

DVA249/DVA267 Linux, HT2023

- Praktiskt prov -

Exempel

Uppgift 1 (2 poäng)

När man kör ett kommando så letar BASH efter ett lämpligt program att köra i alla kataloger som finns i miljövariabeln PATH. Nu har du installerat ett nytt program **UPPAAL** i din hemkatalog. I katalogen `/home/ada/uppaal/bin` finns det nya programmet som ska vara tillgängligt.

Skriv ett kommando som lägger till katalogen i PATH-variabeln på lämpligt sätt.

Uppgift (2 poäng)

I en av inlämningsuppgifterna (Laboration 2) skrev du ett skript som säkerhetskopierade alla log-filer i `/var/log`. Studera exempellösningen i Kodruta 1.

```
#!/bin/bash
cd ~
mkdir Logs
cp /var/log/*.log Logs/
tar cvzf Logs/logs.tar.gz Logs/*.log
rm Logs/*.log
ls Logs
```

Kodruta 1: logs.bash

Modifera skriptet så att det istället säkerhetskopierar hela din hemkatalog och sparar säkerhetskopian i katalogen `/tmp`.

Skriptet ska uppfylla följande kriterier:

- Skriptet ska skapa ett arkiv med alla filer från din hemkatalog.
- Arkivet ska vara komprimerat och döpt till `backup-<username>.tar.gz`, där **<username>** är användarnamnet hos den som kör skriptet (skriptet ska fungera för olika användare).
- Arkivet ska sparas i katalogen `/tmp`.

DVA249/DVA267 Linux, HT2023

- Praktiskt prov -
Lösningsförslag

Lösningsförslag uppgift 1

```
PATH=$PATH:/home/ada/uppaal/bin
```

Lösningsförslag uppgift 2

```
#!/bin/bash  
cd /home  
tar cvzf /tmp/backup-$USER.tar.gz $USER
```

DVA249/DVA267 Linux, HT2023

- Laboration 1 -

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 Konfiguration av laborationsmiljö

Under denna kurs kommer vi att arbeta med operativsystemet GNU/Linux, och det måste installeras innan vi kan använda det. För att göra det enklare kommer vi att använda oss av *virtualisering*. Det innebär att vi skapar en virtuell dator, på vilken vi sedan installerar en Linuxdistribution.

Virtuella maskiner (VM) har flera fördelar, dels kommer ändringar man gör i Linux enbart påverka den virtuella datorn, vilket ser till att oavsett vad man kör för kommandon kommer man inte kunna förstöra något förutom på den virtuella datorn. Dels kan man spara '*snapshots*'. En snapshot är en form av säkerhetskopia, som kan användas för att gå tillbaka till en viss tidpunkt ifall något går snett.

1.1 Installera Linux på en virtuell maskin

Vi rekommenderar att du installerar VirtualBox och där skapar en virtuell maskin, och sedan installerar Xubuntu på den VM:en. Manualen för VirtualBox finns på:

<https://www.virtualbox.org/manual/UserManual.html>

OBS! I datorsalar

Ifall du använder dig av datorerna i laborationssalarna är VirtualBox redan installerat. Du kommer behöva en extern hårddisk eller ett USB-minne att spara din virtuella maskin på.

OBS! Nya mac-datorer med M1- och M2-processorer

VirtualBox är inte kompatibelt med Apples nya M1- och M2-processorer. Det finns alternativ som ska fungera med nyare mac-datorer, exempelvis [UTM](#) eller [VMware Fusion Player](#).

Tänk på att dessa kan kräva Ubuntu för ARM som kan laddas ner här:

<https://cdimage.ubuntu.com/jammy/daily-live/current/jammy-desktop-arm64.iso>

Vi har ingen tillgång till nya mac-datorer med M1/M2 processorer och kan därför inte testa dessa lösningar.

1.1.1 Installera VirtualBox

1. Gå till www.virtualbox.org.
2. Ladda hem den VirtualBox-binärfil som hör till ditt OS (*Windows, MacOS X eller Linux*).
3. Gå till platsen där filen laddades hem och dubbel-klicka på den för att starta VirtualBox-installationen.
4. Följ instruktionerna i installationsdialogerna.
5. Vänta på att installationen blir klar.

1.1.2 Installera Xubuntu

När VirtualBox är installerat kan en virtuell maskin skapas.

1. Gå till <https://xubuntu.org/>
2. Ladda hem den senaste distributionen eller senaste LTS-versionen av Xubuntu 64-bitars version (`xubuntu-XX.xx-desktop-amd64.iso`).
3. Starta VirtualBox.
4. Klicka 'New'. Detta öppnar en guide som hjälper dig att skapa en virtuell maskin.
 - Välj 'ISO Image', hitta filen du laddade hem i tidigare steg (`xubuntu-XX.xx-desktop-amd64.iso`). *I tidigare versioner av VirtualBox väljer du ISO-fil när du startar upp maskinen för första gången.*
 - Kryssa i 'Skip Unattended Installation'.
 - Sätt mängden RAM till 2 GB. Ifall du har mycket RAM-minne på din dator kan du välja 3-4 GB, men inte mer än hälften av din dators totala RAM-minne.
 - Skapa en virtuell hårddisk, 30 GB. *I tidigare versioner av VirtualBox väljer du VDI och Dynamically allocated.*
5. Starta din VM.
6. Följ installationsguiden för Xubuntu.

Efter installationen kan du starta din VM (genom att dubbel-klicka på din VM i den vänstra menyn på huvudsidan i VirtualBox). Din VM kommer att starta och ladda in det OS som är installerat.

Innan du börjar arbeta med din VM, spara en *snapshot*. En snapshot är en ögonblicksbild av hur din VM ser ut just nu. Du kan senare återgå till det ögonblicket och börja arbeta med en helt ren installation av Xubuntu. Detta är väldigt användbart för att testa mjukvara eller olika konfigurationer.

Mer information om snapshots finns på:

<https://www.virtualbox.org/manual/UserManual.html#snapshots-take-restore-delete>

2 Introduktion till GUI

Moderna Linuxdistributioner installeras som standard med en skrivbordsmiljö (*‘desktop environment’*). En skrivbordsmiljö bestämmer utseende och funktion hos bland annat fönster, menyer och skrivbordet. Vissa är väldigt avancerade och påminner om hur till exempel Windows och MacOS ser ut, medan andra är minimalistiska och erbjuder enbart några få funktioner. Några av de mer populära skrivbordsmiljöerna är GNOME, KDE, Unity och Xfce. Xubuntu använder Xfce. Det går att installera flera skrivbordsmiljöer på en maskin. Om du installerat flera skrivbordsmiljöer kan du välja vilken du vill använda vid inloggning. För denna kurs går det bra att använda sig av Xfce.

2.1 Bekanta dig med GUI

Starta din VM och logga in i skrivbordsmiljön. Bekanta dig med hur saker ser ut:

- Prova ikonerna och menyerna som finns på skrivbordet.
- Byt skrivbordsbakgrund.
- Se till att du kan arbeta med fönster (öppna, stänga, ändra storlek, etcetera).
- Starta olika applikationer.
- Använd filhanteraren (*‘file manager’*).
- Öppna en terminal (finns olika, till exempel *‘Terminal’*).

Du kan få hjälp genom hjälp-menyn i olika fönster eller genom att trycka på **F1**.

Prova att byta mellan virtuella terminaler genom att trycka på **Ctrl** + **Alt** + **F1**, **Ctrl** + **Alt** + **F2**, ..., **Ctrl** + **Alt** + **F7**. Logga in på en av terminalerna. Se till att du förstår vad som händer när man byter mellan olika virtuella terminaler. Vad är en virtuell konsol?

Hitta och öppna *‘Keyboard’*-applikationen. Ta reda på kortkommandot för att starta en terminal (*‘Launch Terminal’*).

3 Introduktion till Terminalen

Nu när du har bekantat dig med GUI:et kommer vi att istället fokusera på ett annat sätt att arbeta med en Linux-dator. Terminalen låter dig att konfigurera och arbeta med datorn enbart genom att skriva text-kommandon. Det finns både för- och nackdelar med att jobba grafiskt kontra att jobba med textkommandon. *I denna kurs kommer vi primärt jobba mot terminalen.* Vi börjar med att titta på grundläggande användning av terminalen.

Tips: Ofta används ord som kommandoprompt (*‘command prompt’*), CLI (*‘command line-interface’*) och *‘console’* som synonymer till terminal.

