangentbordet, "standard-in", ta den från en fil. Man använder då i stället tecknet < (”till kommandot från”). Vi avstår från någon uppgift på detta.

Uppgift 4. Några användbara UNIX-kommandon: textfiler

Under en stor del av UNIX’ livstid (hela 70-talet och större delen av 80-talet) var det dominerande sättet att arbeta med datorn via text-terminaler. I de flesta UNIX-system finns det därför fortfarande en uppsjö av program som bearbetar textfiler på olika sätt. Det skulle vara ganska jobbigt att lära sig hur alla fungerar, men eftersom deras funktionalitet överlappar en hel del behöver man inte det heller. Vi kommer nedan att stöta på några av de vanligaste, som kan vara bra att känna till för husbehov.

Låt oss till att börja med titta på tre kommandon `wc`, `grep` och `sort` som alla kan användas på två olika sätt. Om man skriver ett eller flera filnamn på kommandoraden läser programmen indata från dessa filer. Annars tas indata från "standard-in", dvs det man skriver på tangentbordet på raderna efter kommandot, om man inte har gjort någon omdirigering med "<". Vid inmatning på detta sätt anges slut på indata med CTRL-D. Mer information om kommandona fås på vanligt sätt (manualblad resp sid 58).

☐ Tag med hjälp av `wc` reda på antal rader, ord och tecken i filen `Berling.txt`. Svar:

_____

☐ Kommandot `grep` används för att söka efter teckenföljder i en fil. När `grep` funnit en rad som innehåller den sökta textsträngen så skriver `grep` ut denna rad. Skriv ut de rader i filen `Berling.txt` som innehåller teckenföljden `hon`. Ta också reda på vilka som innehåller *ordet* `hon`. (Tips: läs vad `grep -w` betyder i manualbladet för `grep`.)

☐ Kommandot `sort` används för att sortera text eller tal. Sortera raderna i `Berling.txt` alfabetiskt.

Uppgift 5. Rörledningar (pipes)

I denna uppgift ska vi göra några övningar med rörledningar, beskrivs kortfattat i Gula Boken, sid 23. Genom att känna till några enkla grundkommandon och hur man kopplar samman kommandon med rör, kan man med ganska lite arbete lösa till synes ganska komplicerade uppgifter. Vi avslutar denna uppgift med att konstruera en alfabetisk lista med namnen på de som nu går i E1.

UNIX är ett fleranvändarsystem. Detta betyder bland annat att man måste logga in innan man kan börja använda datorn, att alla filer har en ägare som bestämmer vilka andra användare som ska få läsa och ändra i filen. Någonstans i systemet måste det alltså finnas information om vilka användare som har rätt att använda datorn. I UNIX finns denna information traditionellt i filen `/etc/passwd`. I filen finns för varje användare bland annat användarnamn, lösenord, hemkatalog och vald kommandotolk. En trevlig egenskap hos denna fil är att den är en vanlig textfil och att alla användare har rätt att titta i den.

☐ Tag en titt på filen `/etc/passwd` med `more` (eller `less`)!

Som kanske framgår beskriver varje rad i filen en användare. Raderna består av ett antal fält som är åtskiljda av kolon. De första två fälten innehåller användarnamnet och lösenordet[1]. Det femte fältet är användarens riktiga namn.

> **Anm.** Mer information kan fås med kommandot `man 4 passwd`, där 4 ger avsnitt 4 i manualen, som beskriver olika filformat. (Om man inte anger något avsnitt får man 1, som beskriver kommandon.)

☐ Hur många registrerade användare finns det i Etek-systemet? _____

☐ De som är inskrivna på E i år har sin hemkatalog i `/u1/e99`. Hur många är inskrivna i år? (Tips använd kommandona `grep` och `wc`, och ett rör) _____

Kommandot för att räkna ut detta blev: _____

Slutligen ska vi konstuera namnlistan och sortera den. Man kan använda kommandot `cut` för att välja ut vissa fält ur varje rad i en fil. I vårt fall blir det `cut -f 5 -d :` (eftersom vi vill välja ut det femte fältet och fälten skiljs åt av kolon. Se `man cut` för förklaring). För sorteringen antar vi lite förenklat att efternamnet är det andra namnet. Sorteringen kan då göras med `sort +1`, där `+1` betyder att `sort` ska bortse från första ordet på varje rad vid sorteringen.

☐ Sätt nu samman alltihop till ett kommando som sparar namnlistan i filen `E1-namnlista`. Hur blir kommandot?

_____

☐ Utför kommandot och spara filen!

