Programmering 1

Laboration 1: Terminalanvändning, texteditering och kompilering i Linuxmiljö.

Denna laboration är tänkt att både utföras och **redovisas** under de schemalagda timmar som finns anslagna för ändamålet. Ifall du inte har möjlighet att delta i laborationstillfällena, eller inte hinner slutföra laborationen under den schemalagda tiden måste du redovisa dina resultat i en **skriftlig laborationsrapport** som skickas in via Moodle. Ett obligatoriskt delkrav för att kunna bli godkänd i kursen är att **alla laborationer är godkända**.

1. ***Starta upp den textbaserade terminalmiljön i Linux***

   I kursen Programmering 1 kommer du dagligen att använda Linux "terminalmiljö" för att köra kommandon (t.ex för att kompilera och köra era egna program). Följande övningar är avsedda att introducera några av denna terminalmiljös kommandon och egenskaper.

   Att göra: Starta upp Terminalapplikationen:
   Öppna "Startmenyn" och välj *Applications* => *System* => *Terminal*.

   Du ser nu en text i stil med **`[`**joakim@lab02 **`~]$`**
   Detta ser litet kryptiskt ut, men betyder i översättning: "Du är användaren **`joakim`** och du är inloggad på en dator som heter **`lab02`**. Du befinner dig i ditt eget Linux-hemområde, som i Linuxmiljö hänvisas till med hjälp av symbolen **`~`** , uttalas "tilde". Det avslutande dollartecknet kan slutligen översättas som "Jag väntar ödmjukt på att få uppfylla dina befallningar"

   

   Tecknet **`~`** fås fram genom att trycka **`alt gr`** och **`^`** följt av ett mellanslag.

2. ***Använd kommandot ls för att lista filer***

   Kommandot **`ls`** kan användas för att lista filerna och katalogerna i en viss katalog.

   Att göra: Kör kommandot **`ls`** i din hemkatalog. Testa även **`ls -l`** och **`ls -a`** för att få mera detaljerade listningar. Man kan även kombinera de två 'switcharna' med kommandot **`ls -la`**.

   

   Om du råkar ge ett felaktigt kommando och t.ex. kommer in i ett läge där kommandoprompten inte fungerar som den ska kan du testa att trycka **`Ctrl-C`** för att avbryta – det funkar oftast.

3. ***Använd kommandot cd för att byta katalog***

Kommandot **`cd`** ("**c**hange **d**irectory") används för att förflytta sig mellan olika kataloger. "Rotkatalogen" högst uppe i kataloghierarkin betecknas som redan nämnts **`/`**.

Att göra: Testa t.ex. att gå till rotkatalogen med kommandot **`cd /`** och sedan tillbaka till ditt hemområde med **`cd ~`**
Gå tillbaka till rotkatalogen och testa sedan **`cd /home/ditt_användarnamn`** - detta är ett annat, aningen mera krångligt, sätt att flytta sig till ditt hemområde.

Du kan använda *tabulatorknappen* för att automatiskt fylla i slutet på kommandon och katalog/filnamn som du håller på och skriver. Pröva detta t.ex. i samband med **`cd`**-kommandot!

Du kan bläddra bland tidigare givna kommandon med pil upp/pil ned.

Att redovisa: Tag (t.ex. med hjälp av kommandot **`cd`**) reda på vad katalogerna **`.`** och **`..`** står för i listan som du ser när du ger kommandot **`ls -a`** (i vilken katalog som helst).

4. ***Använd kommandot man för att få information och hjälp för olika kommandon***

Använd **`man`** för att få mera information om de olika kommandona.

Att göra: Testa t.ex. **`man cd`** eller **`man pwd.`**
- **`cd`** är ett så kallat 'inbyggt' kommando vars man-avsnitt har buntats ihop med en hel hög andra inbyggda kommanon
- Du kommer ut ur manualen genom att trycka **`q`**

5. ***Skapa och ta bort filer och kataloger***

Filer och kataloger kan skapas och tas bort antingen via den grafiska filhanteraren eller med hjälp av kommandon i terminalfönstret. Ibland kan det senare alternativet faktiskt vara betydligt smidigare.

- Kommandona **`mkdir`** och **`rmdir`** skapar respektive raderar kataloger.
Exempel:
`mkdir myDirectory` skapar en underkatalog vid namn **`myDirectory`** i den katalog du för tillfället befinner dig
`rmdir myDirectory` raderar katalogen **`myDirectory`**

- Kommandona **`touch`** och **`rm`** skapar respektive raderar filer
Exempel:
`touch newFile` skapar en ny, tom, fil vid namn **`newFile`**
`rm newFile` raderar filen

- Kommandot **`cp`** används för att kopiera filer
Exempel:
`cp ~/myOldFile ~/mySubDirectory/` flyttar filen **`myOldFile`** från hemkatalogen till underkatalogen **`mySubDirectory`**

Kom ihåg att du kan använda tabulatorknappen för att minska på skrivandet när du anger katalognamn!