När man använder terminalen skriver man in kommandon, som kan vara allt ifrån ett par tecken (**ls**) till långa (**sudo timedatectl set-timezone Europe/Stockholm**). För att köra (även kallat exekvera) ett kommando, starta en terminal, skriv in kommandot och tryck sedan på Enter.

1. Öppna en terminal.
2. Skriv in kommandot **ls**.
3. Tryck på Enter.

3.1 Hitta hjälp och information

Linux har inbyggda hjälpsystem och det är viktigt att du lär dig hur du hittar hjälp och information via dessa. Användbara kommandon för detta är `man`, `info` och `apropos`. Du kan få information om kommandon genom att i en terminal skriva `man info` eller `info man`. Dessutom kommer kommandot `apropos 'system information'` ge en lista av manualsidor som innehåller texten '*system information*' i dess beskrivning.

Mycket hjälp hittar du i Linuxsystem självt, men det finns också väldigt bra hjälp på internet som du hittar via sökmotorer (till exempel Google eller Bing). Några användbara länkar:

- <http://www.tldp.org>
- <http://www.linux.org>
- <http://www.justlinux.com>
- <https://wiki.linuxquestions.org>

Tips: Använd er av de inbyggda hjälpsystemen i Linux när ni arbetar med laborationerna i denna kurs. Om ni fastnar och inte hittar svaret kan ni söka vidare på internet. Till en början kan det verka krångligare, men om man lär sig att använda de inbyggda kommandona är det hjälpsamt dels när man sitter utan ett enkelt sätt att söka på internet, dels går det i många fall också snabbare.

1. Hitta ett lämpligt kommando för att ta reda på systeminformation. Hur kan du använda detta kommando för att ta reda på namnet ('*hostname*') hos din dator?
2. Hur kan du, via ett kommando, ta reda på vilken kernelversion ('*kernel release*') som körs just nu på din dator?
3. Hur kan du söka efter en sträng i en manualsida ('*manual page*', sida man får via `man`)? Hur kan du hitta nästa och föregående förekomst av den strängen? Kan du söka efter en sträng som innehåller mellanslag?

3.2 Kommandohistoriken och kommandoradseditering

Kommandohistoriken ('*command history*') är ett användbart verktyg som sparar ett fördefinierat antal kommandon och låter användaren återanvända tidigare kommandon. Verketet består av två komponenter: en fil som heter `.bash_history` och kommandot `history`. Tillsammans bildar de ett avancerat kommandoradsediteringsverktyg.

Skriv in och exekvera följande kommandon (du behöver inte förstå den output du får):

- `ls -a`
- `cat /etc/hosts`
- `pwd`
- `uname`
- `id`
- `echo $USER`
- `help`
- `df`

1. Kör `history`. Vad får du för output?
2. Du kan gå igenom historiken genom piltangenterna. Använd `↑` och ändra `echo $USER` till `echo $HOSTNAME`. Prova även följande kortkommandon när du ändrar på kommandoraden: `ctrl+a` och `ctrl+e`. Vad gör de?
3. Du kan söka i historiken genom att först trycka `ctrl+r`, och sedan strängen du vill leta efter, en bokstav i taget. Vad händer när du trycker `ctrl+r` en gång till när du hittat kommandot du letade efter?
4. Kör `!!`. Förklara vad kommandot har för funktion.

Tips: Öppna manualsidan för `BASH` och sök efter "HISTORY EXPANSION"

5. Vad gör kommandot `!cat`?
6. Öppna manualsidan för `BASH` och leta efter variablerna `HISTSIZE` och `HISTFILESIZE`. Vad är skillnaden på de två variablernas funktion?

3.3 Alias

Du kan skapa alias, ett slags kortkommando, för vanligt använda kommandon genom att använda kommandot `alias`. Observera att ifall du skapar ett alias med `alias`-kommandot är det inte permanent, utan finns endast kvar tills du stänger din terminal. För att göra bestående alias kan du lägga in `alias`-kommandon i `.bashrc`-filen som ligger i din hemkatalog ('*home directory*'). Du kan se innehållet i din `.bashrc`-fil genom att skriva `cat .bashrc` i terminalen.

Tips: Titta i `ALIAS`-sektionen i manualsidan för `BASH` för att hitta information.

Börja med att köra `alias` för att lista dina nuvarande alias.

1. På många Linux-system skriver `ls` ut alla filer i kolumner utan någon extra information. Prova att köra `ls` och sedan `ls -l -p --color=auto` och jämför output.
2. Undersök vad flaggorna och parametrarna har för funktioner (det vill säga `-l`, `-p`, `--color=auto`).
3. För att enkelt kunna få den mer utförliga outputen vill vi ha ett alias för det. Skapa ett alias `lx` för det längre kommandot. Prova att köra `lx` och se att det fungerar.
4. Vad händer om ett alias har samma namn som ett kommando som redan finns? Prova att skapa ett alias som heter `ls`, till exempel för det längre kommandot. Om du nu kör `ls`, vad får du för output?
5. Kör kommandot `\ls`, vad får du för output?
6. Ta bort aliaset som heter `ls`. Se till att du får samma output som du fick när du körde `ls`-kommandot i början av övningen.

4 Tab-completion

En användbar teknik för att förenkla när man skriver kommandon och filnamn är att använda sig av automatisk komplettering ('*tab-completion*').

1. Skriv `as` i terminalen utan att trycka `Enter`. Tryck sedan på `↵`/`→` (TAB-tangenten) två gånger. Vad händer?
2. Skriv `cat` följt av ett mellanslag och tryck sedan på `↵`/`→` två gånger. Vad händer?
3. Byt arbetskatalog till `/bin` genom att köra kommandot `cd /bin`. Upprepa sedan steg 1 och 2 i denna övning.
4. I katalogen `/bin`, prova att skriva `cat x` där `x` är första bokstaven i något filnamn i katalogen. Prova därefter att trycka `↵`/`→` två gånger. Vad händer?
5. Skriv delar av ett filnamn tills det blir unikt, till exempel `sle`. Tryck sedan `↵`/`→`. Vad händer?

5 Miljövariabler

Miljövariabler är globala platshållare för vanligt använda argument och konfigurationsinställningar till applikationer och kommandon. Systemet återskapar automatiskt miljövariabler när ett nytt skal öppnas och återställer dess värden. Du kan få en lista över alla miljövariabler (det vill säga exporterade variabler) genom kommandot `printenv`. Du kan även använda det till att skriva ut värdet på miljövariabler.

Tips: `echo` kan användas för att skriva ut värdet av miljövariabler. För att visa värdet på en variabel används dollartecken (`$`) framför namnet på variabeln när den ska skrivas ut.

1. Vad är värdet av variabeln `PWD`? Hur kan du skriva ut värdet av `PWD` utan att använda kommandot `pwd`?
2. Förklara funktionen/betydelsen av följande miljövariabler: `HOME`, `PATH`, `USER` och `SHELL`.
3. Hur lägger du till din hemkatalog (`/home/<yourusername>/bin`) till variabeln `PATH`? Använd `$HOME` (eller `~`) för att hänvisa till din hemkatalog.

Tips: Elementen i `PATH`-variabeln är separerade med kolon (`:`).

6 Typer av kommandon och var de lagras

Det finns flera olika typer av kommandon. Externa kommandon kan dessutom vara placerade i olika sökvägar.

Använd kommandona `type`, `help`, `man` och sektionen `SHELL BUILTIN COMMANDS` i manualsidan för `Bash` för att undersöka följande åtta kommandon:



<code>ls</code>	<code>cd</code>	<code>mkdir</code>	<code>cp</code>	<code>chown</code>	<code>chmod</code>	<code>df</code>	<code>umask</code>
-----------------	-----------------	--------------------	-----------------	--------------------	--------------------	-----------------	--------------------

1. Vad är skillnaden mellan kommandon inbyggda i skalet (interna kommandon) och externa kommandon?
2. Vilka av de åtta kommandona är inbyggda?
3. Var är externa kommandon normalt placerade (vilken katalog)?

DVA249/DVA267 Linux, HT2023

- Quiz 1 -

Denna uppgift genomförs **individuellt och lämnas in genom att göra quiz "Laboration 1" 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.

