Mälardalens universitet

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 -alF"` 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 distibutioner 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`). Vektygen `make` och `gcc` måste vara installerade innan du börjar.

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

1. Installera vektygen `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?

Mälardalens universitet

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.

Mälardalens universitet

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:
Installning 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.