

# Linux 2

# ITINF 2021

## Lektion 3

# Uppvärmning

- Kommentarer / önskemål efter lektion 2?
  - större övningar?
  - övningar i grupp?
- Grupper kommer bestämmas under dagen!

# Idag

- Paket och installation med `apt` / `yum`
- Installationsexempel: Apache webbrowser
- Services – start etc manuellt resp vid boot
- Cron och schemaläggning
- Loggar

# Pakethantering

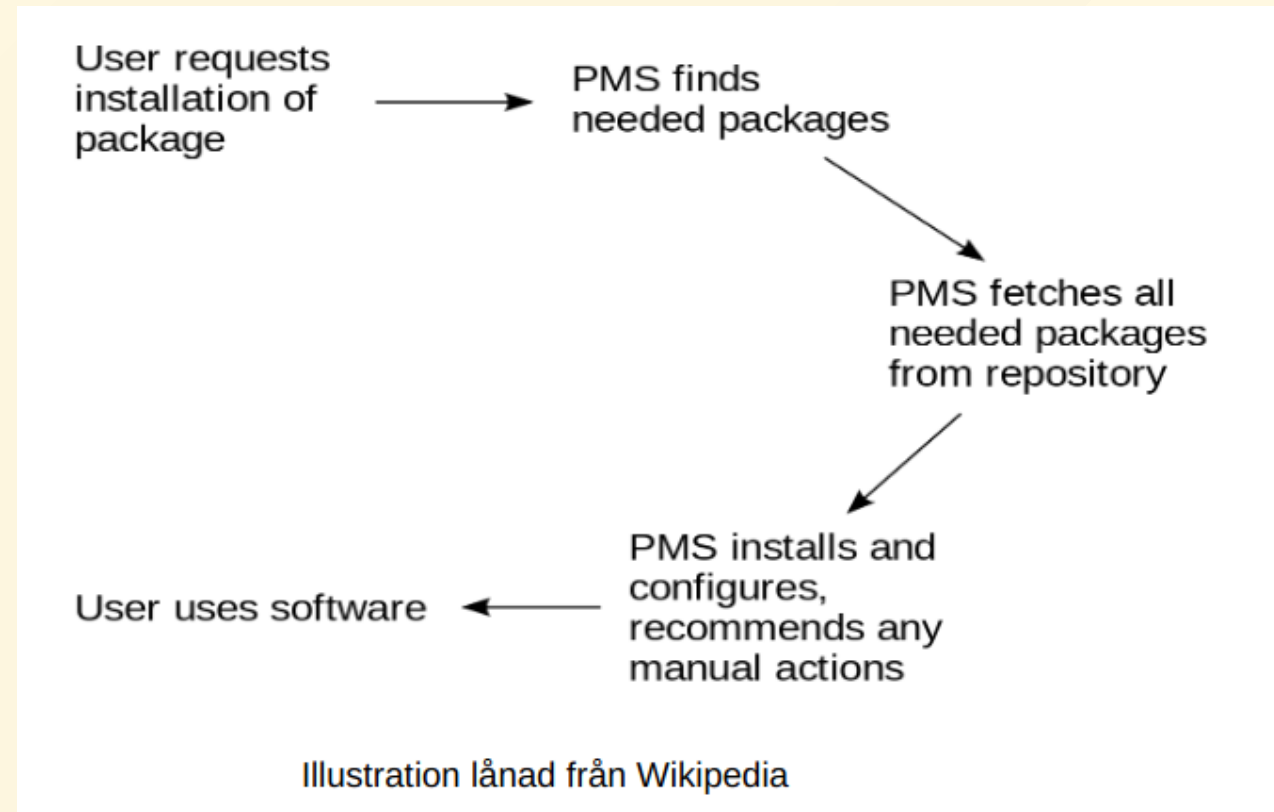
# Standardpaket

- Egen paketering av programversioner för installation sker ofta genom ett deployment- verktyg, t ex Maven eller Jenkins, eller så skapar man paket med sina filer
- Standardpaket installeras oftast med hjälp av pakethanterare som även sköter beroenden

# Pakethantering

- Paket innehåller ett program i skick som är enkelt att installera, plus information om beroenden
- Pakethanteraren sköter kontakt med repository, Nedladdning, installation av paket samt även När nyare versioner av paket finns tillgängliga.

