Mälardalens universitet

DVA249/DVA267 Linux, HT2023

- Laboration 3 -

Förberedelser

Läs instruktionerna på Canvas och läs veckans kurslitteratur innan du börjar med laborationen. Vi rekommenderar även att du tittar på videomaterialet som tillhör laborationen. Svaren till flera av uppgifterna finns i kursmaterialet.

1 Läsa textfiler

1. Textfiler kan öppnas med `cat`, `more` och `less`. Vad är det för skillnad på dessa?
2. Öppna en textfil med kommandot `less` och beskriv hur du gör följande:
   (a) Söker efter specifika ord
   (b) Gå till sista raden i filen
   (c) Gå till första raden i filen

2 Kontrollera output från kommandon

Program kan arbeta med output från andra program. Detta kan åstadkommas genom att kombinera omdirigeringsoperatorer ('*redirection operators*') med filer eller att använda pipes, så kallade pipelines.

2.1 Pipes och omdirigering

Innan du ger dig på omdirigering kan det vara en bra idé att köra kommandot `set -o noclobber`. Använd `set -o` för att se status på noclobber. För att permanent slå på noclobber kan kommandot läggas till i `~/.bashrc`. Vad har `noclobber` för effekt?

> **Tips:** Alla kommandon som finns i filen `~/.bashrc` körs varje gång du startar ett nytt BASH-skal. Därför är det ett bra ställe att lägga till exempel alias som man vill alltid ska vara definierade.

1. Hur kan du räkna antalet filer och kataloger i en katalog med hjälp av kommandona `ls` och `wc` samt pipe?
2. Beskriv hur du med en kommandorad kan uppnå följande:
   (a) Skapa en fil med namnet `listing.log` som innehåller en lista med filer och kataloger i katalogen `/usr/bin`. Använd omdirigering.
   (b) Lägga till innehållet i katalogen `/etc` till filen `listing.log`.

3. Ladda ner `numbers.txt` från Canvas och svara på följande frågor:
   (a) Hur kan du sortera innehållet i `numbers.txt` i stigande ordning? Använd `cat`, `sort` och pipe.
   (b) Modifiera föregående kommando för att ta bort dubbletter.
   (c) Hur kan du sortera innehållet i `numbers.txt` i stigande ordning och spara resultatet i en textfil? Använd endast `sort` och omdirigering.
4. Lista de åtta största filerna och katalogerna i katalogen `/usr/bin`. Använd `ls`, `sort`, `head` och pipe.

3 Textredigerare

Det finns många textredigerare tillgängliga för Linuxsystem både för GUI och CLI. Exempel på textredigerare är `gedit`, `mousepad`, `vi`, `vim`, `emacs`, `kate` och `pico`. För Xubuntus grafiska gränssnitt finns textredigeraren Mousepad installerad som standard. Gedit är en av de mest kända textredigerarna för GUI. Om du vill kan du installera denna textredigerare.

För terminaler i Unix/Linux finns alltid textredigeraren Vi (eller den nyare versionen Vim) tillgänglig. Det finns andra textredigerare som kan vara enklare att använda, men dessa är inte alltid installerade på systemen. Exempel på andra textredigerare är `nano`, `pico` och `emacs`. De två förstnämnda är enkla editorer medan Emacs är en väldigt avancerad editor. Eftersom Vi/Vim alltid finns tillgänglig på alla Linuxsystem är det viktigt att lära sig enkel textredigering i Vi/Vim. Lär dig grunderna i Vi/Vim genom att göra *Lab 11 - Basic Scripting* i NetAcad.

Ubuntu/Xubuntu kommer även med `nano` installerat. Nano är en enklare textredigerare och är lämplig för editering av mindre textfiler.

1. Testa minst en textredigerare i GUI, exempelvis `mousepad` i Xubuntu eller `gedit` (*döpt till* `Text Editor`) i Ubuntu. Starta textredigeraren, skriv text och spara filen i din hemkatalog.
2. Testa nano i terminalen.
   (a) Skapa en textfil
   (b) Öppna filen med `nano`
   (c) Skriv text
   (d) Spara filen i din hemkatalog.
3. Testa Vi i terminalen:
   (a) Skapa en textfil
   (b) Öppna filen med `vi`
   (c) Skriv text
   (d) Spara filen i din hemkatalog.

> **Tips:** Titta på Lab 11 i NetAcad, manualsidan för Vi eller på `https://www.tutorialspoint.com/unix/unix-vi-editor.htm`.

4 Reguljära uttryck

Reguljära uttryck (‘*regular expressions*’ eller ‘*regex*’) är en sträng som beskriver vilket mönster som ska hittas i en text.