1. Skriv ett kommando för att visa manualsidan för `ls`.
2. Skriv ett kommando som skriver ut värdet på miljövariabeln `PATH`.
3. Är `alias` ett internt eller externt kommando?
4. I kommandot `ls -l -p --color=never /usr/bin`, vad är argumentet/argumenten?
5. I kommandot `ls -l -p /usr/bin`, vad är flaggan/flaggorna?
6. Skapa ett alias `visa` som skriver ut värdet på variablerna `PATH` och `HOME`.
7. Om du skriver in `egr` i terminalen och sedan trycker på /, vilket kommando fylls i?

Quizet anses vara klar när du har 10 poäng på uppgiften i Canvas.

DVA249/DVA267 Linux, HT2023 - Inlämningsuppgift 1 -

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.

Ta en screenshot på er virtuella maskin. Bilden ska tydligt visa en virtuell maskin installerad i VirtualBox eller i någon annan virtualiseringsprogramvara. I den virtuella maskinen ska ett terminalfönster vara öppet.

DVA249/DVA267 Linux, HT2023

- Laboration 2 -

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 Filhantering i det grafiska gränssnittet

I denna laboration kommer vi titta på grunderna i Linux filsystem. Målet med uppgiften är att förstå den hierarkiska strukturen hos filsystemet, lära dig hur du navigerar, skapar/tar bort/döper om filer och kataloger och vart viktiga filer finns. Vi börjar med att titta på det via det grafiska gränssnittet.

1.1 Filer och kataloger

Starta en filhanterare ('*file manager*') via GUI och hitta din hemkatalog ('*home directory*'). Sökvägen till din hemkatalog är oftast `/home/<yourusername>`. Om ditt användarnamn (namnet du använda för att logga in) är `abc123` skulle din hemkatalog vara `/home/abc123`.

1. Undersök vilka kataloger som finns i din hemkatalog.
2. Använd sökfunktionen i filhanteraren och sök efter följande kataloger eller filer: `local` och `messages`. Hur kan du se ifall det är en katalog eller vanlig fil?

OBS! Du kan behöva starta din sökning i en annan katalog än din hemkatalog, till exempel i root-katalogen (`/`).

3. Gå tillbaka till din hemkatalog. Skapa en ny fil med namn `file1`. Skapa även en ny katalog i din hemkatalog och döp den till `testDirectory`. Kopiera filen du skapade (`file1`) till `testDirectory`. Markera sedan filen `file1` i `testDirectory`, kopiera den och klistra in den i samma katalog. Vad händer?
4. Ta bort alla filer i katalogen `testDirectory` och ta sedan bort katalogen `testDirectory`.
5. Som standard visar inte filhanteraren dolda eller backup-filer. Hitta ett sätt att (i filhanteraren) permanent visa dessa filer. Hur gör du det?

2 Filhantering i kommandotolken

Nu ska vi utforska hur filhantering fungerar i kommandotolken via kommandon.

2.1 Bläddra i filsystemet

Använd kortkommandot (med tangentbordet) för att öppna en ny kommandotolk. Kör kommandot `ls`. Du ser filerna i din hemkatalog och det bör vara samma filer som du såg i den grafiska filhanteraren.

1. Prova nu att köra `ls` med flaggan `-a`, sedan med flaggan `-l` och till sist med flaggan `-F`. Förklara vad de olika flaggorna har för funktioner.

Tips: Du kan hitta information om kommandon med kommandona `info` och `man`.

2. Experimentera med kommandona `cd` och `pwd` i kommandotolken och beskriv vad de används till.
3. Bläddra igenom följande underkataloger till root-katalogen (`/`). Hitta och beskriv funktionen av dessa kataloger. Vilka typer av filer är sparade i de olika katalogerna?

<code>/home</code>	<code>/usr</code>	<code>/var</code>	<code>/tmp</code>	<code>/etc</code>
--------------------	-------------------	-------------------	-------------------	-------------------

3 Filanvändning

1. Skapa en katalog `lab3test` i din hemkatalog. Vilket kommando använde du? Vilka bokstäver och symboler kan och kan inte användas i ett filnamn?
2. Byt nu din arbetskatalog till `lab3test` och skapa en fil vid namn `lab4test.txt`. Hur kan du skapa en tom fil (utan att använda dig av en texteditor)?
3. I katalogen `lab3test`, skapa två underkataloger `subdir1` och `subdir2`.
 - (a) Vad är den absoluta sökvägen till `subdir1`?
 - (b) Vad är den relativa sökvägen till `subdir1`?
4. Alla kataloger innehåller två specialkataloger: `.` och `..`, beskriv vad de betyder och hur de kan användas.
5. När du ska hantera flera filer samtidigt kan dessa skrivas var för sig i kommandoraden. Ett mer effektivt sätt är att använda sig av globbing.
 - (a) Beskriv hur och när asterisk (`*`) och frågetecken (`?`) används.
 - (b) Ge ett exempel med kommandot `ls` och `*`
 - (c) Ge ett exempel med kommandot `ls` och `?`

3.1 Filhantering

I denna uppgift ska vi lära oss att kopiera, flytta, döpa om och ta bort filer.

OBS! Att flytta, döpa om, ta bort och kopiera filer kan allvarligt skada ditt humör. Många timmars arbete kan gå förlorat om du råkar ta bort eller skriva över filer. Var försiktig när du gör denna del av laborationen. Flaggan `-i` är en bra livlina. Skapa gärna ett alias för `cp`, `mv` och `rm` med flaggan `-i`.

1. Använd kommandot `cp` för att kopiera `lab3test.txt` till en fil med namnet `lab3test_backup.txt`
 - (a) Vilket kommando använde du?
 - (b) Läs manualsidan för `cp` och beskriv följande flaggor: `-v`, `-r`, `-p`, `-b`, `-u` och `-i`.
2. Vilka av följande kombinationer av kommandon och argument är korrekta? Beskriv varför de inkorrekta kombinationerna inte fungerar.
 - (a) `cp file1.txt file2.txt`
 - (b) `cp file1.txt file1.txt`
 - (c) `cp file1.txt file2.txt file3.txt`
 - (d) `cp file1.txt file2.txt subdir1/`
 - (e) `cp subdir1/ file2.txt`
 - (f) `cp subdir1/ file1.txt file2.txt`
 - (g) `cp file1.txt subdir1/file4.txt`
 - (h) `cp file1.txt subdir1/`
3. Hur kan du kopiera alla filer med filändelsen `.txt` till en katalog? **Tips:** Använd globbing
4. Vilket kommando kan användas för att
 - (a) *flytta* filen `file2.txt` från den aktuella katalogen till underkatalogen `subdir2`?
 - (b) *ta bort* filen `file4.txt` från underkatalogen `subdir1`?
5. Anta att du i din hemkatalog har katalogerna `dir1` och `dir2`. I `dir1` finns två filer och en underkatalog enligt Figur 1. Vad händer om du kör kommandot `mv -i dir1/* dir2/` från hemkatalogen?

```
awk03@ubuntu:~/dir1$ ll
total 16
drwxrwxr-x  3 awk03 awk03 4096 nov 23 14:52 ./
drwxr-xr-x 20 awk03 awk03 4096 nov 23 14:49 ../
-rw-rw-r--  1 awk03 awk03  108 nov 23 14:52 aa.log
-rw-rw-r--  1 awk03 awk03    0 nov 23 14:21 bb.txt
drwxrwxr-x  2 awk03 awk03 4096 nov 23 14:49 subdir/
awk03@ubuntu:~$ mv -i dir1/* dir2/
```

Figur 1: Listar innehållet i `dir1`

3.2 Filsökning

Tidigare i laborationen har du använt sökfunktionen i det grafiska gränssnittet. I denna uppgift ska du söka efter filer i terminalen istället.

1. Beskriv kortfattat funktionen av följande kommandon. Hur skiljer de sig?
 - (a) `find`
 - (b) `locate`
 - (c) `whereis`
 - (d) `which`
2. Hur kan du utföra en skiftlägesokänslig ('*case insensitive*') sökning med `find` och `locate`?

3.3 Arkivering och komprimering

Filer kan paketeras för säkerhetskopiering eller för smidigare filöverföring. Dessa arkiv kan även komprimeras för att spara diskutrymme.

I denna uppgift tittar vi på olika arkiveringstyper och komprimeringsverktyg.