# Pakethantering



# Paket

- `tar` (eller möjligen `zip`) för mer manuell hantering
- `apt` (dpkg-paket) för Debian-familjen, bl a Ubuntu
- `yum` (rpm-paket) för Red Hat-familjen, bl a Fedora



# apt

- `apt update` -- uppdatera *listan med paket*
- `apt upgrade` -- uppgradera *ett specifikt paket*
- `apt install <package>`
- `apt remove <package>`
- `apt list --installed`
- `apt list --upgradable`

# Övning 1

Uppvärmningsövning!

Lista alla paket du har installerade på din Linux-maskin. Se efter vad du känner igen.

# Installation

När programmet är installerat är det redo att köra!

Om programmet är en daemon/service/bakgrundsprocess, så måste den det instrueras att starta i bakgrunden mha `systemd` eller `init.d`.

(en daemon/service är något som ska vara igång hela tiden, snarare än något som startas av en användare manuellt när det behövs).

# Daemoner

**med systemd**

# Daemon

“ In multitasking computer operating systems, a daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user. ”

*(Wikipedia)*

# systemd och systemctl

- `systemd` är både namnet på ett paket av program för att initiera och hantera daemoner, och namnet på en daemon som är central för den hanteringen
- Det kommando man mestadels använder heter `systemctl`

# systemd och systemctl

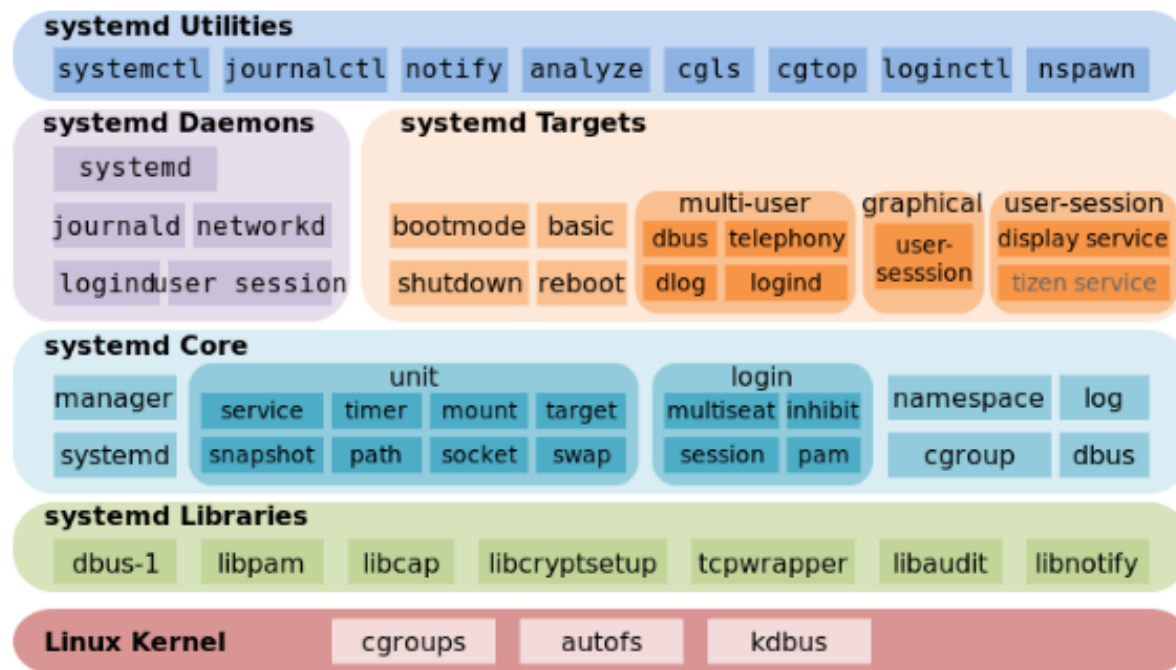


Bild från [www.linux.com](http://www.linux.com)

# init.d

- Föregångaren till systemd (men används fortfarande)
- `/etc/init.d/` -- start/stop-script per daemon
- `/etc/rc.d/rc.[0-6]` -- vad som ska starta för varje runlevel
  - runlevel 0: stäng av dator
  - runlevel 1: single-user/recovery
  - runlevel 2-4: CLI utan/med nätverk
  - runlevel 5: GUI
  - runlevel 6: reboot
- `/etc/inittab` -- legacy-fil som bestämmer vad som händer vid varje runlevel



# systemctl

- Hjälpmedel för att starta deamoner och även lägga in att de skall startas vid boot
- `/etc/systemd/system/` (dina egna inställningar)
- `/lib/systemd/system/` (från pakethanteraren)

