Uppgift 2, Grundläggande Linux

Termin: HT20

Kurs: Linux som utvecklingsmiljö

Student: Fredrik Strömbäck, 19920823-1556

Del 1

Testa följande kommandon och program. Beskriv deras funktion och ge exempel på 2-3 användbara växlar och vad de används till (alla kommandon har dock inte växlar):man, info, cp, mv, mkdir, rmdir, rm, find, cd, pwd, df, ps, du, tar, seq, whoami, users, who, whereis, cat, tee, more, less, uniq, tail, echo, which, wget, cut, grep, sort och wc.

Kommandon

man: Används för att öppna upp manualen för ett kommando/program som man kan använda i terminalen.

Exempel:

man ls: Kommadot visar manualen för ls-programmet.

man -k: Med hjälp av -k kan söka efter en manual, argumentet som skickas med gör till en reguljärt uttryck för att hitta en passande manual.

man -w: Med -w berättar vart manualen som ”man” hittar, finns i filsystemet.

info: Används för att presentera Info-filer till ett kommando/program i terminalen. Filerna presenteras i ett annat format där innehållet kan delas upp i sidor vilket gör att t.ex. länkning för lättare navigering fungerar. Innehåller oftast mer information än vad ”man” presenterar.

Exempel:

info ls: Kommandot visar dokummentationen för ls-programmet.

info -k: Med hjälp av –k söker man upp alla de infofiler som finns som innhåller den sträng som man skickar in som argument.

info -a: Hittar alla manualer som matchar argumentet och listar dem därefter.

cp: Används för att kopiera filer.

Exempel:

cp test.jpg test2.jpg: Kommandot kopiera test.jpg och kopian döps till test2.jpg

cp -i: Med -i kommer cp varna användaren innan en fil skrivs över.

cp -r: För att kunna kopiera en hel katalog (och underliggande kataloger) behövs -r läggas till.

mv: Används för att flytta på filer. Kan även användas för att ändra namn på en fil.

Exempel:

mv test.jpg ../test2.jpg: Flyttar filen test.jpg en katalog upp och ändrar namnet till test2.jpg.

mv -i: Med -i kommer mv varna användaren innan en fil skrivs över (precis som cp).

mv -n: För att förhindra att en fil skrivs över om en fil skulle hittas med samma namn.

mkdir: Används för att skapa kataloger.

Exempel:

mkdir test: Skapar en ny katalog vid namn test.

mkdir -p one/two/three: För att skapa föräldrarkataloger om det behövs. Så även katalogerna ”one” och ”two” skapas om de inte redan finns.

mkdir -m: Används för att sätta rättigheter redan när katalogen skapas.

rmdir: Används för att ta bort tomma kataloger.

Exempel:

rmdir test: Tar bort en katalog vid namn test.

rmdir -p one/two/three: Tar bort katalogstrukturen som skickas med som argument, så länge katalogerna är tomma efter att underkatalogen har tagits bort.

rmdir -v: Presenterar ett informationsmeddelande efter varje katalog som har bearbetats.

rm: Används för att ta bort filer och kataloger (flagga/växel behövs för att ta bort kataloger).

Exempel:

rm test.jpg: Tar bort filen test.jpg.

rm -r: För att ta bort en hel katalog samt underkataloger och filer.

rm -i: Ber användaren att bekräfta innan filen tas bort.

find: Används för att hitta filer och kataloger i en katalogstruktur.

Exempel:

find .: Skriver ut alla filer och kataloger som kan hittas under katalogen man befinner sig i.

find . -name test.jpg: För att hitta en fil eller katalog med ett specifikt namn, under den katalog man befinner sig i.

find . -user testuser: Hitta filer som ägs av en specifik användare, under den katalog man befinner sig i.

cd: Används för att byta katalog som man befinner sig i.

Exempel:

cd Documents/one/two/three: Förflyttar mig/skalet till mappen som heter three.

pwd: Används för att skriva ut katalogen man befinner sig i.

df: Skriver ut information om filsystemet totala lagringsutrymme och tillgängligt utrymme.

Exempel:

df -h: Visar informationen så det blir lättare för oss männsikor att läsa.

df –total: Visar även totala lagringsutrymme.

ps: Visar de processer som körs i det nuvarande skalet.

Exempel:

ps -A: Visar alla processer som körs

ps -x: Visar alla processer som ägs av användaren som man är inloggad som.

du: Presenterar hur mycket lagringsutrymme en fil eller alla filer i en katalog tar upp.

Exempel:

du -h: Presenterar resultaten i enheter som är lättare att läsa för oss människor.

du -a: Presenterar även underliggande kataloger.

tar: För att skapa eller packa upp arkiv-filer.

