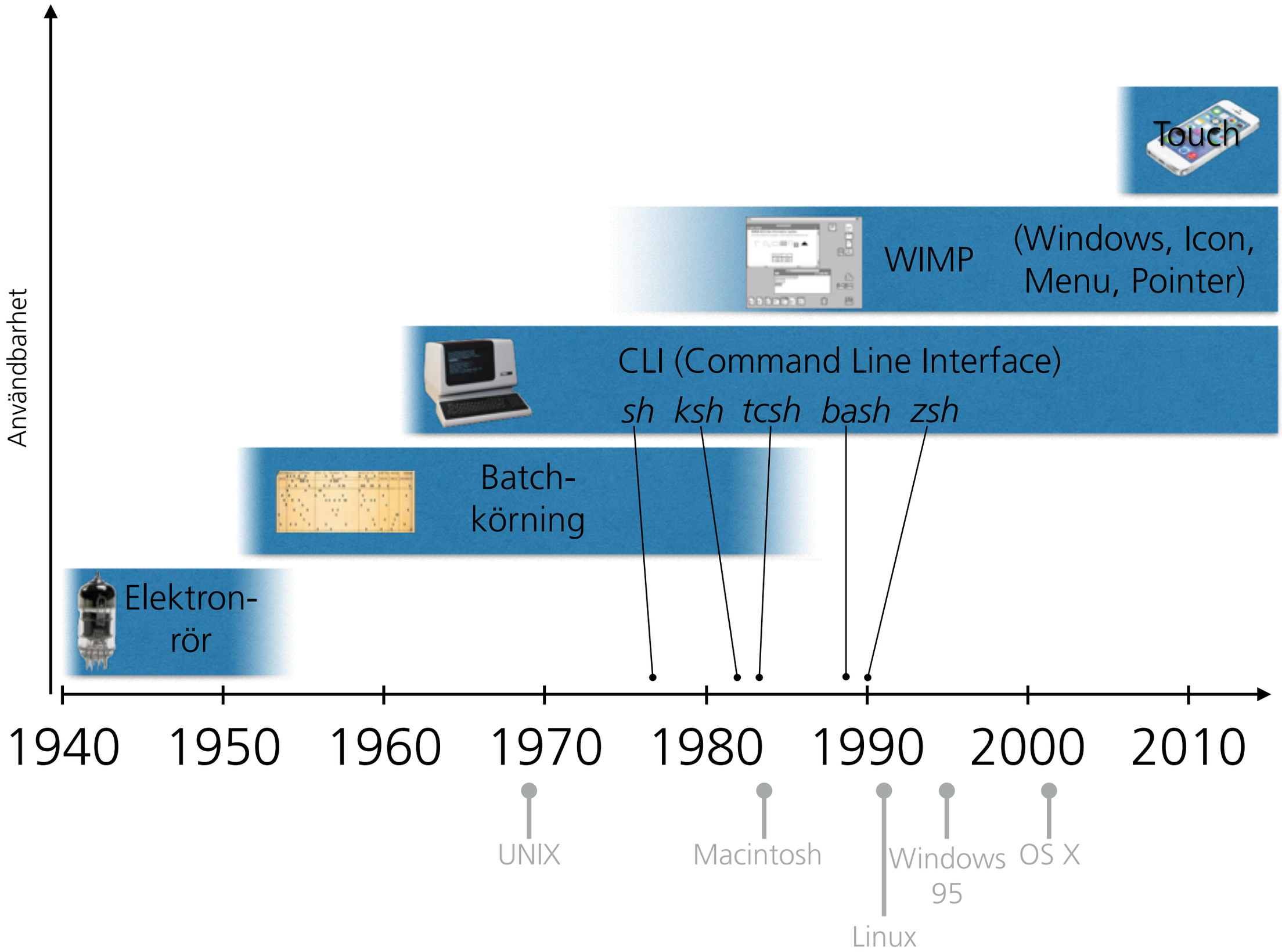