- Kommandot **`mv`** används för att flytta filer och kataloger
Exempel:
`mv ~/myOldFile ~/myNewFile` flyttar (=i detta fall byter namn på) filen **`myOldFile`** till **`myNewFile`**

Undvik 'skander', mellanslag och specialtecken (förutom _ och -) i fil- och katalognamn. Linux klarar troligen av dessa, men det kan uppstå problem om du t.ex. flyttar över filerna till ett annat operativsystem.

6. *Skapa en katalogstruktur*

För att hålla reda på alla filer som du kommer att skapa under kursens gång behövs en lättnavigerbar katalogstruktur. Exakt hur den ser ut är upp till dig själv men du bör ha någon form av struktur för att lätt hitta bland dina program.

Att göra: Gå till din hemkatalog och skapa en lämplig katalogstruktur för lagring av egna program under kursens gång.

Skapa en katalogstruktur med en "toppkatalog" **`programmering1`** (eller liknande) och underkataloger t.ex. **`labb1`**, **`labb2`**, **`inl1`**, **`inl2`**, etc. Du kan skapa några underkataloger nu och skapa ytterligare underkataloger efterhand som de behövs under kursens gång. Använd terminalfönstret och kommandon som **`mkdir`** och **`cd`**, eller den grafiska fönsterhanteraren för att skapa katalogerna.

Att redovisa: Visa upp din katalogstruktur för läraren. Visa också att du med ett kommando i terminalfönstret kan förflytta dig från en katalog (t.ex. **`programmering1/labb1`**) till en annan (t.ex. **`programmering1/labb2`**)

7. *Starta och använd en texteditor från terminalen*

Under denna programmeringskurs kommer vi att använda en texteditor för att skriva och spara källkodsfiler. Dessa kompilerar och testkör vi sedan från kommandoraden i ett terminalfönster. En användarvänlig editor med bra stöd för en mängd olika programmeringsspråk är **`kate`**.

Att göra: Starta editorn genom att ge kommandot **`kate`** .

Du kan foga ett **&**-tecken efter kommandots namn. Testa också att starta kate på detta sätt (**`kate &`**).

Att redovisa: Vad har **&** för inverkan?

Utforska programmets olika menyer och flikar. Skriv litet text, spara texten, stäng filen och öppna sedan filen på nytt för att göra ändringar. Genom att använda snabbval (tex **`<Ctrl>S`**) istället för att gå via menyer eller ikoner effektiverar man arbetsflödet. Det är även praktiskt att förhandsöppna en fil då man startar editorn genom att man befinner sig på rätt ställe i filsystemet och anger ett filnamn som parameter till **`kate`** från terminalfönstret.

Att göra: Använd **`cd`** för att gå till katalogen **`~/programmering1/labb1`**.
Starta editorn och skapa samtidigt en ny tom fil genom att ge kommandot **`kate test.txt`**.
Observera att du alternativt kan ange hela sökvägen till den nya filen när du startar editorn:
`kate ~/programmering1/labb1/test.txt`

8. *Starta och använd andra program från terminalen*

När du startade **`kate`** räckte det med att ge programmets namn. Detta fungerar endast om programmet i fråga finns lagrat i någon av de kataloger som systemet är konfigurerat att söka efter kommandon i. Dylika kataloger är t.ex. **`/bin`**, **`/usr/bin`** och **`/usr/local/bin`**

Att göra: Testa t.ex. att endast skriva **`ka`** och trycka tabulatorknappen två gånger. Detta visar en lista över alla dylika kommandon som *börjar* på ka.

För att starta program i andra kataloger måste hela sökvägen till denna katalog anges. Detta gäller t.om. om programmet finns lagrat i samma katalog som du för tilfället befinner dig i – i detta fall startas programmet med kommandot **`./programmets_namn`**

Att göra: Testa att starta programmet **`mybreakout`**, som finns lagrat i katalogen **`/home/public/prog1`**. Starta först programmet **utan** att förflytta dig från din egen hemkatalog, och sedan genom att **först** flytta dig till **`/home/public/prog1`** och därefter starta programmet därifrån. Kom ihåg att du kan använda tabulatorknappen för att minska på skrivandet!

Att redovisa: Vilka kommandon använde du för att starta programmet?