Exempel:

tar -cf testtar.tar *.txt: Skapar en ny arkiv-fil som heter ”testtar” och lägger in alla txt-filer som hittas i aktuella katalogen.

tar -xf testtar.tar: Packar upp alla filer från testtar.tar och lägger i aktuell katalog.

tar -tf testtar.tar: Visar alla filer som finns i testtar.tar.

seq: Genererar nummer med tillsagt intervall.

Exempel:

seq -f ”Nr:%02g” 10: Gör så att siffrorna kan presenteras i ett visst format som man väljer. Min output blir Nr:01, Nr:02… Nr:10.

seq -s ” nr:” 10: Separerar siffrona med strängen ” nr:”. Min output blir 1 nr:2 nr:3… nr:10

whoami: Presenterar vem jag är inloggad som just nu.

users: Presenterar de användarna som är inloggade på systemet just nu.

who: Presenterar användarna som är inloggade, när de loggade in och vart dem loggade in ifrån (remote host name).

who -b -H: Presenterar när datorn bootade upp senast.

whereis: Används för att presentera vart binära/källkods filer och manualer för ett kommado finns i systemet.

Exempel:

whereis ls: För att hitta filerna och manual gällande ls.

whereis -b: Presenterar endast de binära filerna.

whereis -M /usr/share/man/: Begränsar whereis till att bara söka efter manualer i katalogen som skickas med.

cat: Läser innehållet och skriver ut innehållet ur filer.

Exempel:

cat -n text.txt: Presenterar innehållet i text.txt och skriver ut rad nummer för allt innehåll i filen.

cat -E text.txt: Markerar slutet på varje rad.

tac text.txt: Presenterar innehållet i filen i omvänd ordning.

tee: Kan både skriva ut inputen i standard output och sparar det i en fil.

Exempel:

tee -a text.txt: Skriver inte över innehållet utan lägger endast till den nya inputen.

more: Presenterar textfiler i terminalen men delar upp dfilen i sidor om den skulle var för lång.

Exempel:

more -d text.txt: Visar hjälpmedel för navigation.

more -s text.txt: Slår ihop tomma rader som kommer efter varandra till en enda tom rad.

less: Presenterar innehållet i en fil sidvis, men laddar inte hela filen på en gång utan endast den del som presenteras.

Exempel:

less -F text.txt: Om filens innehåll kan presenteras på första sidan så avslutas less automatiskt.

less -E text.txt: När användaren når slutet av filen så avslutas less automatiskt.

uniq: Hjälper en hitta duplikationer av rader som kommer efter varandra i en fil och filtrerar ut dessa.

Exempel:

uniq -c text.txt: Rapporterar hur många gånger en rad upprepades.

uniq -d text.txt: Presenterar endast rader som det finns flera utav.

uniq -u text.txt: Presenterar endast unika rader.

tail: Presenterar ett x-antal av de sista raderna av en fil. 10 är default.

Exempel:

tail -n 3 text.txt: För att skippa default och bara printa de sista 3 raderna.

tail -f text.txt: Presenterar endast de 10 sista raderna men uppdaterar om en ny rad tillkommer.

echo: Skriver ut inputen i terminalfönstret.

Exempel:

echo -n hello: Tar bort efterföljande ny rad.

echo -e: Tillåter att formatera strängen med ”/” kommandon.

which: Presenterar vilka filer som skulle exekverats till det följande kommandot.

wget: Används för att ladda hem filer över olika protokoll över internet.

Exempel:

wget https://testsida.com: Laddar hem källkods filen för testsida.com.

wget -i filMedURL: Läser en URL från en fil.

cut: Används för att klippa ut en del av varje rad och presentera resultatet.

Exempel:

cut -b 1,2,3 text.txt: Klipper ut och presenterar de karaktär 1,2 och 3 från filen text.txt.

cut -c 2,7 text.txt: Klipper ut och presenterar kolumn 2 och 7.

grep: Söker i en fil efter ett specifikt mönster av karaktärer och presenterar de rader som innehåller dess mönster.

Exempel:

grep -c ”Linux” text.txt: Räknar hur många gånger mönstret kan hittas och presenterar endast siffran.

grep -n ”Linux” text.txt: Presenterar raderna där mönstret dyker upp och på vilken rad i filen.

sort: Används för att sortera en fil i en viss ordning.

Exempel:

sort -r text.txt: Sorterar raderna i filen i omvänd bokstavsordning.

sort -n nummer.txt: För att sotera filer med nummeriska värden i.

wc: För att räkna ord, rader, karaktärer eller bitar.