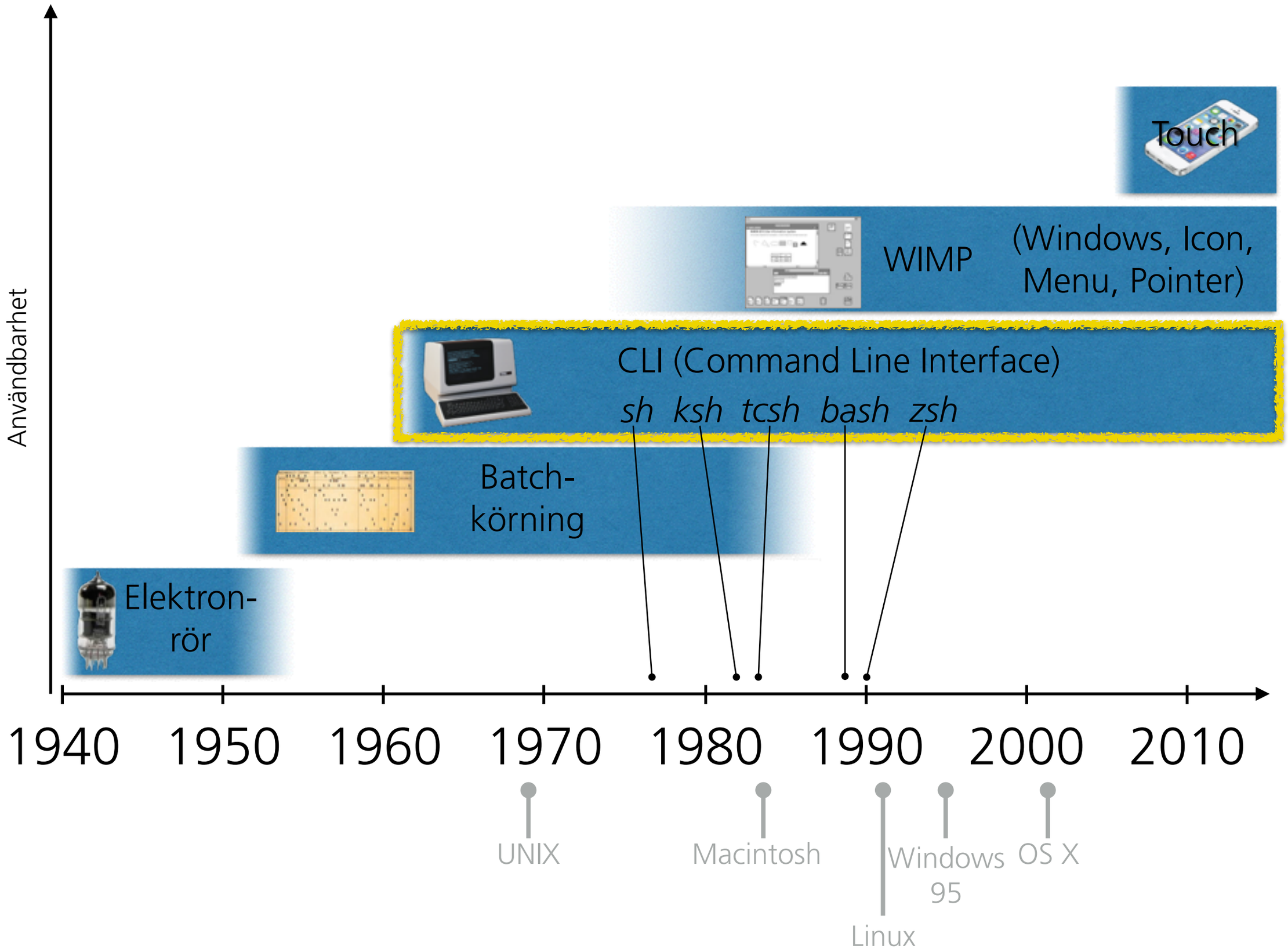