Linux använder sig av ett system med *användargrupper*, där varje användare kan höra till ett antal olika grupper. Grupptillhörigheten avgör till vilka kataloger du har åtkomst. Varje användare har fullständiga rättigheter till sin egen hemkatalog. **`/home/public`** är en katalog där de användare som hör till gruppen lärare har fullständiga rättigheter, medan de som hör till gruppen studerande endast har läsrättigheter. (Då du ger kommandot **`ls -l`** indikerar bokstavskombinationerna **`drwx`** vilka rättigheter du har till de olika filerna och katalogerna.)

9. *Programkod i C*

Källkod i C skrivs som vanliga textfiler och editorn **`kate`** har bra stöd för denna typ av aktivitet. Kate kan anses vara en enkel så kallad "integrerad utvecklingsmiljö" (IDE) eftersom den förutom en texteditor också innehåller en filhanterare samt möjlighet att öppna ett inbyggt terminalfönster för att t.ex. kompilera program. Dessutom stöder Kate *syntax highlighting*, vilket innebär att programkod visas med en viss färgsättning som gör det lättare att se hur programkoden är strukturerad, samt automatisk *indentering* av programkod.

Att göra: Skriv raden nedan i en fil som heter **`hello.c`**. Spara denna i katalogen för laboration 1. **Obs: Skriv inte av den första raden ("Programkod 1....")**
Verifiera att kate automatiskt färglägger nyckelordet **`int`** med blå färg när du sparar filen.

Programkod 1: Första raden för ett program i C

```
int main ()
```

Det finns olika stilar för programmering och det är egentligen fritt fram att använda vilken stil som helst så länge man är konsekvent, t.ex. vad gäller indentering. En tumregeln är dock att man för ett projekt/arbetsplats använder den stil som är rådande där. I **`kate`** kan man ställa in en del av stilen på förhand samt aktivera sådana funktioner som underlättar programmeringen.

Att göra: Sök under menyerna *Settings* och *Tools* tills du hittat allt det du behöver ändra/kryssa i för att:
- indenteringen skall göras enligt "C-style" (borde redan vara aktiverat när textfilen slutar på .c)
- radnummer skall visas
- indentering görs med fyra mellanslag
- Funktionerna "Highlight range between selected brackets" och "Automatic brackets" skall vara aktiverade.

Att redovisa: Utöka nu ditt tidigare program genom att skriva källkoden nedan. Vad allt märker du att nu händer automatiskt?

Programkod 2: Första programmet i C

```
#include <stdio.h>
int main ()
{
printf ("Hello, World!\n");
return 0;
}
```

10. *Kompilering och körning av program*

En källkodsfil i C måste kompileras innan programmet verkligen kan testköras i systemet. Allt detta utför vi under denna kurs från terminalfönstret. Eftersom det är vanligt med syntaxfel och "buggar" i program blir det ofta så att man hoppar fram och tillbaka mellan texteditorn och terminalfönstret (skriv kod - spara -kompilera - korrigera - spara - kompilera - kör - korrigera - osv ...). Snabbvalet **<Alt><Tab>** flyttar smidigt mellan aktiva program på din desktop.

Alternativt kan man även använda **kate**s inbyggda terminal. En fördel är då att denna terminal automatiskt kan hållas synkroniserad med den källkodsfil du för tillfället arbetar med – behovet av att flytta sig mellan olika kataloger med **cd** minskar på detta sätt betydligt.

Att göra: Se till att du snabbt kan förflytta dig från ett separat terminalfönster till editorn och tillbaka med hjälp av ett snabbval. I terminalen skall du så klart befinna dig i samma katalog som du sparat källkodsfilen i. Kör kommandot **gcc hello.c** och studera filen som produceras. Du kan testköra den genom att ge kommandot **./a.out**. (Kom ihåg den inledande punkten!).

Pröva även att använda **kate**s inbyggda terminal. Kontrollera att plugin-modulen "Terminal tool view" är aktiverad och att den automatiska synkroniseringen mellan terminalen och editorn är aktiv. Den aktiva katalogen i terminalen skall automatiskt förändras då du i **kate** öppnar filer från olika kataloger.

Att redovisa: Kontrollera innehållet i katalogen med **ls -la**. Observera att åtkomstflaggorna för filen **a.out** inkluderar bokstaven **x**. Vad tror du detta innebär? Vad händer då du kör **a.out**?

Ändra ditt program så att hälsningsfrasen är svensk istället för engelsk. Hur ser raden som börjar med **printf** ut i så fall? Vad händer om du utelämnar **\n** i slutet av **printf**()-satsen?

Tag slutligen reda på (Google is your friend) hur du kan få **gcc** att ge den körbara (*executable*) filen ett annat namn än **a.out**.