Exempel:

wc -w text.txt: Räknar hur många ord det finns i filen text.txt.

wc -m: Räknar hur många karaktärer det finns i en fil.

Del 2 Användare och grupper

Skapa tre kataloger, data, admin and market.
Skapa tre användare för Adam Andersson, Peter Pettersson och Lisa Persson ch ge dessa lämpliga användarnamn.
Skapa tre grupper datagroup, admingroup and marketgroup.
Sätt upp ett system där Adam och Peter kan hantera och jobba med filer i data, Adam och Lisa kan hantera och jobba med filer i admin och Peter och Lisa kan hantera och jobba med filer i market. Inga andra ska ha access till dessa kataloger.

2a: Hur du löste uppgift 2, och bifoga kopior på adekvata delar från filerna /etc/group, /etc/passwd samt utskrifter från ls -l <the directory where you created the above directories>

Svar:

1. För att skapa de tre katalogerna använde jag mig av

```
$mkdir data admin market
```

2. För att skapa de olika användarna använde jag

```
$sudo adduser <användarnamn>
```

```
$cat /etc/passwd
```

```
adam_a:x:1001:1004:Adam,,,:/home/adam_a:/bin/bash
peter_p:x:1002:1002:Peter Pettersson,,,:/home/peter_p:/bin/bash
lisa_p:x:1003:1003:Lisa Persson,,,:/home/lisa_p:/bin/bash
strombach@ThinkPad-T480:/etc$
```

3. För att skapa grupperna

```
$sudo groupadd <gruppnam>
```

4. För att lägga till användare i grupper

```
$sudo gpasswd -a <användarnamn> <gruppnamn>
```

```
$cat /etc/group
```

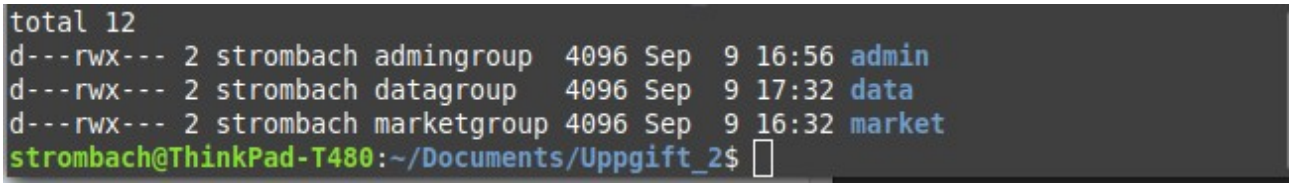
```
datagroup:x:1004:adam_a,peter_p
admingroup:x:1005:adam_a,lisa_p
marketgroup:x:1006:lisa_p,peter_p
strombach@ThinkPad-T480:/etc$
```

5. Ändra ägare till mappen och underkataloger och filer

```
$sudo chgrp -R <gruppnamn> <katalognamn>
```

6.Ta bort access för andra användare.

```
$sudo chmod -R uo-rwx <katalognamn>
```



2b: I uppgift 2 står det att ingen annan användare ska ha tillgång till katalogerna. Är detta möjligt eller finns det en användare som aldrig går att utestänga?

Svar:

Den enda användaren som alltid har tillgång till allt är **root.** Denna användare går aldrig att utesluta.

Del 3 Filsystemet

Beskriv vad följande kataloger i filstrukturen innehåller för typer av filer:
/boot: Innehåller de filer som krävs för att starta upp systemet (”boota”).
/etc: Innehåller många om inte alla konfigurationsfiler för systemet.
/sbin: Innehåller program som endast superuser/root kan exekvera, dessa program kan då köras med ”sudo”.
/bin: Innehåller de program som kan köras som vilken användare som helst.
/usr: En blandning av filer som t.ex. manualer, ikoner, bibliotek och program som behöver delas mellan program och tjänster. Tidigare användes denna som ”/home”.
/var: Innehåller bl.a. cache, log och temporära -filer. Filer som kan komma att ändras under tiden systemet är igång. Därför gavs namnet efter ”varible”.
/dev: Devices. Alltså filer för alla komponeter som används i systemet. T.ex. ett usb-minne, en webbkamera eller en hårddisk.
/home: Detta är mappen där en användare har sina egna filer så som bilder, videor och dokument.