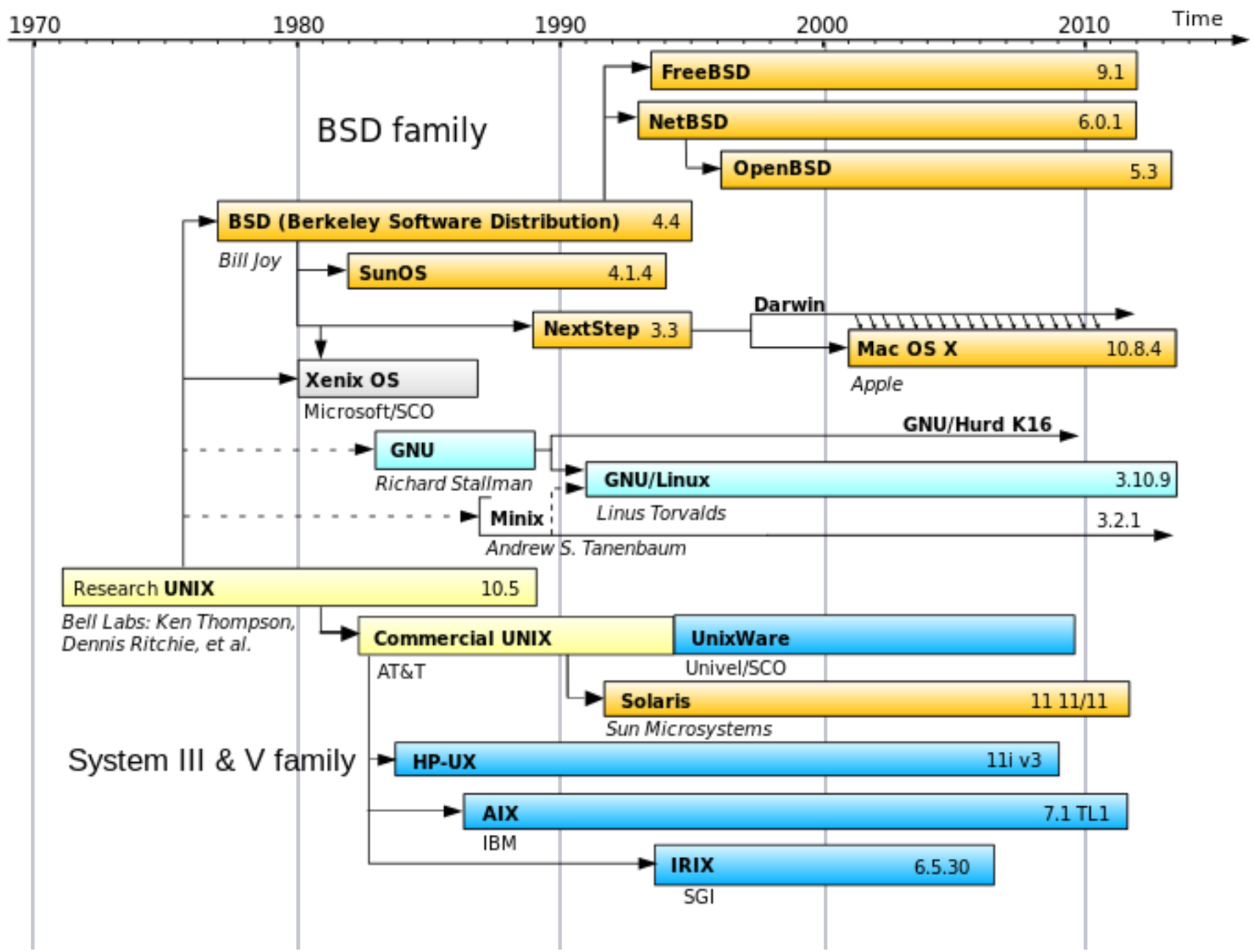
Terminal