Kommandot `grep` används tillsammans med reguljära uttryck och utökade reguljära uttryck (‘*extended regular expressions*’) för sökning av textmönster. För utökade reguljära uttryck används grep tillsammans med flaggan `-E`.

1. Skriv ett kommando för att lista alla filer i katalogen `/usr` och dess underkataloger där namnet innehåller *man*. Använd `find`, `grep` och pipe. Delar av den förväntade outputen visas i Figur 1.

```
...
/usr/share/man/pt/man5/apt.conf.5.gz
/usr/share/man/pt/man5/apt_auth.conf.5.gz
/usr/share/libreoffice/help/en-US/text/shared/01/packagemanager.html
/usr/share/libreoffice/help/media/icon-themes/cmd/lc_dismantle.svg
...
```

Figur 1: Exempel på output för listning av kataloger innehållande *man*

2. Kommandot `grep` kan användas för att matcha textmönster. Läs igenom guiden på http://www.regular-expressions.info eller på manualsidan för reguljära uttryck (`man 7 regex`). Förklara vad för output följande kommandon kommer generera:
   (a) `ls -C1 /usr/bin | grep '^a'`
   (b) `ls -C1 /usr/bin | grep 'st$'`
   (c) `ls -C1 /usr/bin | grep 'p.*n'`
3. Förklara betydelsen av följande tecken i reguljära uttryck:
   (a) . (punkt)
   (b) * (stjärna)
   (c) + (plustecken)
   (d) ? (frågetecken)
4. Katalogen `/usr/bin` innehåller vanliga Linuxkommandon. Dock är alla filer i katalogen inte vanliga filer, istället är flera symboliska länkar till andra platser. Visa hur man kan använda `ls` och `grep` för att lista alla symboliska länkar i katalogen `/usr/bin`. **Tips:** Ifall du listar filer i katalogen i ‘*long format*’ så beskriver första tecknet på varje rad filtypen.

5 SED - Stream EDitor

Ibland är det användbart att kunna manipulera text i skript eller från kommandoraden. Två kommandon som ofta används för detta syfte är `awk` och `sed`. SED (Stream EDitor) är ett kraftfullt verktyg som används för att transformera text från en fil eller från en pipe. Med hjälp av `sed` kan du söka, hitta och byta ut, stoppa in och ta bort ord och rader med text. Du kan hitta mer information och hjälp på https://www.gnu.org/software/sed/manual/sed.html.

SED består av ett antal olika kommandon (som anges som ett argument till `sed`) som du kan använda för att utföra textmanipulation. Bland dessa kommandon finns `s`-kommandot ('*substitute*'). Syntaxen är `s/regex/replacement/flags`. Kör följande `ls`-kommando i en terminal:

```
ls /usr/bin | grep "pri" | sed "s/pri/###/"
```

Beskriv output av kommandot. Vad gjorde SED? **Tips:** Ibland måste man ge flaggan `-e` explicit för att indikera att `"s/pri/###/"` är SED-kommandot som ska köras, det vill säga `sed -e "s/pri/###/"`.

Fortsätt med att skriva ett kommando som ersätter alla 'a' med 'e' (använd SED-kommandot `s`). Hur ser kommandot ut? Du kan testa ditt SED-kommando genom att köra följande:

```
echo "abcde abcde ae" | sed <SED-kommando>
```

Output för exemplet bör bli `ebcde ebcde ee`. **Tips:** För att byta ut alla tecken på en rad (inte bara första matchningen) måste du använda den globala operatorn för SED-kommandot.

I nästa steg, skriv ett kommando, genom att använda dig av SED-kommandot `y`, som ändrar följande:

- alla 'a' till 'o'
- alla 'd' till 'm'
- alla 'i' till 'y'
- alla 'r' till 'e'
- alla 't' till 'n'
- alla 'y' till 'a'

Testa ditt SED-kommando med följande kommando:

```
echo "di datkri lavrs bytytys" | sed <SED-kommando>
```

Vad får du för output?

6 Skalskript

Skalskript (‘*shell scripts*’) är enkla program skrivna i ett tolkat programmeringsspråk som är inbyggt i Linuxskalet. Eftersom du kommer skriva dina skript i `BASH` bör du döpa dina filer med filändelsen `.bash` och använda programmet `bash` när du kör dina skript. Kör *inte* dina skript med hjälp av `sh`. Skript i BASH och SH är inte samma sak.

> **Tips:** Glöm inte att ange shebang i början av varje skript du skriver, för BASH är den `#!/bin/bash`. Kom dessutom ihåg att lägga till kommentarer i dina skript.