1. Lösenorden lagras förstås inte i klartext, utan är krypterade...

Uppgift 6. En nyttig användning av rör

I Uppgift 15 i Laboration nr 6 provade vi att söka information om kommandon genom att använda `apropos`. Ett problem är att `apropos` hittar väldigt många svar om det man söker efter har ett kort namn. Vill man hitta information om C-kompilatorer får man t ex inte mycket hjälp av `apropos c` (prova!). En lösning på detta är att kombinera `apropos` med `grep`.

För att hitta information om C-kompilatorer kan det vara lämpligt att söka efter `compiler` med `apropos` använda `grep` för att välja ut de rader i svaret som innehåller ordet `C`. Hur blir kommandot för detta?

_____

Hur många rader långt blir svaret då man utför kommandot? (Tips: räkna raderna manuellt, eller kombinera kommandot med `wc`.) _____

2.3 Kommandofiler

Uppgift 7. Kommandofiler

När man startar ett terminalfönster startas normal kommandotolken `tcsh`, som läser de kommandon man matar in på tangentbordet och utför dem. En naturlig fråga är nu: kan `tcsh`, liksom kommandona vi tittade på i Uppgift 4, även läsa indata från en fil? Svaret är ja! Har man en fil som innehåller kommandon kan man få `tcsh` att utföra dem genom att skriva `tcsh` *filnamn*.

Filer som innehåller kommandon kallas inte helt oväntat för **kommandofiler** (se sidan 29 i Gula Boken). En kommandofil är alltså en textfil (tillverkas med t ex *NEdit*) bestående av ett antal kommandorader.

☐ Gör med *NEdit* en fil med innehållet

```
date
echo $1 $USER
```

Spara den under namnet `kommando1`. Provkör kommandofilen genom att mata in t ex

```
tcsh kommando1 Hejsan
```

Vad skrivs ut? _____

Förklaring: `$1` är en variabel som ersätts med det ord som skrivs efter `kommando1` och `$USER` ersätts med användarnamnet. Kommandot `echo` skriver ut en textföljd, dvs kommandofilanropet kommer att skriva ut datumuppgifter och sedan `Hejsan` *`användarnamnet`*.

Det går även att göra kommandofiler lika lätta att använda som vanliga kommandon (dvs man slipper skriva namnet på kommandotolk varje gång man använder dem). `kommando1` görs körbar med UNIX-kommandot `chmod` enligt

```
chmod +x kommando1.
```

Därefter används kommandofilen med t ex

```
kommando1 Hejsan
```

☐ Prova detta!

> **Anm.** Kommandofiler som används på detta sätt tolkas av kommandotolken `sh`, inte `tcsh`, som på E är den normala kommandotolken för det man själv skriver in i ett terminalfönster. I grunden fungerar båda på samma sätt, men lite mer avancerade saker skrivs på olika sätt. Vill man ange explicit vilken kommandotolk en kommandofil ska tolkas av kan man ange detta på första raden i filen. För att få `tcsh` kan man skriva `#! /bin/tcsh -f`

Uppgift 8. Kommandot e9rwho

☐ Tillverka ett nytt kommando som fungerar som `rwho` (som vi provade i Uppgift 1), men utskriften begränsas till användare vars namn börjar på `e9`. Kalla kommandot `e9rwho`.

Uppgift 9. Finns det några handledare i närheten?

☐ Om man inte ser någon handledare när man vill bli godkänd kan man ju alltid kolla om någon handledare är inloggad. Kommandot `~hallgren/Intro/vvh` använder v (se Uppgift 1) och en variant på `grep` för att konsturera en lista med inloggade handledare. Prova det! Titta också hur det är gjort. Lägg märke till att kommandofiler kan innehålla kommentarer (text som kommandotolken struntar i) som föregås av #. Vad heter varianten av `grep` som används? _____

Uppgift 10. Kommandot inskrivna

☐ Tillverka ett nytt kommando med namnet `inksrivna`, som ger en lista med namnen på de som är inskrivna på E ett visst år. Med

```
inskrivna 99
```

☐ ska man alltså få namnlistan vi tillverkade i Uppgift 5, men kommandot ska även fungera för andra årtal. Hur många av de som skrevs in 1989 har fortfarande kvar sitt konto på Etek-systemet? _____

3 Avslutande kommentarer

Vi har sett några användbara kommandon som bearbetar textfiler och hur man kan sätta samman enkla kommandon med rör för att lösa med komplicerade uppgifter. Detta fungerar tack vare att kommandona är gjorda så att de använder "standard-in" och "standard-ut" för in- och utmatning.

Nuförtiden är det förstås inte bara text man bearbetar i sin dator, utan även bilder och ljud, t ex. Tekniken med rör kan fortfarande användas. För bildbehandling finns t ex PNM-paketet. Ett kommando som konverterar en GIF-bild till en JPEG-bild med halva storleken skulle t ex kunna se ut så här:

```
giftoppm bild.gif | pnmscale 0.5 | cjpeg >bild.jpg
```