– vad det är
– vad man kan göra med den

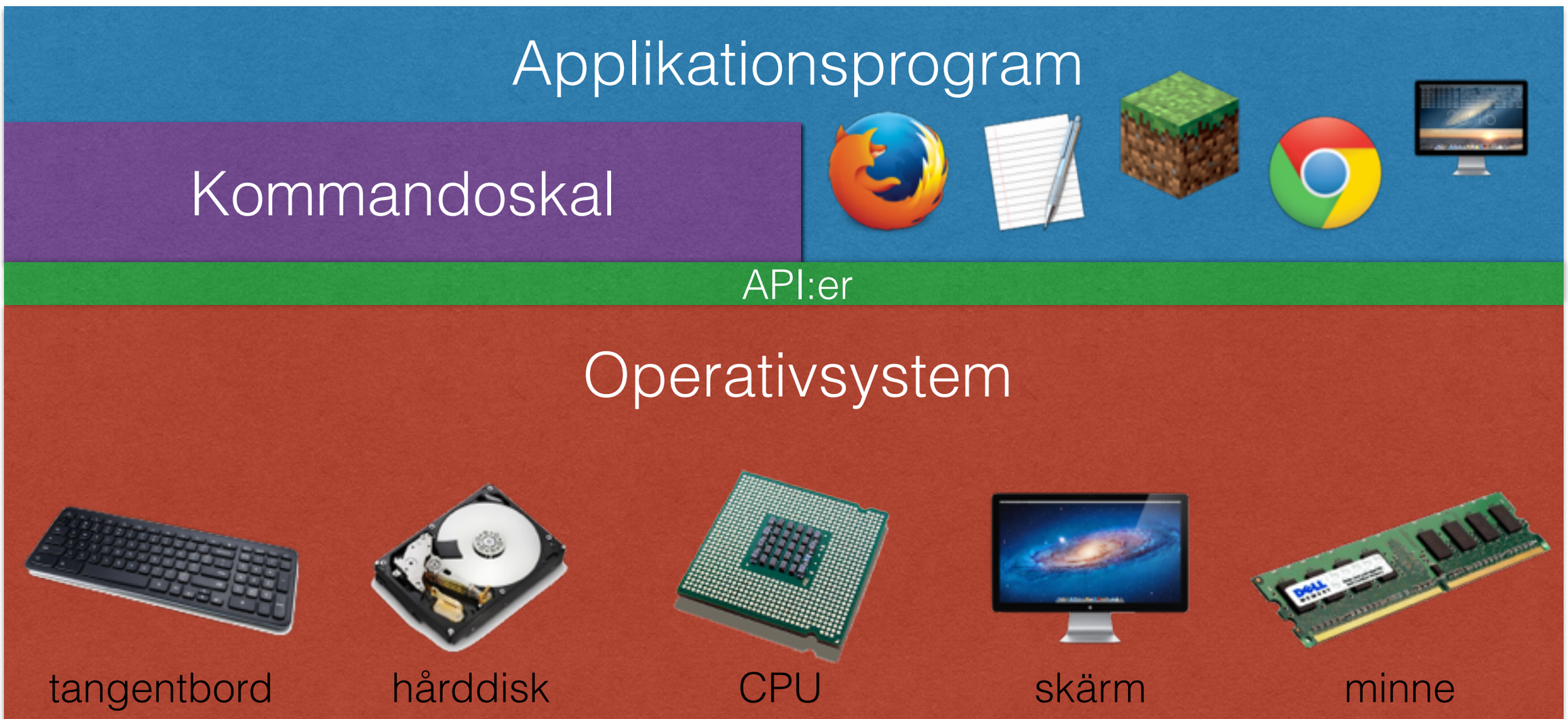

Touch
WIMP
(Windows, Icon, Menu, Pointer)
CLI (Command Line Interface)
sh ksh tcsh bash zsh
Batch-körning
Elektron-rör
Användbarhet
1940
1950
1960
1970
1980
1990
2000
2010
UNIX
Macintosh
Linux
Windows 95
OS X


Touch
WIMP
(Windows, Icon, Menu, Pointer)
CLI (Command Line Interface)
sh ksh tcsh bash zsh
Batch-körning
Elektron-rör
Användbarhet
1940
1950
1960
1970
1980
1990
2000
2010
UNIX
Macintosh
Linux
Windows 95
OS X

UNIX-historia:



Hur fungerar datorn?