1. Titta på funktionen av följande arkiveringsverktyg: `tar`, `gzip` och `gunzip`. Beskriv varje verktyg med en mening.
2. Skapa ett arkiv med `tar`-kommandot. Du kan exempelvis skapa en ny katalog, skapa filer i katalogen och sedan skapa ett arkiv som innehåller alla filer.

Tips: Använd arkiveringsverktyget i en katalog där du inte har något viktigt eftersom filer kan bli ersatta ('*overwritten*') eller borttagna vid felaktig användning.

Tips: Enligt praxis döps tar-arkiv med filändelse `.tar`. Ett tar-arkiv kan exempelvis heta `projekt_version_2.tar`.

3. Skapa en ny katalog och packa upp ('*extract*') ditt arkiv i den katalogen med hjälp av `tar`. Vilket kommando använder du?
4. Gör nu samma övning som i de föregående två stegen men denna gång ska du komprimera tar-arkivet med `gzip`.

Tips: Man brukar döpa ett komprimerat tar-arkiv med filändelse baserat på vilken komprimeringsteknik som används. Ett tar-arkiv komprimerat med `gzip` ges filändelsen `.tar.gz`, till exempel `projekt_version_2.tar.gz`.

4 Introduktion till skript

Skalskript ('*shell scripts*') är enkla program skrivna i ett tolkat programmeringsspråk som är inbyggda i ett Linuxskal. Eftersom du kommer skriva dina skript i **BASH** så bör du döpa dina filer med filändelsen **.bash** och använda dig av **BASH** (**bash**) när du kör dina skript.¹ Kör *inte* dina skript med **sh** (detta är en gammal version av **BASH** och inte samma sak). I denna laboration kommer vi enbart skapa väldigt enkla skript som består av flera kommandon som körs efter varandra.

1. Öppna en texteditor och skriv in kommandon (ett per rad) enligt Kodruta 1.

```
#!/bin/bash
echo "Första kommandot!"
echo "Kommando nummer två..."
echo "Tredje kommandot är det sista."
```

Kodruta 1: kommandon.bash

2. Fundera på vad du tror att du kommer få för output.
3. Spara filen som **kommandon.bash**
4. Prova att köra skriptet genom att köra kommandot **bash kommandon.bash**
5. Vad får du för output? Är det samma som du trodde?

Tips: Notera att skriptet börjar med en speciell rad, den så kallade '*shebang*'-raden: **#!/bin/bash**. I denna laboration behöver du inte förstå vad den betyder, men varje skript du skapar kommer att ha den raden längst upp.

Man använder sig ofta av skript för att automatisera uppgifter. Om man har ett antal kommandon som man vill köra ofta, kan man skapa ett skript och istället för att skriva in alla kommandon varje gång räcker det med att köra skriptet. I laboration 1 använde vi oss av alias för att skapa kortkommandon. Aliasen försvinner varje gång man stänger och öppnar en ny terminal. För att slippa skriva om dem kan vi skapa ett skript som skapar aliasen åt oss.

1. Skriv ned tre kommandon för att skapa tre alias **alone**, **altwo** och **althree**. Vad aliasen gör för något får du bestämma själv.
2. Skapa en skript-fil **aliases.bash** som innehåller en shebang följt av de tre kommandon som du skrivit ned, se Kodruta 2.

```
#!/bin/bash
alias alone=...
alias altwo=...
alias althree=...
```

Kodruta 2: aliases.bash

3. Öppna en ny terminal, se till att aliasen är odefinierade genom att köra dem. Kör sedan **source aliases.bash** för att definiera dem, och kolla att de fungerar som du tänkt.

Tips: För att alias ska skapas i den BASH-instans som du befinner dig i måste **source** användas istället för **bash**.

Vi kommer i senare laborationer titta på var man kan placera sina kommandon för att de ska köras automatiskt varje gång man startar en ny terminal.

¹Ifall du har installerat ett annat skal, se till att använda dig av **BASH** när du gör inlämningsuppgifterna.

DVA249/DVA267 Linux, HT2023

- Quiz 2 -

Denna uppgift genomförs **individuellt och lämnas in genom att göra quiz "Quiz 2" 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.

Vilka av följande kombinationer av kommandon och argument är korrekta?

- (a) `cp file1.txt file2.txt`
- (b) `cp file1.txt file1.txt`
- (c) `cp file1.txt file2.txt file3.txt`
- (d) `cp file1.txt file2.txt subdir1/`
- (e) `cp subdir1/ file2.txt`
- (f) `cp subdir1/ file1.txt file2.txt`
- (g) `cp file1.txt subdir1/file4.txt`
- (h) `cp file1.txt subdir1/`

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

DVA249/DVA267 Linux, HT2023

- Inlämningsuppgift 2 -

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 skript med namnet `logs.bash` som gör följande:

1. Byter arbetskatalog till hemkatalogen `/home/<username>/`. Använd `$HOME` (eller `~`) för att hänvisa till din hemkatalog.
2. Skapa en katalog med namnet `Logs` i din hemkatalog.
3. Kopiera alla filer som slutar på `.log` från katalogen `/var/log/` till `Logs`-katalogen.
4. Skapa ett komprimerat tar-arkiv som innehåller filerna i `Logs`-katalogen. Arkivet ska sparas i `Logs`-katalogen.
5. Ta bort alla filer som slutar på `.log` i `Logs`-katalogen.
6. Lista innehållet i `Logs`-katalogen.

Skriptet ska använda den **relativa sökvägen** till `/home/<username>/Logs/` och den **absoluta sökvägen** till `/var/log/`. Använd skriptet i Kodruta 3 som grund för ditt skript. Skriv klart skriptet enligt punkterna ovanför.

```
#!/bin/bash
# Kommandorad nr 1
# Kommandorad nr 2
# Kommandorad nr 3
echo '-----'
# Kommandorad nr 4
# Kommandorad nr 5
echo '-----'
# Kommandorad nr 6
```

Kodruta 3: `logs.bash`

Kör ditt skript genom att ange skriptet som argument till kommandot `bash`. Figur 2 visar ett exempel på det förväntade resultatet.

OBS! Antalet `log`-filer och vilka ni har rättigheter att kopiera kan skilja sig mellan olika Linux-maskiner. Därför är Figur 2 endast ett exempel.

```
cp: cannot open '/var/log/boot.log' for reading: Permission denied
-----
<Eventuell output fr n kommandorad 4 eller 5>
-----
logs.tar.gz
```

Figur 2: Exempel output

Uppgiften anses vara klar när du har 1 poäng på uppgiften i Canvas. Deadline för uppgiften är 17/11 kl 23:59.

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) Lägg 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 dubletter.
 - (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 operatoren 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-formatting.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 kommandossubstituering (*'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 variabeln 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 testat 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)
7-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.

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
<code>-a, --all</code>	Skriver ut all information
<code>-v, --version</code>	Skriver ut Linuxversion
<code>-i, --ip</code>	Skriver ut IP-adressen
<code>-m, --mac</code>	Skriver ut MAC-adressen
<code>--help</code>	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.

DVA249/DVA267 Linux, HT2023

- Laboration 4 -

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 Processhantering

När ett kommando körs i kommandotolken skapas en process för programmet som körs. Varje process i Linux har ett process-id (PID). Det är viktigt att förstå hur processer fungerar och hur vi kan hantera processerna i ett system.

1.1 Information om processer

1. Genom att starta **Task Manager/System Monitor** i GUI kan alla processer som körs på din maskin listas. Bekanta dig med detta program.
 - (a) Hur kan du sortera listan efter processernas CPU-användning? Den process som använder mest CPU hamnar högst upp.
 - (b) Kan du döda ('*kill*') en process från **Task Manager/System Monitor**? Om ja, hur?
2. Ett av de viktigaste verktygen när det kommer till processhantering i terminalen är **ps**.
 - (a) Hur kan alla processer listas med kommandot **ps**?
 - (b) Hur kan alla processer för en specifik användare listas?
3. Lista alla processer med verktyget **top**.
 - (a) Hur kan du sortera listan efter processernas CPU-användning?
 - (b) Hur kan du sortera listan efter processernas minnesanvändning?
4. Vilket kommando ritar upp alla processer i en trädstruktur?
5. Starta två terminalfönster. Starta **xterm** genom att köra kommandot **xterm** i det ena terminalfönstret. Ta reda på process-id för **xterm** i det andra terminalfönstret (använd **ps** och **grep**).

Info: **xterm** är en minimalistisk terminal med färre funktioner än standardterminalen.