# systemctl

- `systemctl start <service>` -- starta nu
- `systemctl stop <service>`
- `systemctl enable <service>` -- starta vid boot
- `systemctl disable <service>`
- `systemctl status <service>`
- `systemctl restart <service>`
- `systemctl is-active <service>`

# systemctl

- `systemctl list-dependencies <service>`
- `systemctl cat <service>`
- `systemctl show <service>`

# Övning 2

- Installera webbservern Apache.
  - Paketet heter `apache2`
- Starta tillhörande daemon
- Kontrollera att daemonsen är igång
- Ta en titt med `systemctl list-dependencies` `systemctl cat` och `systemctl show`

**Exempel:** `systemctl cat apache2`

# Göra en egen service / daemon

1. Först behöver man förstås programmet som skall bli en service
2. Sen, skapa en fil för servicen i `/etc/systemd/system/` (eller `/lib/systemd/system/`)
  - `systemctl daemon-reload` för att be `systemd` läsa configfiler
  - Nu kan du starta din service

# Göra en egen service / daemon

Mall från <https://www.shubhamdipt.com/blog/how-to-create-a-systemd-service-in-linux/>

```
[Unit]
```

```
Description=<människo-läsbar beskrivning av tjänsten>
```

```
[Service]
```

```
User=<kör som vem? t ex root>
```

```
WorkingDirectory=<vad ska cwd vara för scriptet? t ex /tmp>
```

```
ExecStart=<kommandorad för att starta scriptet>
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target # betyder: runlevel 2, dvs när allt som behövs för CLI är igång
```

# Övning 3

- Skapa ett enkelt program med den enda funktionen att det skriver till en loggfil (välj själv var denna skall ligga) varje gång det startas, med tidsstämpel, och sedan lägger sig i en oändlig loop som inte gör något
- Gör en daemon av detta program – den kan t ex heta `exercise3d`
- Starta din daemon, starta om den etc och kontrollera att den skriver till filen du pekat ut och ligger kvar som bakgrundsprocess.



# Övning 3

```
#!/bin/bash
myfile=/tmp/exercise3d.log
timestamp=`date +%Y-%m-%d_%H-%M-%S`

echo $timestamp ": started" >> $myfile;

while true
do
    sleep 1
done
```

# Övning 3

```
[Unit]
```

```
Description=Övning 3
```

```
[Service]
```

```
User=nevyn
```

```
WorkingDirectory=/var/scripts
```

```
ExecStart=/var/scripts/exercise3d
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

# Användning av daemoner

- Vad behöver ligga som bakgrundsprocess?
- Vad bör startas automatiskt vid boot?

# Övning 4

- Fundera på vad ni tycker är ett typiskt driftscenario med en webbapplikation och i det sammanhanget följande:
  - Vilka program vill man skall ligga som bakgrundsprocesser?
  - På vilka kriterier tycker ni att just de programmen bör ligga som bakgrundsprocesser?

# crontab

aka schemaläggning

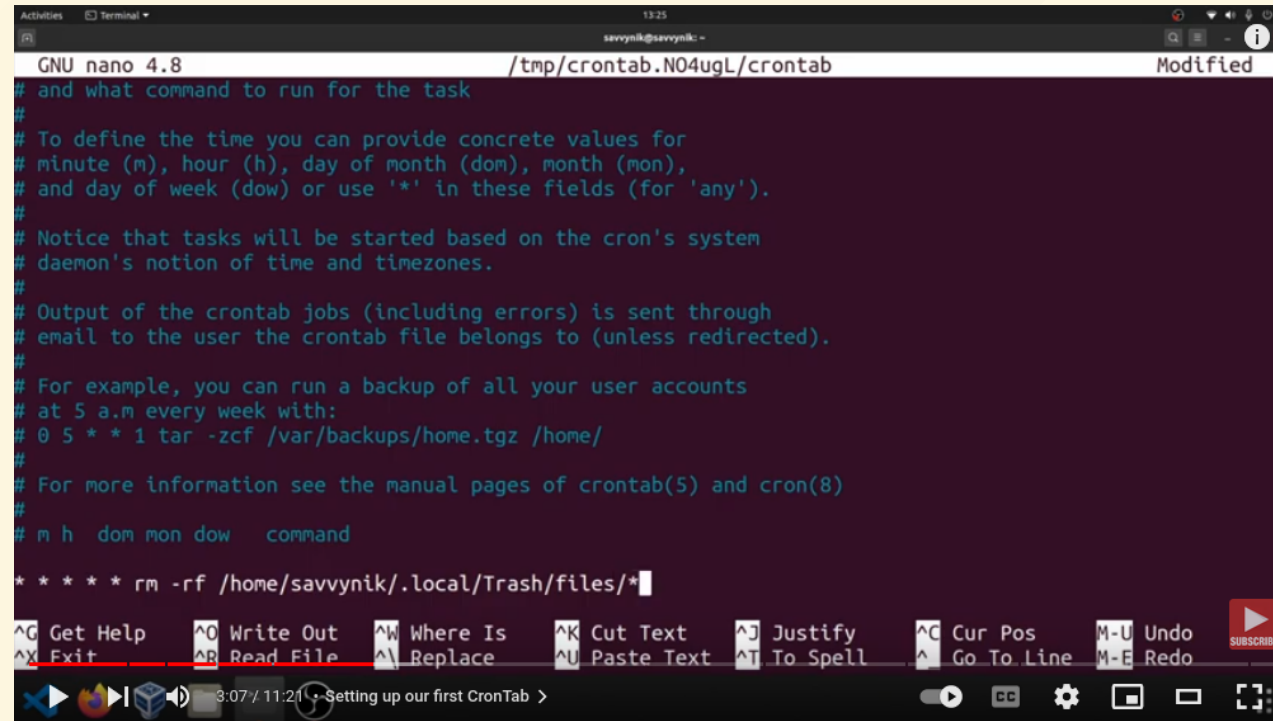
# Daemon kontra schemaläggning

- Program som skall vara *igång hela tiden*, t ex för att det lyssnar efter förbindelser: **daemon**
- Program som skall köras *regelbundet*, automatiskt, men är klart när det är klart: **cronjob**

# Schemaläggning

- `crond`
- Systemets schemaläggning i `/etc/crontab` och `/etc/cron.d/`
- Användare kan också schemalägga användar-specifika jobb med `crontab`

# Schemaläggning med cron