Exempel #1

Problem:
Ett antal kataloger i ett filträd saknade filen ”start.txt”.

Man kan lösa detta en katalog i taget, men det är manuellt arbetskrävande (och tråkigt)

Lösning:
Skriv en for-loop i bash och lös det direkt:

```
for i in $(find . -type d)
# ’i’ kan se ut så här: ’./servers/fyra_fem/dokumentation’
do
  if [ ! -f "$i/start.txt" ]; then
    echo "===== $(basename $i) =====" > ”${i}/start.txt"
    chown _www:_www $i
  fi
done
```

Exempel #2

Problem:
Jag blev inte nöjd med filerna i förra exemplet och vill ta bort allihopa.

Lösning:
Hitta dem med find och radera dem direkt:

```
find . -name start.txt -exec rm -rf {} \;
```

Om jag i stället vill ändra rättigheterna på filerna så att alla får läsa filen gör man så här:

```
find . -name start.txt -exec chmod ugo+r {} \;
```

(u=user, g=group, o=other; r= read)

Exempel #3

Problem:
Jag vet att en av mina filer innehåller texten ”Blittan Blattan”. Var finns filen?

Lösning:
Leta i filinnehållet med `grep`:

```
grep -r "Blittan Blattan" *
```

På OS X kan man använda kommandot `mdfind`:

```
mdfind "Blittan Blattan"
```

Problem:
Jag vet att jag har en fil som heter något med XYZ:

Lösning:
Leta i filinnehållet med `find`:

```
find . -name *XYZ*
```

(söker i befintlig katalog och nedåt i filträdet)

Alternativt kan man använda `locate`:

```
locate XYZ
```

(söker i hela filsystemet)

Exempel #4

Problem:
Jag kan inte avmontera en USB-tumme – något program ”håller” den. Vilket?

Lösning:
Använd kommandot `lsof`:

```
lsof | grep /Volumes/tumme
```

Problem:
Mitt program skriver något i en fil, men jag vet inte vilken!

Lösning:
Använd `lsof` för att ta reda på vilka filet mitt program har öppna:

```
lsof -p $(pgrep "Programnamn")
```

Exempel #5

Problem:
Jag undrar vilka datorer min dator kommunicerar med!

Lösning:
Använd `lsof` för att se vilka program som lyssnar (”LISTEN”) eller har en förbindelse igång (”ESTABLISHED”):

```
lsof -i4 | egrep "LISTEN|ESTABLISHED"
```

Kör du detta som dig själv kan du endast se dina egna förbindelser. Kör det som root i stället för att se alla *förbindelser!*

Exempel #6

Problem:
Jag vill köra backup automatiskt hemifrån mot en dator på jobbet, utan att betala något för det

Lösning:
Använd kommandot rsync och gör hela backupkörningen i en enda rad (“oneliner”):

```
ssh -p 22 computer.cs.lth.se 'echo "password" | hdiutil attach /
Volumes/Backup/Privat_backup.sparseimage -nobrowse' >& /dev/null ; /
usr/local/bin/rsync -aNHAXxv --protect-args --fileflags --force-change
--rsync-path="/usr/local/bin/rsync" /Users paravel.cs.lth.se:/Volumes/
Privat_backup/Users | tail -n 3 | grep bytes | /Users/peterm/bin/
sendEmail -f <mailto:Peter.Moller@comhem.se>
Peter.Moller@comhem.se<mailto:Peter.Moller@comhem.se> -t
<mailto:Peter.Moller@me.com>
Peter.Moller@me.com<mailto:Peter.Moller@me.com> -u Backuprapport -s
mail1.comhem.se  -xu user -xp password ; ssh -p 22 computer.cs.lth.se
hdiutil detach /Volumes/Privat_backup -quiet
```