Del 4

Använd de tillåtna kommandona enligt nedan och kombinera ihop dem med pipor så att följande funktioner nås:
a. Lista alla filer, större än 1MB, i den befintliga katalogen i en sorterad lista. Man ska alltså inte traversera ner i katalogstrukturen! Tillåtna kommandon: **find, sort**

Svar:

```
strombach@ThinkPad-T480:~/Downloads/Zotero_linux-x86_64$ find ./ -maxdepth 1 -ty
pe f -size +1M
./libmozavcodec.so
./pdfinfo
./zotero.jar
./libxul.so
./pdftotext
./omni.ja
strombach@ThinkPad-T480:~/Downloads/Zotero_linux-x86_64$ find ./ -maxdepth 1 -ty
pe f -size +1M | sort
./libmozavcodec.so
./libxul.so
./omni.ja
./pdfinfo
./pdftotext
./zotero.jar
strombach@ThinkPad-T480:~/Downloads/Zotero_linux-x86_64$
```

Kommando: $find ./ -maxdepth 1 -type f -size +1M | sort

b. Räkna hur många olika rader det finns i filen adress.txt. Dvs hur många rader skulle det bli om man tar bort alla dubletter. Tex ska denna fil returnera värdet 4. Du ska förutsätta att filen adress.txt finns i befintlig katalog.
Tillåtna kommandon: **sort, uniq, wc**

Svar:

```
strombach@ThinkPad-T480:~/Documents/Uppgift_2$ sort adress.txt | uniq | wc -l
4
strombach@ThinkPad-T480:~/Documents/Uppgift_2$
```

Kommando: $sort adress.txt | uniq | wc -l

c. Lista alla inloggade användare med enbart inloggningsnamnet i en sorterad lista. Ingen användare får listas mer än en gång. Tillåtna kommandon: **who, cut, sort, uniq**

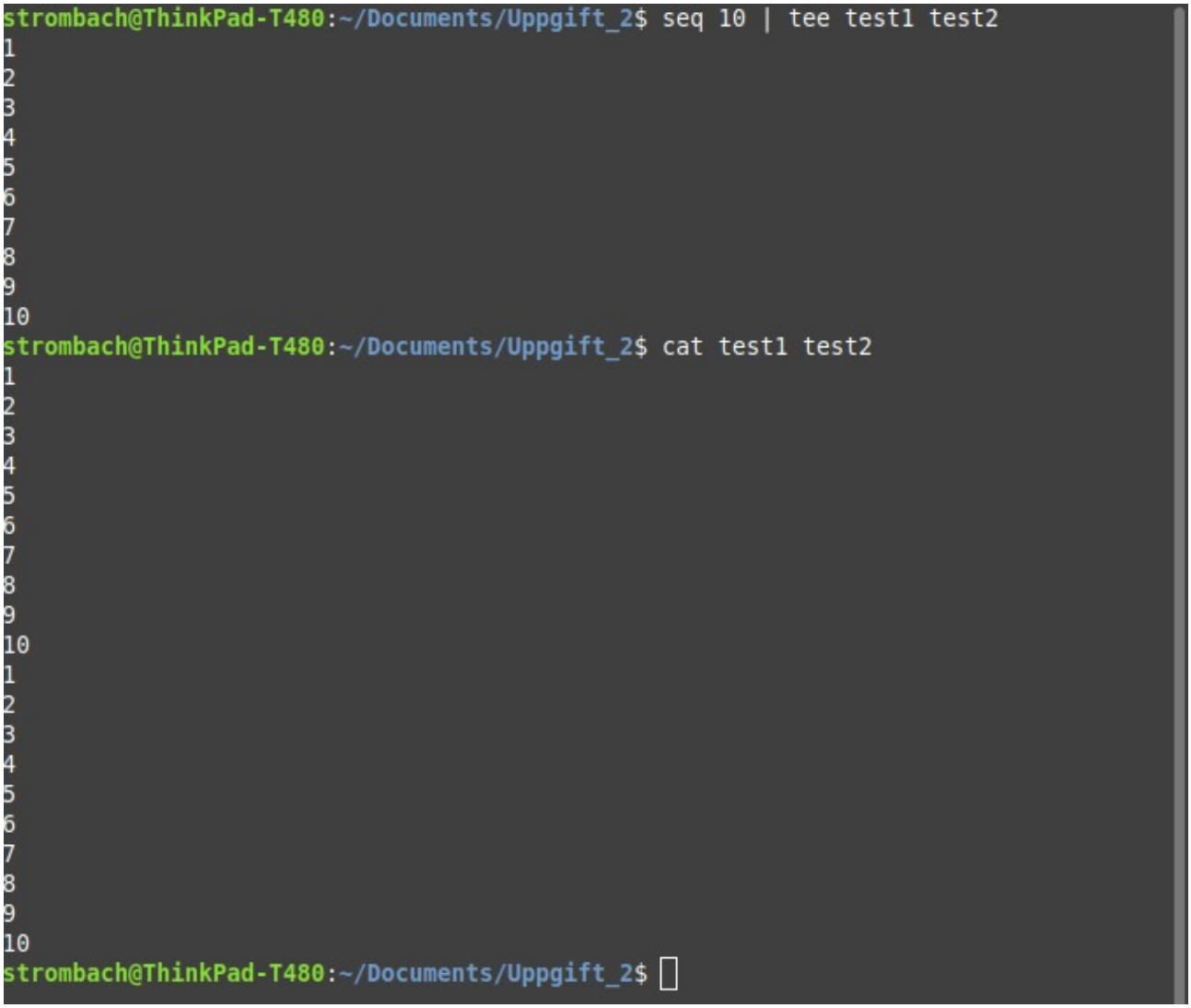
Svar:

```
strombach@ThinkPad-T480:~$ who
strombach tty7         2020-09-08 16:55 (:0)
adam_a   tty8         2020-09-10 11:41 (:1)
strombach@ThinkPad-T480:~$ who | sort | uniq | cut -d " " -f 1
adam_a
strombach
strombach@ThinkPad-T480:~$
```

