Introduktion till Linux och små nätverk

Användare och säkerhet

I Linux och Unix så har man alltid haft olika konton för olika personer. För att komma åt systemet så behöver man ett sådant konto. Det skyddas normalt av namnet på kontot, inloggningsnamnet, samt ett lösenord. Inloggningsnamnet är normalt upp till 8 små bokstäver/siffror där man börjar med ”`a-z`” och sedan upp till sju tecken från ”`0-9a-z_-`”. Era inloggningsnamn på högskolan är även inloggningsnamn till högskolans Unix-konton. För att få dessa aktiverade behöver ni dock ansöka om speciellt.

Till varje konto hör även en eller flera grupper av konton, som används för ge grupper av användare lika rättigheter. Exempelvis så tillhör en fil en användare och en grupp. Vilka dessa är kan man se i informationen om filen. Mer om filrättigheter senare.

Det finns en speciell användare som förbigår alla rättighetskontroller och det är användaren `root`. Användarkontot `root` skall bara användas för att administrera datorn, och inte för inloggning eller något annat. Det för att det är lätt att göra misstag som man inte kan göra som vanlig användare. Så ***logga aldrig*** in som användaren `root` och speciellt viktigt, surfa inte som användaren `root`. Det gör att man kan få in virus och annan elak programvara i datorn. Så vanligtvis så slår man av inloggning av `root` över nätverket och kanske även hindra inloggning från datorns konsol.

Varje användare har ett användar-id (`uid`), som är ett positivt heltal. Detsamma gäller grupper som har ett gruppid (`gid`). Så när man loggar in med korrekt användarnamn och lösenord, så får programmet/processen ett användar-id och ett huvud-grupp-id samt kan få flera sekundära grupp-id. Det är inte någon större skillnad på huvud-grupp-id och sekundära grupp-id, annat än att huvud-grupp-id får man alltid när man loggar in medans de andra bestämmer administratören om användaren kan få.

Användarkonton finns uppradade i en speciell fil som heter `/etc/passwd`. Den filen är läsbara för alla och i den står inloggningsnamn, krypterat lösenord, `uid`, `gid`, namn i klartext, hemmakatalog och kommandoskal. Alla dessa fälten är åtskilda med ett kolon-tecken (`:`). Om lösenordsfältet är tomt (blankt) så kan man logga in utan lösenord, undvik alltid detta! Om lösenordsfältet innehåller ett `x`, så kan man inte logga in över huvud taget på det kontot. Om lösenordsfältet är en `*`, så kommer en annan fil att användas som bland annat innehåller bättre lösenord. Den filen heter `/etc/shadow` och är bara läsbar för systemadministratören `root`.

Slutligen så för alla grupper så finns det en fil som heter `/etc/group`. Den har en rad för varje grupp som finns i systemet och består av gruppnamnet, lösenord för gruppen (vilket vanligtvis inte används), gruppens id (`gid`) samt en lista med de användare som vid inloggning tillhör gruppen.

För att ändra lösenord så använder man kommandot `passwd(1)`. Alla användare kan använda det för att ändra sitt lösenord. Men för att ändra andras lösenord så behöver man vara administratör (`root`). För att lägga/ta bort användare så finns kommandot `adduser(8)`/`deluser(8)` samt för att hantera grupper så använder man `addgroup(8)`/`delgroup(8)`. Se även manualen för `passwd(5)`, `group(5)` och `shadow(5)`.

Filrättigheter

När man skall skapa en fil eller katalog i en katalog, så kontrollerar Linux om man har rättighet i

den aktuella katalogen att skapa nya saker. Om man vill läsa innehållet i en fil, så kontrolleras Linux om man får läsa innehållet. För att se vilka rättigheter man har i en fil eller katalog så kan man använda växel `-l` (long) till kommandot `ls`. Man kan även lägga till växeln `-a` (all) för att se alla filer och kataloger som börjar med en punkt (`.`). Då kan det se ut så här:

```
$ ls -la
totalt 0
drwxr-xr-x  2 anders anders 2048 jan 15 13:48 .
drwxr-xr-x 21 anders anders 4096 jan 15 13:36 ..
-rwxr-x--x  1 anders anders    0 jan 15 13:49 a.sh
-r--rw-r--  1 anders users     0 jan 15 13:49 b.txt
$
```

Först på raden står det 10 tecken där första tecknet anger vilken sort fil det är, `-` är datafil, `d` är katalog etc. Sedan kommer tre grupper om tre tecken. Första gruppen om tre tecken är vilka rättigheter som ägaren till filen har, andra vilka rättigheter som gruppen som filen tillhör har samt slutligen vad alla övriga har för rättigheter på filen. Om ägaren skall göra något med filen så används bara första gruppen och de andra två ignoreras, om det är gruppen så gäller den andra gruppen av rättigheter och är man varken ägare eller ingår i filens grupp, så gäller tredje gruppen rättigheter.