```
GNU nano 4.8 /tmp/crontab.N04ugL/crontab Modified
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * rm -rf /home/savvynik/.local/Trash/files/*
```

[https://www.youtube.com/watch?v=owLj5Vk\\_LVI](https://www.youtube.com/watch?v=owLj5Vk_LVI)



# Crontab

```
1 2 3 4 5 <cmd>
```

- 1: minut (0-59)
- 2: timme (0-23)
- 3: dag av månad (1-31)
- 4: månad av år (1-12)
- 5: dag av vecka (0-7)
- Stöd för listor ( `1,2,3` ) och intervall ( `2-4` ) och kombo ( `1,3-4,7` )

```
50 0 * * 1-5    /usr/bin/testscript.sh # 00:50 varje vardag
0  * * * *      /bin/beep           # varje timme
45 2 1 * *      /u/backup.sh        # 02:45 första varje månad
```

# Crontab

- För att lägga in jobb i användarens egen crontab:

```
crontab -e
```

- Behöver inte ange användare, då jobben körs som ägaren.

- Läsa användarens crontab:

```
crontab -l
```

- Ta bort alla cronjob:

```
crontab -r
```

# Övning 5

- Gör ett script som kontrollerar om daemonen `apache2` är igång och skriver resultatet till en loggfil (som du själv väljer) med tidsstämpel.
- **Testkör ditt script** från terminal.
- Lägg in ett cronjob som kör ditt script en gång per minut måndag-fredag varje vecka. Se att din loggfil fylls på.
- När du testat en stund vill du nog ta bort cronjobet igen 😊

# Övning 5

- Script fixar ni vid det här laget

- Men här är en ledtråd:

```
if [ `systemctl is-active apache2` == "active" ]
```

- Crontab:

```
* * * * 1-5 <script_path>
```

# Loggar

# Loggar: `/var/log`

- `/var/log/syslog` and `/var/log/messages` store all global system activity data, including startup messages. Debian-based systems like Ubuntu store this in `/var/log/syslog`, while Red Hat-based systems like RHEL or CentOS use `/var/log/messages`.
- `/var/log/auth.log` and `/var/log/secure` store all security-related events such as logins, root user actions, and output from pluggable authentication modules (PAM). Ubuntu and Debian use `/var/log/auth.log`, while Red Hat and CentOS use `/var/log/secure`.
- `/var/log/kern.log` stores kernel events, errors, and warning logs, which are particularly helpful for troubleshooting custom kernels.
- `/var/log/cron` stores information about scheduled tasks (cron jobs). Use this data to verify that your cron jobs are running successfully. (*notera: inte sant i ubuntu! loggas till syslog*)

(Från <https://www.loggly.com/ultimate-guide/linux-logging-basics/>)

# Loggläsning

- `tail -f /var/log/syslog`, kanske tillsammans med `grep`
- `less` is `more`
- `dmesg`
- Skapa larm när något hänt tillräckligt många gånger (cronjobb till hjälp)
- Titta i specifik logg när något verkar ha gått fel

# Loggläsning

`tail /var/log/syslog` exempel:

```
Aug 13 10:08:25 mensaab6 systemd[9783]: Listening on GnuPG
cryptographic agent and passphrase cache.
Aug 13 10:08:25 mensaab6 systemd[9783]: Listening on GnuPG
cryptographic agent and passphrase cache (restricted).
Aug 13 10:08:25 mensaab6 systemd[9783]: Reached target Sockets.
Aug 13 10:08:25 mensaab6 systemd[9783]: Reached target Basic
System.
Aug 13 10:08:25 mensaab6 systemd[9783]: Reached target Default.
Aug 13 10:08:25 mensaab6 systemd[9783]: Startup finished in 71ms.
Aug 13 10:08:25 mensaab6 systemd[1]: Started User Manager for UID
1001.
```



# Loggläsning

`dmesg` exempel:

```
[ 13.633925] wlan0: authenticated
[ 13.636666] wlan0: associate with 30:b5:c2:96:44:49 (try 1/3)
[ 13.641591] wlan0: RX AssocResp from 30:b5:c2:96:44:49 (capab=0x411
status=0 aid=2)
[ 13.655527] wlan0: associated
[ 7320.966704] CPU1: Package temperature above threshold, cpu clock throttled
(total events = 1)
[ 7320.966706] CPU5: Package temperature above threshold, cpu clock throttled
(total events = 1)
[ 7320.971681] CPU2: Core temperature/speed normal
[ 7320.971682] CPU3: Package temperature/speed normal
[ 7320.971684] CPU5: Package temperature/speed normal
```

# Skriva till loggar

- Vad är intressant att logga?
  - Felsituationer
  - Inloggningar och annat säkerhets-relaterat
  - Nyttjad kapacitet
- Applikationsloggar kan med fördel få egna filer
  - Ta en titt i `/var/log` och dra slutsatser om vilka applikationer som har egna loggfiler

# Övning 6

- Ta en titt i `/var/log/apache2/`
  - Var letar du om du får ett fel i t ex en php-sida på din webbserver?

# Intressanta loggar

- `/var/log/syslog`
- `/var/log/messages` (samma som `dmesg`) ( `/var/log/boot.log` ibland)
- `/var/log/auth.log`
- `/var/log/faillog`, `faillog -a`

# Övning 7

- Bygg följande funktioner:
  - Var 10:e sekund kontrolleras ifall filen `/var/testfil1` ändrats. Om den ändrats skrivs ett radnummer (börja på 1), tidsstämpel och meddelandet att filen har ändrats till en logfil på lämpligt ställe.
  - Varje gång logfilen innehåller minst 10 rader flyttas innehållet till slutet av en annan fil på lämpligt ställe, en bak-fil, och logfilen är tom tills nästa rad tillkommer.
  - En gång per timme raderas bak-filen.
- Fundera särskilt på om / hur det skall delas upp i flera program, och använd `systemctl` respektive `crontab` där det är tillämpligt.

```
$ cat changelogger.sh
#!/bin/bash
filename=/var/testfil1
logfile=/var/log/mytestd/mytestd.log
rollfile=/var/log/mytestd/mytestd.log.bak
n=1
md1=$(md5sum $filename)

while true
do
    sleep 10
    md2=$(md5sum $filename)
    if [ "$md1" != "$md2" ]
    then
        timestamp=`date +%Y-%m-%d_%H-%M-%S`
        echo $n ": " $timestamp " file changed" >> $logfile
        n=$((n+1))
        if [ `wc -l $logfile | awk '{print $1}'` -ge 10 ]
        then
            cat $logfile >> $rollfile
            truncate -s 0 $logfile
        fi

        fi
        md1=$md2
    done
```

```
$ crontab -e

# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
0 * * * * rm -f /var/log/mytestd/mytestd.log.bak
```

```
$ cat /etc/systemd/system/changelogger.service
```

```
[Unit]
```

```
Description=Logs changes to /var/testfil1
```

```
[Service]
```

```
User=nevyn
```

```
WorkingDirectory=/tmp
```

```
ExecStart=/home/nevyn/changelogger.sh
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl daemon-reload
```



Tillbakablick, reflektion, kommentarer ...