Förklaring:
1. Koppla upp mot min dator på jobbet (använder sparad ssh-nyckel) och montera den krypterade och lösenordsskyddade disk-imagen där backupen skall läggas
2. Kör rsync från min /Users till denna disk (med verbose på)
3. Ta en rad från utskriften på rsync och skicka till min mailadress så att jag får reda på att något har hänt
4. Koppla upp mig igen och avmontera disk-imagen

Exempel #7

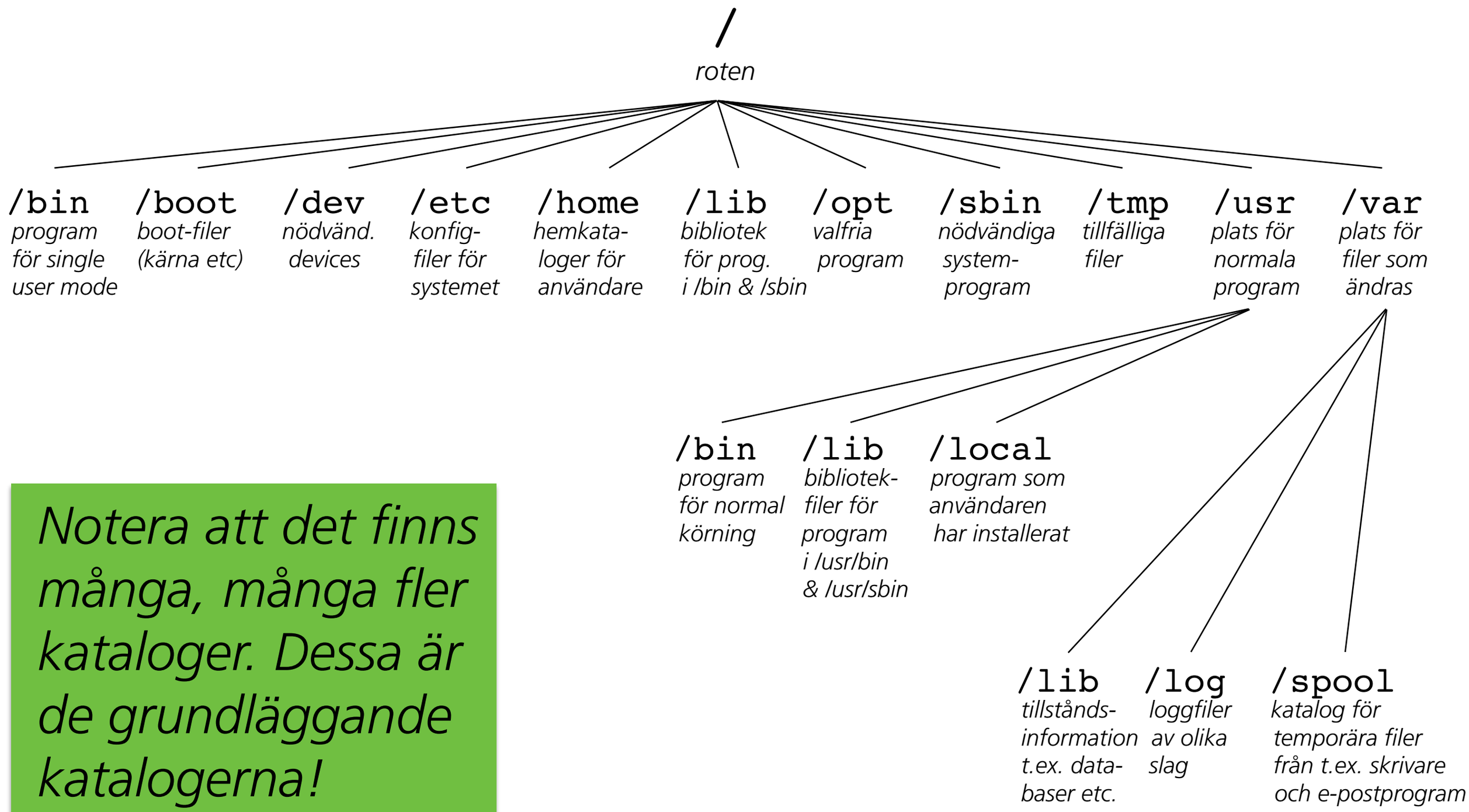
```
Hostname: Peter-Mollers-MacBook-Pro.local (NAT: 10.0.1.14 / 130.235.16.55)   Running: Mac OS X 10.10.2   Uptime: 24 days

Active interfaces, in service order:
1. Wi-Fi                en0  10.0.1.14        fe80::1610:9fff:fece:fd95

Established IPv4-connections: (24 Feb 10:26)    (Explanation: Normal, Safe protocol, Ambiguos DNS, No DNS-name, User is root)

Program          Port          User          #   Hostname [.local]                                          Country          City
2BUA8C4S2C.com.  62057         cs-pmo        1   localhost                                                  Sweden           Lund
apsd             http          root          2   host-230.akamai-cluster.sunet.se                           Sweden           Ystad
apsd             5223          root          2   17.110.225.15 could not be looked up! (DNS timeout)        United States    Cupertino
awacsd           https         root          1   p01-conduit.connectivity.icloud.com                        United States    Cupertino
Box Sync         https         cs-pmo        1   74.112.184.85 could not be looked up! (DNS timeout)        United States    Los Altos
Box Sync         https         cs-pmo        1   74.112.185.86 could not be looked up! (DNS timeout)        United States    Los Altos
CalendarAgent    https         cs-pmo        4   imap.lu.se                                                 Sweden           Lund
CalendarAgent    https         cs-pmo        2   17.172.192.29 could not be looked up! (DNS timeout)        United States    Cupertino