Här ser man då att filen `a.sh` tillhör användaren `anders` och gruppen `anders` medans filen `b.txt` tillhör användaren `anders` och gruppen `users`. Man ser även för filen a.sh att anders får läsa(`r`), skriva(`w`) och exekvera filen som program (`x`). Alla i gruppen anders får läsa (`r`) och exekvera (`x`) filen men inte skriva i den (`w` är bytt mot `-`). Filen `b.txt` är lite lustig för att ägaren `anders` får inte skriva i filen men gruppen `users` får göra det. Detta gäller även om `anders` skulle ingå i gruppen `users`.

För att ändra rättigheter så använder man med fördel kommandot `chmod`, eftersom rättigheterna kallas mode, så kommandot kan läsas som *change mode*. Kommandot tar som enklast två argument, rättigheter och filnamn. Rättigheterna representeras med bokstäver, där man först vilka grupper av rättigheter som skall ändras, user (`u`), gruppen (`g`) eller övriga (`o`). Sedan anger man operationen man vill göra `+` för att lägga till, `-` för att ta bort och `=` för att sätta exakt. Vill man göra flera sådana operationer, så kan man skriva dem åtskilda med ett kommatecken (`,`). Exempelvis

```
$ chmod ug+w,o=rx a.sh
```

Lägger till skrivrättigheter (`w`) till user (`u`) och grupp (`g`) samt sätter övrigas rättigheter till läs och exekvera.

Det går även att använda oktala siffror för att representera rättigheterna till kommandot `chmod`, men det är krångligare och lättare att göra fel, så jag rekommenderar starkt att ni använder symboler som visats ovan. Men kort så är varje oktal siffra uppbyggd av tre bitar. Så varje grupp av `rwx` motsvarar då en siffra, så `r-x` blir då `101`, som är oktala siffran 5. Det kan vara bra att förstå när man läser instruktioner, så man inte huvudlöst och utan kritik kör en instruktion som gör dumheter.

För att ändra ägare kan man använda kommandot `chown` och grupp ändras med `chgrp`. Läs manualen[1] för att få reda på hur man använder dessa kommandon.

Access Control List (ACL)

Vanliga rättigheter enligt ovan har varit med sedan Unix-tiden, d.v.s. 1970-talet. Det finns nu en

1 Använd `man chown` för att läsa manualsidan. `man 1 chown` visar från kommandoavdelningen (nr 1).

utökad variant av rättigheter som bygger på samma princip, med `r`, `w` och `x`-rättigheter. Men istället för att bara sätta för en användare och en grupp, så kan man här sätta rättigheterna för många användare och många grupper. Så i en fil:s Access Control List (ACL), så finns en lista med ägare och användare, en lista med grupper samt övriga ej angivna användare/grupper. Det finns även en s.k. mask som anger maximala rättigheter som kan ges med ACL:en, förutom om man är ägaren.

I kursen kommer vi inte att beröra detta sätt att kontrollera rättigheter till filer och kataloger. Men för er som är intresserade av dessa, så titta gärna på manualsidorna för `acl(5)`, `getfacl(1)` och `setfacl(1)` för mer information.

Initiering av program

När program startar i Linux, så brukar de läsa konfigurationsfiler för att veta hur de skall fungerar. Vilka dessa konfigurationsfiler hittar man normalt i manualsidan till kommandot.

Normalt så tittar ett Linux-program i minst två kataloger efter inställningar. Först i `/etc`-katalogen för att hitta standardinställningar som administratören av datorn har satt in. Den konfigurationsfilen används av alla användare. Sedan så brukar programmen leta efter en konfigurationsfil i användarens egna hemmakatalog(`$HOME` eller `~`). Eftersom varje användare har egna hemmakataloger, så är det användarens egna inställningar som lagras där. Så även om administratören bestämt sig för en inställning, så kan användarna själva sätt något annat i sina egna konfigurationsfiler, eftersom de läses efter standardinställningarna i `/etc`-katalogen och eventuella inställningar där ersätter de andra värdena som är satta tidigare.