I BASH skriver man generellt sett miljövariabler och skalvariabler med VERSALER. Att då ge lokala skriptvariabler namn med gemener hjälper att undvika konflikter.

> **Tips:** Du kan hitta bra tutorials på:
>
> http://www.tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html

Var noggrann med att använda mellanrum på rätt ställen i BASH. Här är några exempel:

- `myvar1=''Hello Linux''` Denna tilldelning fungerar
- `myvar2 = ''Hello Linux''` Denna tilldelning fungerar ej
- `''string'' == ''string''` Denna jämförelse fungerar
- `''string''=='string''` Denna jämförelse fungerar ej

6.1 Argument

I denna del ska vi skapa ett skript som accepterar ett godtyckligt antal argument. Skriptet ska använda en variabel (kallad `num_of_arg`). Variabeln ska bli tilldelad det faktiska antalet argument givet till skriptet vid exekvering. Skriptet ska skriva ut värdet av `num_of_arg`, följt av värdet av det första argumentet. På nästa rad ska värdet av alla argument skrivas ut. *Lös denna uppgift utan att använda loopar. Använd inbyggda variabler!*

1. Skapa en ny fil och döp den till `6.1-argument.bash`
2. Skriv en shebang på första raden.
3. Skriv kommandona som krävs för att lösa uppgiften.
4. Spara filen.
5. Kör skriptet genom att köra kommandot `bash 6.1-argument.bash` i katalogen där du sparade skriptet.

När du kör skriptet bör output se ut som i Figur 2.

```
awk03@xubuntu:~$  bash 6.1-argument.bash aa bb ccc d
Number of arguments: 4
First argument: aa
All arguments: aa bb ccc d
```

Figur 2: Exempel på output från `6.1-argument.bash`

6.2 Manipulation av text

Skapa ett skript `6.2-manipulation.bash` som tilldelar värdet av `PATH` till en ny variabel `mypath`. Skriptet ska sedan skriva ut innehållet i `mypath` till terminalen följt av varje separat sökväg i `mypath` på varsin rad. Använd ett SED-kommando och `mypath` för att byta ut alla kolon mot ny rad-tecken (`\n`). *Använd inte en loop i skriptet.* När du kör skriptet bör output se ut som i Figur 3 (output beror på värdet av `PATH`).

```
awk03@xubuntu:~$  bash 6.2-manipulation.bash
mypath: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
Separated:
/usr/local/sbin
/usr/local/bin
/usr/sbin
/usr/bin
/sbin
/bin
/snap/bin
```

Figur 3: Exempel på output från `6.2-manipulation.bash`

6.3 Formatering av output

Skriv ett skript `6.3-formatering.bash` som kör `date`-kommandot. Output från `date` ska sparas i en variabel. Efteråt ska skriptet skriva ut en sträng som ser ut som följer (med dagens datum): `Today is Friday, December 31, 1999`

> **Tips:** Titta på lämpliga argument till alternativet `FORMAT` för `date`-kommandot.

6.4 If-satser och loopar

För att styra flödet i skript kan if-satser och loopar användas. I denna laboration använder vi if-satser, for-loopar i BASH-stil (som liknar for-loopar i Python) och for-loopar i C-stil (som liknar for-loopar i C).

6.4.1 If-satser

Skapa ett skript `6.4.1-scrutinize.bash` som gör följande: om du ger argumentet `scrutinize` ska skriptet skriva ut `NO WAY`. Annars ska skriptet skriva ut `You can always try again`.

6.4.2 For-loop i BASH-stil

Skapa ett skript `6.4.2-bashfor.bash` som skapar samma output som i uppgift 6.2, men denna gång ska du använda en for-loop i BASH-stil, se Kodruta 1.

```
for VAR in LIST/SEQUENCE
do
   COMMANDS
done
```

Kodruta 1: For-loop i BASH-stil

I ditt skript, skapa en mellanrumsseparerad lista (‘*space separated list*’) genom att ta innehållet från `PATH` och byt ut alla kolon mot mellanrum. Spara listan i en variabel och använd sedan den ovanstående for-loopen för att iterera över listan och skriva ut varje sökväg på en egen rad.

> **Tips:** Använd kommandsubstituering (‘*command substitution*’), `VAR=$(COMMAND)` för att spara output från ett kommando i en variabel (https://www.gnu.org/software/bash/manual/html_node/Command-Substitution.html).

> **Tips:** Som ett alternativ till SED kan du använda dig av BASH interna fältseparator (‘*internal fieald separator*’), förkortat IFS. I sådana fall sätter man värdet av variablen IFS till kolon och så kan du iterera över en kolonseparerad lista direkt, det vill säga innehållet i `PATH`.