Tips: Titta även på kommandot **pidof**.

6. Starta **xterm** och anteckna process-id, stäng **xterm** och starta **xterm** igen. Fick **xterm**-processen samma process-id? Varför/Varför inte?

Tips: Det är nu möjligt att förklara funktionen av `export`-kommandot. Om man exporterar en variabel `VAR` i en terminal, så kommer alla underprocesser till terminalen (det vill säga alla processer som startas av terminalen) att få en kopia av `VAR` i sin miljö. Observera att värdet av variabeln kopieras när underprocessen startas men kommer inte uppdateras ifall den senare ändras i ursprungsterminalen.

Tips: I hemkatalogen finns en fil `.bash_rc` som körs varje gång man startar ett nytt skal. Till exempel, ifall du vill ha ett alias definierat kan du skriva `alias ll='ls -a1F'` i din `.bash_rc`-fil så kommer aliaset alltid vara tillgängligt i dina skal.

1.2 Bakgrundsprocesser

1. Hur kan du starta ett program (till exempel `xterm`) som en bakgrundsprocess?
2. Starta fyra `xterm`-processer i bakgrunden (som bakgrundsprocesser).
 - (a) Kör kommandot `jobs` (internt kommando) och beskriv vad kommandot används för.
 - (b) Hur kan informationen från `jobs` användas med kommandona `fg`, `bg` och `kill`?

Tips: Titta i manualsidorna för `BASH` (`JOB CONTROL`).

3. Starta ett program som en förgrundsprocess från terminalen.
 - (a) Hur kan du stoppa processen (*'suspend the process'*) utan att döda den?
 - (b) Hur kan du flytta processen till bakgrunden?
 - (c) Hur kan du flytta processen till förgrunden igen?

1.3 Att döda processer

Normalt stängs program med applikationernas inbyggda stängningsfunktion, exempelvis `close` eller `exit`. Hur kan vi tvångsstänga ett program som låser sig och inte tar emot någon input? Vi kan döda processer genom att skicka olika signaler till program som låst sig.

1. Stäng alla terminalfönster och öppna två nya terminalfönster, terminal 1 och terminal 2. Starta `xterm` i terminal 1. Anta nu att `xterm` har låst sig och slutar svara på input från användaren. Hur kan du döda `xterm`-processen från terminal 2?
2. Kör kommandot `sleep 20` i ett nytt terminalfönster. Kan du arbeta i terminalen under tiden programmet körs? Varför/Varför inte?
3. Starta nu tre instanser av `xterm` som bakgrundsprocesser. Hur kan du döda alla instanser av `xterm` med ett kommando?

Tips: Titta på kommandot som kan döda processer med hjälp av namn istället för process-id.

4. Det finns flera signaler som kan skickas till program med kommandot `kill`. Beskriv kortfattat följande signaler:

- `SIGHUP` (1)
- `SIGINT` (2)
- `SIGQUIT` (3)
- `SIGKILL` (9)
- `SIGTERM` (15)

2 Pakethantering

En pakethanterare är ett system som kan användas till att installera, uppdatera och ta bort programvara från ett filsystem. Ett paketbibliotek (*'repository'*) innehåller paketerad programvara som är kompilerad för din Linuxdistribution. Genom att använda verktygen i terminalen kan paket från paketbiblioteket installeras. För Redhat används `yum` och för Debian-baserade distributioner används `apt`, `apt-get` och `dpkg`. Paket kan även installeras genom GUI.

1. Starta applikationen **Software** i GUI. Bläddra igenom listan av installerade applikationer och bekanta dig med programmet. Sök efter applikationen **Calendar** och installera den. Avinstallera sedan applikationen **Calendar**.
2. Öppna ett terminalfönster och uppdatera installerade paket genom att köra kommandona `sudo apt update` och `sudo apt dist-upgrade`.

Tips: Om `apt` eller `apt-get` används i skript kan frontend för `apt` bytas till ett icke-interaktivt frontend och svara `yes` på alla frågor. Använd då `DEBIAN_FRONTEND=noninteractive apt dist-upgrade -y`.

Läs manualsidan för `apt` och `apt-get`. Läs även avsnittet om *Frontends* i manualen för **debconf (7)**. Om `debconf-doc` inte är installerad hittar du manualsidan här: <https://manpages.debian.org/jessie/debconf-doc/debconf.7.en.html>

3. Öppna ett terminalfönster och använd kommandot `apt` för att installera `sl`.
 - (a) Vilket kommando använde du?
 - (b) Vad gör `sl`-applikationen?

Tips: Läs manualen för `sl`

4. Installera `aptitude` med hjälp av `apt`. Du kan nu välja att installera program med `apt` eller `aptitude`.

Tips: Mer information om pakethanterare i Debian-baserade distributioner finns här: <https://www.debian.org/doc/manuals/debian-faq/pkgtools.en.html>

3 Installera program från källkod

En pakethanterare installerar paketerad programvara som är kompilerad för din Linuxdistribution. Men paketbiblioteket kanske inte har den senaste versionen av ett program för den Linuxdistribution du använder. Om den senaste versionen av programmet krävs måste programmet kompileras och installeras från källkod. Samma sak gäller med egenutvecklad programvara som kräver kompilering, exempelvis program skrivna i C eller C++.

Ett av de mest använda verktyget som automatiskt bygger program och kodbibliotek från källkod är Make-verktyget. Make använder så kallade *makefiles* för att automatisera processen.

Tips: Användbar information om kompilering av programvara för Linux från källkod finns här: <https://www.control-escape.com/linux/lx-swinstall-tar.html>

3.1 Kompilera och installera

I denna uppgift ska du installera *ifstat* från källkod (inte via *apt*). Verktygen *make* och *gcc* måste vara installerade innan du börjar.

Installera *ifstat* från källkod genom att följa stegen nedan.

1. Installera verktygen *make* och *gcc* om de inte redan är installerade.
2. Använd verktyget *wget* för att ladda ner arkivet med källkodsfilerna. Använd följande länk: http://ftp.de.debian.org/debian/pool/main/i/ifstat/ifstat_1.1.orig.tar.gz
3. Packa upp arkivet med hjälp av *tar*-kommandot.
4. Gå in i katalogen med de uppackade filerna. Titta igenom katalogen och läs filerna *INSTALL* och *README*.
5. Genom kommandot *./configure* konfigureras utvecklingsmiljön och *makefiles* skapas.
6. Genom kommandot *make* kompileras all källkod. När den är klar finns det körbara verktyget *ifstat* i katalogen.
7. Kör kommandot *sudo make install* för att flytta de nödvändiga filerna till rätt ställe i filsystemet.
8. Svara på följande frågor:
 - (a) I vilken katalog lägger *make ifstat*?
 - (b) Vad gör verktyget *ifstat*?

DVA249/DVA267 Linux, HT2023

- Quiz 4 -

Denna uppgift genomförs **individuellt och lämnas in genom att göra quiz "Quiz 4" 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.

Användaren **awk03** kör kommandot **ps au** för att lista processer. Output för detta kommando visas i Figur 1.

```
awk03@xubuntu:~$ ps au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        1559  0.0  0.0   5828  1732 tty1      Ss+  Nov07    0:00 /sbin/agetty -o
-p -- \u --noclear tty1 linux
systemd+   1727  0.5  9.7 1314048 389816 pts/0    Ssl+ Nov07    6:29 mysqld --
character-set-server=utf8 --collation-server=utf8_bin --default-authentication
-plugin=mysql_n
awk03      256555  0.0  0.1   8276   5152 pts/0      Ss   11:12    0:00 -bash
awk03      256933  0.0  0.1   8284   5132 pts/1      Ss+  11:13    0:00 -bash
awk03      256944  0.0  0.1   8284   5168 pts/2      Ss   11:13    0:00 -bash
awk03      256989  0.8  0.1   8580   4532 pts/2      S+   11:14    0:27 htop
awk03      267334  0.0  0.0   9128   3648 pts/0      R+   12:05    0:00 ps au
```

Figur 1: Output från kommandot **ps aux**

Fyll i det som saknas i kommandot nedan för att döda processen **htop**?

awk03@xubuntu:~\$ kill fyll i det som saknas

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

DVA249/DVA267 Linux, HT2023

- Inlämningsuppgift 4 -

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 `hit_and_run.bash`. Användaren ska ange vilket program som ska installeras, hur många bakgrundsprocesser av programmet som ska köras och hur många sekunder processerna ska köras.