Användarnas egna konfigurationsfiler har normalt namn som börjar med en ”`.`”(punkt-tecken) eller så ligger de i en katalog som har ett namn som börjar med en ”`.`”. Detta för att kommandot ls normalt inte skriver ut filer och kataloger vars namn börjar med punkt. För att få med dem, så använder man växeln `-a` (all). Även kommandoskalet `bash`, som de flesta av er använder, har konfigurationsfiler som börjar med ett punkt-tecken (`.`). Exempelvis så innehåller filen `~/.bash_login` kommandon som exekveras när användaren loggar in, men inte när de kör andra kommandon. Vilka filer som används och när kan man se i manualsidan för `bash`[2].

Pakethantering av program

En egenskap som definierar Linux-distributioner är vilket paketeringssystem som de använder. De två vanligaste var ju RPM (Red Hat Package manager) och DPKG (Debian Package), som vi redan nämnt. RPM används av Red Hat-baserade Linuxdistributioner och DPKG används av Debian-baserade Linuxdistributioner. Vi kommer att titta på DPKG, som används av exempelvis Debian, Ubuntu och Mint med flera.

Enligt konventioner så har filer med DPKG-paket filslutet `.deb` och RPM-paket `.rpm`.

Distributionerna packar program i dessa format för att installera samt hålla reda på vilka filer som hör till ett paket. Så att installera ett sådant program är vanligtvis bara att skriva rätt kommando eller peka på rätt knapp i ett grafiskt program.

Alla paket i en distribution finns normalt i ett förråd av paket tillgängligt via Internet. För Debian kallas det systemet för att hantera detta för APT. Varje maskin har en databas som innehåller information om alla paket som finns i förråden och vilka paket som är installerade. Så normalt så hämtar administratörer aldrig paketen manuellt, utan det gör APT-systemet.

2 Använd `man bash` eller `info bash`.

Vi kommer att titta på hur vanligt kommando ser ut, nämligen `aptitude`, `apt-get` och `dpkg`. För grafiska så finns det exempelvis pakethanteraren `Synaptic`. Men kommandon är enklare att automatisera och de är alla baserade på DPKG.

Debian-paket

Ett paket är ett filarkiv, ungefär som zip eller tar, som innehåller alla filer som programmet består av, datafiler med information om paketet samt några skript. När Debian installerar ett program, så kommer det att packa upp filerna på ett temporärt ställe. Sedan exekveras ett skalprogram som finns i paketet och heter `preinstall`. Därefter flyttas filerna till rätt ställe i filsystemet och slutligen kommer ett annat skalprogram från paketet att köras, som heter `postinstall`. Dessa två skalprogram ser till att förbereda för installationen, samt efteråt göra justeringar.

När paketet skall tas bort, så finns det även två skalprogram som exekveras innan filerna tas bort och ett som slutligen exekveras efter att filerna tagits bort, `preremove` och `postremove`.

Paketen innehåller även information om vilka andra paket som måste vara installerade för att programmen skall fungera, vilka som man normalt förväntar sig skall vara installerade och vilka som normalt inte är men kan behövas ibland. Paketen kan även innehålla information om vilka paket som det inte kan vara installerade med och vilka paket som de kan ersätta.

Så när ett paket installeras så kommer de paket som måste vara installerade och de som normalt förväntas att även installeras automatiskt. Så man behöver inte bry sig om installera dessa manuellt.

Installation

Vanligen använder man kommandot `aptitude(8)` eller `apt-get(8)`. Bägge fungerar lika, men Debian rekommenderar att man använder `aptitude`. Aptitude kan även användas som ett textbaserat menyprogram, om man vill lite och titta på paketen.

För att köra `aptitude` menybaserat, så skriver man bara programmets namn i ett kommandofönster. Notera att man behöver vara `root` för att kunna installera program. Enklast är att använda programmet `sudo(8)`.

```
$ sudo aptitude
```

För att avsluta programmet, så gör man kommandot '`q`'.

Anta att vi vill installera paketet `ntp`. Programmet ntp ställer in tiden på den lokala datorn efter atomur som finns på internet.

Först uppdaterar man databasen över tillgängliga paket som finns i Debian arkiv på datorn med följande kommando:

```
$ sudo aptitude update
```

Sedan kan man installera paketet med följande kommando:

```
$ sudo aptitude install ntp
```

För att läsa information om paketet så tittar man med fördel i katalogen `/usr/share/doc/ntp` för paketet `ntp`. Motsvarande katalog finns för alla paket under katalogen `/usr/share/doc`. Så glöm ej bort att titta där när ni installerar program.

Om man sedan vill se till att alla paketen har senaste säkerhetsuppdateringarna, så kan man uppgradera till senaste version. Det gör man med följande kommandon:

```
$ sudo aptitude update
$ sudo aptitude upgrade
```

Notera att första raden uppdaterar databasen i datorn och den andra raden uppgraderar alla installerade program. Enkelt eller hur?