Kommando: $who | sort | uniq | cut -d ” ” -f 1

d. Skriv siffrorna 1 tom 10 till filerna test1 och test2.Tillåtna kommandon: **seq, tee**

Svar:



Kommando: $seq 10 | tee test1 test2

Del 5

Använd valfri editor och skapa 3 filer, a.txt, b.txt och c.txt, med 3-4 rader valfri text. Packa ihop de 3 filerna i ett arkiv med namn del_5.tar. Använd tar för att verifiera att de 3 filerna är intakta. Använd sedan tar för att lägga till ytterligare en fil, d.txt, till arkivet.

Svar:

1. Skapa filerna

$touch a.txt b.txt c.txt

2. Skriv in 4 rader lorem ipsum med nano.

$nano a.txt

3.Skapa arkiv

$tar -cf del_5.tar a.txt b.txt c.txt

4.Kolla så att filerna ligger i del_5.tar.

$tar tvf del_5.tar

5. Skapa d.txt och använd nano för att skriva in lorem ipsum.

$touch d.txt

$nano d.txt
6. Lägg till d.txt till arkivet.
$tar -rf del_5.tar d.txt

Del 6

Testa följande kommandon:
ls / > lsoutput.txt
ls /hh > lsoutput2.txt
ls / 2> lserror.txt
ls /hh 2> lserror2.txt

Jämför de resultat du får på terminalen och till filerna för resp. kommando och beskriv vilka slutsatser du kan dra av resultaten.
Beskriv de 3 fildescriptorerna (STDIN, STDOUT och STDERROR) vilka alltid finns till hands för ett program. Hur används de och hur kan du omdirigera dem?

I exemplet ovan använder vi ett enkelt >-tecken. Vilken blir skillnaden om vi istället skriver >>?
tex ls / >> lsoutput.tx

Svar:
ls / > lsoutput.txt: Alla namn på katalogerna läggs i en .txt-fil. Terminalen visar även alla kataloger i /
ls /hh > lsoutput2.txt: En .txt-fil skapas men är tom. Terminalen skriver ut: ”ls: cannot access '/hh': No such file or directory”
ls / 2> lserror.txt: En tom .txt-fil skapas men namnen på katalogern i / skrivs ut i terminalen.
ls /hh 2> lserror2.txt: Här skrivs felmeddelandet ut i textfilen och inget skrivs ut i terminalen.

Slutsatser: Utan 2:an skrivs felmeddelandet (om det finns något) ut i terminalen. 2:an är kopplad till felmeddelanden. Skrivs då 2:an framför ”>” så är det endast felmeddelandet som kommer skickas till filen, annars tar den resultatet av ”ls /” och skickar till filen.

STDIN: Detta är den datastream som hanterar input i skalet, den accepterar text som input. Denna stream har värdet 0.
STDOUT: All output från skalet/programmet/kommandot hanteras av datastreamen stdout och har värdet 1. Med hjälp av värdet kan man omdirigera outputen till t.ex. en fil ”ls / 1> test.txt”.
STDERROR: Här är datastreamen som hanterar de error/felmeddelanden som uppstår i ett program/kommando. Har värdet 2.

>> vs >: Skillnaden på resultatet här är att ”>” skapar en fil om den inte redan finns och fyller det med data från t.ex. stdout. Existerar filen redan så kommer innehållet i filena att skrivas över utan varning.

Medans ”>>” också skapar en ny fil om den inte finns och skickar in omdirigerad data, så kommer den inte skriva över innehållet om filen redan finns utan kommer bara att lägga till den nya datan i slutet av filen.