Skriptet ska ta tre argument och körs med följande kommando:

```
sudo bash hit_and_run.bash <program> <antal processer> <antal sekunder>
```

Använd kodskelettet i Kodruta 1 nedan. Byt ut kommentarerna med din kod/kommandon.

```
#!/bin/bash
if # $1 inte är installerad
then
    echo "Installning $1"
    #uppdatera installerade paket
    #installera $1
fi

echo "$1 installed"
echo "Running $2 processes of $1 for $3 seconds in xterm"

for #(( $2 antal processer ))
do
    #Kör programmet i ett xterm-fönster som bakgrundsprocess
done

sleep $3

#döda alla xterm-processer

echo -n "Uninstall $1? (y/n)"
read val

if [[ "$val" == "y" ]]
then
    echo "Uninstallning $1"
    #avinstallera $1
fi
```

Kodruta 1: Kodskelett `hit_and_run.bash`

Användbara kommandon är `dpkg`, `apt`, `apt-get`, `kill`, `killall`, `pidof` och `ps`. Titta även på flaggan `-e` för `xterm`.

Några exempel på program att testa ditt skript med är `cmatrix`, `htop` och `bpytop`. Exempel på output från skriptet visas i Figur 2.

```
awk03@xubuntu:~$ sudo bash hit_and_run.bash bpytop 2 20
[sudo] password for awk03:
Installing bpytop
bpytop installed
Running 2 processes of bpytop for 20 seconds in xterm
Uninstall bpytop? (y/N) y
Uninstalling bpytop
```

Figur 2: Exempel på output från `hit_and_run.bash`

Tips: För "renare" output från skriptet kan ni omdirigera STDIN och STDERR till `/dev/null` när kommandon körs. `/dev/null` är ett svart hål, en speciell fil som endast kan skrivas till och finns på alla Linux-system. Allt som skrivs till `/dev/null` hamnar i det svarta hålet och försvinner.

Uppgiften anses vara klar när du har 1 poäng på uppgiften i Canvas. Deadline för uppgiften är 1/12 kl 23:59.

DVA249/DVA267 Linux, HT2023

- Laboration 5 -

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 Filrättigheter, användare, grupper och länkar

I Linux styrs åtkomsten till filer, kataloger och kringutrustning med användare och grupper. Vem som äger filerna är avgörande för säkerheten. Vanligtvis ägs filen av den användaren som skapade filen. För att ändra filägare krävs administrativa rättigheter.

Kommandot **sudo** kör ett kommando med **root**-rättigheter. När du använder **sudo** innan ett kommando frågar systemet om ditt lösenord innan den kör kommandot som användaren *root*.

Användaren måste vara medlem i gruppen *sudo* för att använda **sudo**. En fördel med att använda kommandot **sudo** istället för att byta användare till *root* är att endast ett kommando exekveras och återgår sedan till din användare. Detta förhindrar bland annat oavsiktlig borttagning och ändring av viktiga systemfiler.

1.1 Hantera länkar

En länk är ett sätt att göra en fil mer tillgänglig, exempelvis genom att ge den fler identiteter eller ge den flera olika namn. Det finns två typer av länkar, symboliska länkar (*'soft link/symbolic link'*) och hårda länkar (*'hard link'*).

1. Börja med att skapa två kataloger i din hemkatalog, **lab5test** och **subdir1**. Skapa sedan en ny fil med namnet **file1.bash** i katalogen **subdir1**.

Skapa sedan en symbolisk länk i katalogen **lab5test** till filen **file1.bash** i **subdir1**-katalogen. Använd filnamnet (**file1.bash**) som länknamn. Använd **ls** för att säkerställa att länken skapades.

2. Vad händer om du tar bort **file1.bash** från **subdir1**? Kan filen öppnas via den symboliska länken i katalogen **lab5test**? Förklara varför eller varför inte.
3. Skapa en ny fil med namnet **file2.txt** i katalogen **subdir1**. Skapa nu en hård länk i katalogen **lab5test** till filen **file2.txt** i **subdir1**. Använd filnamnet som länknamn.
4. Vad händer om du tar bort **file2.txt** från **subdir1**? Kan filen öppnas via den hårda länken i katalogen **lab5test**? Förklara varför eller varför inte.
5. Det går även att skapa symboliska länkar till kataloger. Skapa en symbolisk länk till **/var/log** med namnet **logs** i din hemkatalog. Vilket kommando använde du?
6. Testa nu att skapa en hård länk till en katalog. Vad händer?

1.2 Användare och grupper

1. Testa kommandot `id`. Läs manualen för `id`.
 - (a) Med vilken flagga för `id`-kommandot kan du få fram ditt UID-nummer (*'användar-id/user identification'*)
 - (b) Med vilken flagga för `id`-kommandot kan du få fram en lista med namnet på alla grupper som du är medlem i?
2. Titta nu på filen `group` i ditt filsystem, använd `less /etc/group`
 - (a) Kan du hitta några likheter med `id`-kommandot?
 - (b) Vilket kommando (med pipe) kan du använda för att endast visa rader som inkluderar ditt användarnamn?
3. Titta nu på filen `passwd` i ditt filsystem, använd `less /etc/passwd`
 - (a) Vad innehåller filen?
 - (b) Titta på raden som innehåller ditt användarnamn. Beskriv vad de olika fälten (kolon-separerade) innehåller.
4. Skapa ett användarkonto för *Alice Stone* genom att använda kommandot `useradd` med `sudo`.
 - Användarnamn (LOGIN): `alisto`
 - Kommentar (COMMENT): `Alice Stone`
 - Hemkatalog (HOMEDIR): `/home/alisto`
 - Skal (SHELL): `/bin/bash`
 - Sätt lösenord på kontot.

Vilket/Vilka kommandon använde du?

Tips: Kommandot `su --login användarnam` byter användare till den användare som anges. Läs manualsidan för `su` för mer information.

5. Skapa ett användarkonto för *Bob Anderson* genom att använda kommandot `adduser` (*inte useradd*) med `sudo`.
 - Användarnamn (LOGIN): `boband`
 - Kommentar (COMMENT): `Bob Anderson`
 - Hemkatalog (HOMEDIR): `/home/boband`
 - Skal (SHELL): `/bin/bash`
 - Sätt lösenord på kontot.

Vilket/Vilka kommandon använde du?

6. Skapa en ny grupp med namnet *solarproj* och lägg till Alice och Bob som medlemmar i gruppen. Vilka kommandon använde du?

Användarna *Alice* och *Bob*, och gruppen *solarproj* kommer användas i nästa del av laborationen.

1.3 Ägare och rättigheter

1. Skapa ett skript med namnet `1.3-listing.bash` enligt Kodruta 1.

```
#!/bin/bash
echo
echo Showing content of directory $(pwd)
echo -----
ls
```

Kodruta 1: `1.3-listing.bash`

Ändra rättigheterna på filen `1.3-listing.bash`

- (a) till endast läs-, skriv- och exekveringsrättigheter för användare (`-rwx-----`). Vilket kommando använde du?

Tips: Genom att sätta exekveringsrättigheter på filen som innehåller ett bash-skript behöver vi inte längre ange skriptet som argument till `bash`. Kör skriptet genom att skriva `./1.3-listing.bash`

- (b) till läs- och skrivrättigheter för användaren, endast läsrättigheter för gruppen och inga rättigheter för andra (`-rw-r----`). Vilket kommando använde du?
2. Ändra rättigheterna på filen `1.3-listing.bash` till endast läsrättigheter för användaren och ta bort rättigheter för grupp och andra (`-r-----`).
 - (a) Går det att skriva ut filen `1.3-listing.bash` på skärmen med `cat`? Varför/varför inte?
 - (b) Går det att editera filen `1.3-listing.bash` och spara ändringarna? Varför/varför inte?
 3. Beskriv hur kommandot `umask` kan användas för att ge nya filer läs- och skrivrättigheter för användaren, endast läsrättigheter för gruppen och inga rättigheter för andra (`-rw-r---`).

Tips: Kommandot `umask` påverkar endast filer och kataloger som skapas, inte filer och kataloger som redan existerar. Värdet subtraheras från *mode value* (*default permissions*) som specificeras av program, exempelvis `touch`. Standard *mode* för filer är `666` och `777` för kataloger.