6.4.3 For-loop i C-stil

Skapa ett skript `6.4.3-cfor.bash` som loopar 20 gånger och skriver ut texten `This is iteration <nr>` för varje iteration. Använd en for-loop i C-stil, se Kodruta 2.

```
for ((INITIALIZATION; TEST; STEP))
do
   COMMANDS
done
```

Kodruta 2: For-loop i C-stil

7 Skalskript som verktyg

Skriv ett skript `7-filetype.bash` som läser in ett godtyckligt antal argument och som för varje argument kollar om argumentet är en mapp, en vanlig fil, en exekverbar fil eller en symbolisk länk och skriver ut vilken typ det är. Det ska alltså vara möjligt att skriva `bash 7-filetype.bash *` för att få en lista över alla filer och kataloger och deras typ i den nuvarande arbetskatalogen. *Lös denna uppgift utan att använda dig av kommandot* `file`. Observera att ordningen du testar för olika typer är avgörande! **Tips:** Läs man-sidorna för BASH för att ta reda på hur du kollar filtyp i ‘*conditional expressions*’. Exempel på hur output kan se ut:

```
awk03@xubuntu:~$  bash 7-filetype.bash * missing
Desktop (Directory)
Documents (Directory)
filetype.bash (Executable)
shopping_list (Ordinary file)
myscript4_2.bash (Executable)
missing (Does not exist)
```

Figur 4: Exempel på output från `7-filetype.bash`

DVA249/DVA267 Linux, HT2023

- Quiz 3 -

Denna uppgift genomförs **individuellt och lämnas in genom att göra quiz ”Quiz 3” i Canvas** när du är klar med uppgiften. Uppgiften ska vara inlämnad innan deadline, annars kan vi inte garantera att vi hinner titta igenom era inlämningar innan examinationstillfället.

Vilket av följande kommandon visar information om datorns PCI-bussar?

(a) `uname -a`

(b) `cat /proc/cpuinfo`

(c) `cat /proc/pciinfo`

(d) `lspci`

(e) `lscpu`

(f) `mount`

(g) `df -h`

Quizet anses vara klar när du har 1 poäng på uppgiften i Canvas.

Mälardalens universitet

DVA249/DVA267 Linux, HT2023

- Inlämningsuppgift 3 -

Denna uppgift genomförs **i grupp och lämnas in genom att ladda upp filen i Canvas** när du är klar med uppgiften. Uppgiften ska vara inlämnad innan deadline, annars kan vi inte garantera att vi hinner titta igenom era inlämningar innan examinationstillfället.
Skapa ett skalskript med namnet `myinfo.bash`. Skriptet ska ge information om systemet beroende på vilken flagga användaren anger, se Tabell 1.

Tabell 1: Option och funktion för skriptet `myinfo.bash`

Option	Funktion
`-a, --all`	Skriver ut all information
`-v, --version`	Skriver ut Linuxversion
`-i, --ip`	Skriver ut IP-adressen
`-m, --mac`	Skriver ut MAC-adressen
`--help`	Skriv ut information om skriptet och tillgängliga flaggor

Output till terminalen ska likna exemplet i Figur 5. I skriptet måste du förändra output från olika kommandon för att få en stilren och snygg output till terminalen med enbart den efterfrågade informationen. Skriptet behöver inte hantera flaggor som skrivs ihop, exempelvis `myinfo.bash -vim`.

> **Tips:** Titta på kommandona `ip addr` och `lsb_release -a`. Andra användbara kommandon är `grep`, `awk` och `cut`.

```
awk03@xubuntu:~$ bash myinfo.bash --help
Usage: myinfo.bash OPTION...
Print out system information

OPTIONS
-a, --all        display all information
-v, --version    display linux version
-i, --ip         display IP address
-m, --mac        display MAC address
    --help       display this help and exit
awk03@xubuntu:~$ bash myinfo.bash -v
Linux version: Ubuntu 22.04 LTS
awk03@xubuntu:~$ bash myinfo.bash -i
IP address: 10.0.2.15/24
awk03@xubuntu:~$ bash myinfo.bash -m
MAC address(ether): 00:11:D8:31:3F:05
awk03@xubuntu:~$ bash myinfo.bash -v -i -m
Linux version: Ubuntu 22.04 LTS
IP address: 10.0.2.15/24
MAC address(ether): 00:11:D8:31:3F:05
```

Figur 5: Exempel på output från `myinfo.bash`

Uppgiften anses vara klar när du har 1 poäng på uppgiften i Canvas. Deadline för uppgiften är 24/11 kl 23:59.