Borttagning

För att ta bort ett paket och alla filer som hör till det så kan man även här använda `aptitude`.

Om man vill ta bort paketet, men kunna återinstallera det med samma inställningar, så använder man följande kommando.

```
$ sudo aptitude remove ntp
```

Detta tar inte bort konfigurationsfiler. Det gör att om man installerar paketet igen, så kommer det att ha samma inställning som tidigare.

Om man även vill ta bort konfigurationsfiler, så använder man följande kommando:

```
$ sudo aptitude purge ntp
```

Då kommer även alla konfigurationsfiler från paketet att raderas från datorn.

Debians paketförråd

Var finns paketen som APT via kommandot `aptitude` hämtar?

De finns i förråd som listas i konfigurationsfilen `/etc/apt/sources.list` och i de filer som finns i katalogen `/etc/sources.list.d/`. Så om man vill lägga till eller ta bort ett förråd, så anger man var de finns dessa kataloger och filer. Notera att filerna i `/etc/sources.list.d/` måste avslutas med filnamns-suffixet `.list`.

Logg-filer och felsökning

När något händer i Linux-systemet så skrivs det ofta in i så kallade logg-filer. Dessa är väldigt användbara för att se vad som händer om något går fel och i så fall vad. Det kan vara bra att titta i dem även när allt fungerar så att man lättare känner igen när allt är bra och vad som skiljer sig när det uppstått något problem.

Dessa filer finns i katalogen `/var/log`. Exempel på vanliga logg-filer att titta i är `dmesg`, `syslog` och `messages`. Men det finns många andra, som exempelvis i katalogen `ntpstats` som `ntp`-servern (som hanterar tiden) skriver i.

Dessa filer kan bli ganska stora med tiden, så därför finns en mekanism för att se till att de inte blir för stora. Detta kallas att man *rullar loggarna* och hanteras av paketet `logrotate`. Inställningar för hur det skall göras finns i `/etc/logrotate.d` och filen `/etc/logrotate.conf`.
Vad som händer är att loggfilen kopieras till ett nytt namn där man lägger till en siffra efter namnet, exempelvis `syslog.1`. Filen kan även komprimeras för att spara ännu mera plats och då för de `.gz` i slutet av filnamen, eftersom programmet för komprimering är `gzip`. Då kan det se ut så här: `syslog.2.gz`. För att läsa dessa komprimerade filer så kan man använda sig av kommandona `zmore` eller `zcat`.

Om man vill se vad som händer när man exempelvis sätter in en usb-minnessticka i datorn, så kan man titta vad som händer i logg-filerna `dmesg`, `syslog` och `messages`. Det gör man enklast ge-

nom att ansluta till datorn med ett kommandofönster och skriva följande kommando:

```
$ tail -f /var/log/{dmesg,syslog} /var/log/messages
```

Då kommer `tail` att visa de senaste 10 raderna i filerna, och varje gång en ny rad läggs till, så kommer även den raden att visas.

Tid och synkronisering

För att logg-filer skall vara användbara och att datorn skall fungera bra som fildelare, så behöver alla datorer i nätverket ha rätt tid. Dels för att man skall se när något händer mellan två datorer, och dels att alla filer skall få samma ändringstid. Vissa program kan bete sig väldigt konstigt om tiden inte är korrekt. För att synkronisera tiden används då protokollet NTP (Network Time Protocol), som har ett program som finns paketerat i paketet `ntp`. Så det är bara att installera paketet om man har nätverksanslutning, och så behöver man inte bry sig om att ställa klockan igen.

Konfigurationsfilen `/etc/ntp.conf` och `/etc/default/ntp`. Det är två standardplatser för konfigurationsfiler för Debian-paket. I `/etc/default` finns filer för olika paket som servrar använder sig av. Paketet `ntp` installerar en server, så därför finns filen `/etc/default/ntp`. Om man vill göra mer detaljerade inställningar, så kan man göra det i `/etc/ntp.conf`. Det går att ändra i bägge filerna, men läs gärna dokumentationen i katalogen `/usr/share/doc/ntp/` först.

Referenser

- https://en.wikipedia.org/wiki/Dpkg
- https://en.wikipedia.org/wiki/RPM_Package_Manager
- https://en.wikipedia.org/wiki/Advanced_Packaging_Tool
- https://en.wikipedia.org/wiki/Network_Time_Protocol
- https://wiki.debian.org/Aptitude