4. Skapa en katalog med namnet `projects` i root-katalogen. Skapa sedan katalogen `solarproj` i `projects`-katalogen. Resultatet ska se precis ut som i Figur 1. Vilka kommandon använde du?

```
awk03@xubuntu:/projects$ ls -al
total 12
drwxr-xr-x  3 root root    4096 Nov  28 15:29 .
drwxr-xr-x 21 root root    4096 Nov  28 15:26 ..
drwxrwsr-t  2 root solarproj 4096 Nov  28 16:09 solarproj
```

Figur 1: Listar innehållet i `projects`-katalogen.

2 Nätverk och fjärråtkomst

De flesta Linux-system är anslutna till ett nätverk, antingen som en klient eller en server. Det är därför viktigt att ha grundläggande kunskap om nätverk och nätverksinställningar i Linux. Det är även viktigt att veta hur man fjärransluter till ett Linux-system, eftersom Linux ofta används som operativsystem på servrar.

2.1 Nätverkskonfiguration

I Linux kan nätverksinställningar göras med verktyg i GUI eller genom konfigurationsfiler i CLI. Verktygen som finns tillgängliga för dig varierar beroende på Linuxdistribution. Ubuntu/Xubuntu använder verktygen Netplan, NetworkManager och Networkd. Netplan konfigureras och skickar sedan vidare konfigurationen till antingen verktyget NetworkManager (nätverksinställningar i GUI) eller verktyget systemd-networkd (nätverksinställningar i CLI). Konfigurationen för Netplan görs via en YAML-fil i katalogen `/etc/netplan/`. Läs mer om Netplan och se konfigurationsexempel på <https://netplan.io/>.

2.1.1 Nätverkskort

I terminalen kan du använda följande kommandon för att lista alla installerade nätverkskort. Undersök kommandona och beskriv dessa kortfattat.

1. `lspci | grep -i -E 'network|ethernet'`
2. `sudo lshw -class network`

2.1.2 IP-kommandot

Kommandot `ifconfig` (Net-tools-paketet) var tidigare standard för att lista, stänga av och sätta på nätverksinterface. Kommandot `ifconfig` har ersatts av kommandot `ip` och finns inte längre installerat på nyare versioner av Ubuntu/Xubuntu.

Använd kommandot `ip` och besvara följande frågor:

1. Vilket kommando används för att lista (*'show'*) information om alla nätverksinterface?
2. Vilket kommando används för att stänga av (*'interface down'*) ett nätverksinterface?
3. Vilket kommando används för att sätta på (*'interface up'*) ett nätverksinterface?

2.1.3 NetworkManager

För Ubuntu/Xubuntu med grafiskt gränssnitt är det NetworkManager som hanterar nätverksinterface och inställningar. Netplan läser in konfigurationen i filen `/etc/netplan/01-network-manager-all.yaml` som talar om att *renderer* ska vara NetworkManager, se Kodruta 2.

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
```

Kodruta 2: `/etc/netplan/01-network-manager-all.yaml`

Med endast *version* och *renderer* konfigurerad i YAML-filen, överläter Netplan nätverkskonfigurationen till NetworkManager. Standardinställningarna i NetworkManager är att maskinen får en IP-adress tilldelad via nätverket med hjälp av DHCP.


2.1.4 Networkd

Flytta YAML-filen `/etc/netplan/01-netork-manager-all.yaml` till din hemkatalog. Då har du en backup på konfigurationsfilen ifall något skulle gå fel. Skapa en ny YAML-fil med namnet `01-networkd.yaml` i katalogen `/etc/netplan/`. Byt *renderer* till `systemd-networkd` genom att skriva konfiguration i filen enligt Kodruta 3. Maskinen får fortfarande en IP-adress tilldelad via nätverket med hjälp av DHCP.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: true
```

Kodruta 3: `/etc/netplan/01-networkd.yaml`

Applicera inställningarna genom att köra kommandot `sudo netplan apply`.

OBS! Använd inte  (TAB-tangenten) i YAML-filen. Använd endast mellanslag. Du måste skriva in namnet på ditt nätverksinterface till exempel `enp0s3` eller `ens160`. Ta reda på namnet med hjälp av `ip`-kommandot.

Titta nu på nätverksinställningarna i GUI. Ser du någon skillnad efter byte av *renderer*?

Tips: Om ingenting händer kan du behöva stänga av och sätta på nätverksinterfacet.

2.1.5 Statisk IP-adress

Serverar har oftast manuellt konfigurerade IP-adresser, så kallade statiska adresser. Detta för att servern alltid ska ha samma IP-adress. Konfigurationen görs i YAML-filen.

Ändra nu konfigurationen i YAML-filen till konfigurationen i Kodruta 4. Kör sedan kommandot `sudo netplan apply`.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      addresses:
        - 10.0.2.16/24
      nameservers:
        addresses: [1.1.1.1, 8.8.8.8]
      routes:
        - to: default
          via: 10.0.2.2
```

Kodruta 4: `/etc/netplan/01-networkd.yaml`

Säkerställ att du fick den önskade IP-adressen genom kommandot `ip address`.

OBS! Interface-namn och IP-adresser kan skilja sig beroende på system och nätverk. Konfigurationen i Kodruta 4 fungerar på Xubuntu i VirtualBox.

2.2 Fjärråtkomst

Secure Shell (SSH) är ett verktyg för att ansluta och logga in på andra datorer. SSH kan användas mellan olika operativsystem, exempelvis från Windows, Mac eller Linux till en Linux-server i molnet. Med SSH går det även att kopiera filer mellan system med kommandot `scp`. Kommandot `scp` fungerar ungefär som kommandot `cp` men från en dator till en annan dator.

1. Börja med att installera openssh-server på din maskin.
2. Hur kan du ansluta till en annan dator med `ssh`?

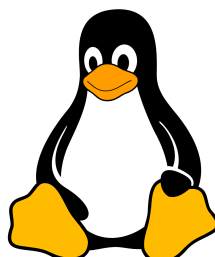
Tips: Använd loopback-adressen (*127.0.0.1* eller *localhost*) om du inte har tillgång till en annan Linux-dator.

3. Hur kan du använda `ssh` för att köra ett kommando på en annan dator utan att först logga in?
4. Hur kan du kopiera en fil från en annan dator till din dator med hjälp av kommandot `scp`?

Tips: Om du senare kommer arbeta mycket med nätverk, Linux-instanser i molnet och fjärranslutningar kan följande vara användbart.

Det går att skapa RSA-autentiseringsnycklar som kan användas för lösenordslös inloggning med SSH. Detta ingår inte i kursen, men du kan hitta information online eller i manualsidorna för `ssh-keygen` och `ssh-copy-id`.

Tips: SSH File Transfer Protocol (SFTP) är ett nätverksprotokoll för filhantering över SSH. Jämfört med `scp` har SFTP utökad funktionalitet.



DVA249/DVA267 Linux, HT2023

- Quiz 5 -

Denna uppgift genomförs **individuellt och lämnas in genom att göra quiz "Quiz 5" 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.

3 Filrättigheter

Bob (*boband*) och Alice (*alisto*) är medlemmar i gruppen *solarproj* och kommer använda katalogen *solarproj* för att tillsammans arbeta med ett mjukvaruprojekt. Figur 2 visar projektets filer (innehållet i katalogen *solarproj*).

```
awk03@xubuntu:/projects/solarproj$ ls -al
total 32
drwxrwsr-t 3 root  solarproj 4096 Nov 28 16:02 .
drwxr-xr-x 3 root  root      4096 Nov 28 15:29 ..
-rw-rw---- 1 boband boband   802 Nov 28 15:44 bearings.jpg
-rw-rw---- 1 boband solarproj 101 Nov 28 15:43 controller.c
drwxrwsr-x 2 alisto solarproj 4096 Nov 28 15:40 old
-----rwx 1 alisto solarproj  19 Nov 28 14:08 power
-rw-r--r-- 1 alisto solarproj  81 Nov 28 15:44 schedule.txt
-rw----r-- 1 boband solarproj 576 Nov 28 15:58 temp
lrwxrwxrwx 1 alisto solarproj  24 Nov 28 16:02 temp.sym -> ../solarproj/temp
```

Figur 2: Listar innehållet i *solarproj*-katalogen.

Studera filrättigheterna i katalogen *solarproj* och svara på följande frågor.

Tips: Skapa filerna, länkarna och katalogerna på din Linux-maskin. Sätt rättigheter enligt Figur 2. Använd kommandot `su --login användarnam` för att logga in som **alisto** eller **boband**.

1. Kan *Bob* ändra innehållet i filen *schedule.txt*? Varför/varför inte?
 - ☐ Ja, gruppen *solarproj* har skrivrättigheter.
 - ☐ Ja, gruppen *solarproj* har läsrättigheter.
 - ☐ Nej, gruppen *solarproj* har inga skrivrättigheter.
 - ☐ Nej, exekveringsrättigheter saknas för alla.
2. Kan *Alice* ta bort filen *controller.c*? Varför/varför inte?
 - ☐ Ja, setgid är satt på katalogen *solarproj*.
 - ☐ Ja, gruppen *solarproj* har skrivrättigheter på filen.
 - ☐ Nej, gruppen *solarproj* har inte skriv- och exekveringsrättigheter på katalogen *solarproj*.
 - ☐ Nej, Sticky bit är satt på katalogen *solarproj*.

3. Vem kan läsa filen `power`? Varför?

- ☐ Bara medlemmar i gruppen *solarproj* kan läsa filen `power`.
- ☐ Alla andra förutom ägare och medlemmar i gruppen *solarproj*.
- ☐ Ingen kan läsa filen `power`.
- ☐ *Alice* kan läsa filen eftersom hon äger den.

4. Varför kan *Alice* men inte *Bob* byta arbetskatalog till `old`?

- ☐ *Alice* har exekveringsrättigheter på katalogen `old` men gruppen *solarproj* har inte exekveringsrättigheter.
- ☐ Setgid är satt på katalogen `old`.
- ☐ Katalogen `old` tillhör gruppen *alisto*.
- ☐ Gruppen *boband* har inte exekveringsrättigheter.

5. Kan *Alice* läsa filen `temp.sym`? Varför/varför inte?

- ☐ Ja, *Alice* har läsrättigheter på `temp.sym` och på filen som länken pekar på.
- ☐ Ja, *Alice* har läsrättigheter på `temp.sym` och kan då även läsa den länkade filen.
- ☐ Nej, *Alice* kan läsa länken men har inte läsrättigheter på filen `temp`.
- ☐ Nej, *Alice* har inte läsrättigheter på `temp.sym`.

4 SSH

Alice har installerat openssh-server på en dator för att låta *Bob* fjärransluta till datorn (Xubuntu/Ubuntu).

1. Vilket kommando ska hon använda för att se status på tjänsten ('*service*') `ssh`?
2. Hur kan tjänsten ('*service*') `ssh` stoppas?
3. Hur kan tjänsten ('*service*') `ssh` startas?

Quizet anses vara klar när du har 10 poäng på uppgiften i Canvas. Deadline för uppgiften är 8/12 kl 23:59.

DVA249/DVA267 Linux, HT2023

- Inlämningsuppgift 5 -

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 `users.bash`. Skriptet ska ge information om användare beroende på vilken flagga användaren anger, se Tabell 1.

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

Option	Funktion
<code>-u, --users</code>	Listar alla användarkonton
<code>-o, --online</code>	Skriver ut alla användarkonton som är inloggade just nu
<code>--help</code>	Skriv ut information om skriptet och tillgängliga flaggor

Output till terminalen ska likna exemplet i Figur 3. 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 `user.bash -uo`.

Tips: Titta på kommandot `who` och filen `/etc/passwd`. För att skriva en snyggare lista på användare kan kommandot `column` användas. Andra användbara kommandon är `grep`, `awk` och `cut`.

OBS! Output i Figur 3 är endast ett exempel. Ert skript kommer förmodligen lista fler användare. Output från kommandot `who` varierar beroende på hur många som är inloggade och om användarna är inloggade lokalt på datorn eller fjärranslutna via SSH. IP-adressen visar från vilken adress användaren har anslutit sig. Vid inloggning lokalt visas inte IP-adressen.

```
awk03@xubuntu:~$ ./users.bash --help
Usage: users.bash OPTION...
Print users

OPTIONS
-u, --users      list all user accounts
-o, --online     List all online users
--help          display this help and exit
awk03@xubuntu:~$ ./users.bash -u
root          systemd-timesync      sshd
daemon        proxy                    anka
bin           www-data                syslog
sync          tss                      fwupd-refresh
games         irc                      uidd
man           gnats                   tcpdump
dnsmasq       nobody                  kalle
mail          systemd-network         awk03
awk03@xubuntu:~$ ./users.bash -o
awk03 pts/0      2023-12-08 10:53 (10.0.0.10)
kalle pts/1      2023-12-08 11:13 (192.168.105.8)
anka pts/2      2023-12-08 13:01 (192.168.105.65)
```

Figur 3: Exempel på output från `users.bash`

Fråga 1 | Skriv ett kommando för att visa manualsidan för `ls`.

`man ls` Alternativa (ej rätt): `Info ls, ls --help`

Fråga 2 | Skriv ett kommando som printar ut värdet på miljövariabeln `PATH`.

`Echo $PATH`

Fråga 3 | Är `alias` ett internt eller externt kommando?

Internt kommando

Fråga 4 | I kommandot `ls -l -p --color=never /usr/bin`, vad är argumentet/argumenten?

`/usr/bin`

Fråga 5 | I kommandot `ls -l -p /usr/bin`, vad är flaggan/flaggorna?

`-l -p`

Fråga 6 | Skapa ett alias visa som printar värdet på variablerna `PATH` och `HOME`.

ex: `alias echo_path_home='echo $PATH && echo $HOME'`

Fråga 7 | Om du skriver in `egr` i terminalen och sedan trycker på `TAB`-tangenten, vilket kommando fylls då i?

`egrep`

Fråga 1 | Vilka av följande kombinationer av kommandon och argument är korrekta?.

(a) `cp file1.txt file2.txt` (d) `cp file1.txt file2.txt subdir1/` (g)
`cp file1.txt subdir1/file4.txt` (h) `cp file1.txt subdir1/`

Fråga 1 | Vilket av följande kommandon visar information om datorns PCI-bussar?.

(d) `lspci`

```
awk03@xubuntu:~$ ps au
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1559  0.0  0.0   5828   1732 tty1     Ss+  Nov07   0:00 /sbin/agetty -o
        -p -- \u --noclear tty1 linux
systemd+      1727  0.5  9.7 1314048 389816 pts/0    Ssl+  Nov07   6:29 mysqld --
        character-set-server=utf8 --collation-server=utf8_bin --default-authentication
        -plugin=mysql_n
awk03         256555  0.0  0.1   8276   5152 pts/0    Ss    11:12   0:00 -bash
awk03         256933  0.0  0.1   8284   5132 pts/1    Ss+   11:13   0:00 -bash
awk03         256944  0.0  0.1   8284   5168 pts/2    Ss    11:13   0:00 -bash
awk03         256989  0.8  0.1   8580   4532 pts/2    S+    11:14   0:27 htop
awk03         267334  0.0  0.0   9128   3648 pts/0    R+    12:05   0:00 ps au
```


**Fråga 1 | Kan *Bob* ändra innehållet i filen *schedule.txt*?
Varför/varför inte?**

Nej, gruppen *solarproj* har inga skrivrättigheter.

Fråga 2 | Kan *Alice* ta bort filen *controller.c*? Varför/varför inte?

Nej, Sticky bit är satt på katalogen *solarproj*.

Fråga 3 | Vem kan läsa filen *power*? Varför?

Alla andra förutom ägare och medlemmar i gruppen *solarproj*.

Fråga 4 | Varför kan *Alice* men inte *Bob* byta arbetskatalog till *old*?

Alice har exekveringsrättigheter på katalogen *old* men gruppen *solarproj* har inte exekveringsrättigheter.

Fråga 5 | Kan *Alice* läsa filen *temp.sym*? Varför/varför inte?

Nej, *Alice* kan läsa länken men har inte läsrättigheter på filen *temp*.

Fråga 6 | *Alice* har installerat openssh-server på en dator för att låta *Bob* fjärransluta till datorn (Xubuntu/Ubuntu).

Vilket kommando ska hon använda för att se status på tjänsten('service') *ssh*?
`systemctl status ssh`

Hur kan tjänsten('service') *ssh* stoppas? `sudo systemctl stop ssh`

Hur kan tjänsten('service') *ssh* startas? `sudo systemctl start ssh`