Dropbox109       https         cs-pmo        1   snt-re3-7c.sjc.dropbox.com                                 United States    San Francisco
Dropbox109       https         cs-pmo        1   d-6a.sjc.dropbox.com                                       United States    San Francisco
Dropbox109       https         cs-pmo        1   client-11b.v.dropbox.com                                   United States    San Francisco
Dropbox109       https         cs-pmo        1   client-17a.v.dropbox.com                                   United States    San Francisco
Dropbox109       17500         cs-pmo        1   cs-petmolmac.cs.lth.se                                     Sweden           Lund
Dropbox109       https         cs-pmo        2   ec2-174-129-20-125.compute-1.amazonaws.com                 United States    Seattle (Greater Duwamish)
Dropbox109       https         cs-pmo        1   ec2-174-129-209-113.compute-1.amazonaws.com                United States    Seattle (Greater Duwamish)
Dropbox109       https         cs-pmo        1   ec2-50-19-219-123.compute-1.amazonaws.com                  United States    New York (Manhatten)
Mail             imaps         cs-pmo        4   imap.lu.se                                                 Sweden           Lund
ocspd            http          root          2   a92-123-206-203.deploy.akamaitechnologies.com              Sweden           Stockholm
opendirectoryd   ldap          root          1   uwdc04.uw.lu.se                                            Sweden           Lund
opendirectoryd   msft-gc       root          1   uwdc05.srv.lu.se                                           Sweden           Lund
Safari           6263          cs-pmo        2   localhost                                                  Sweden           Lund
syncdefaultsd    https         cs-pmo        2   17.172.192.35 could not be looked up! (DNS timeout)        United States    Cupertino

Established IPv6-connections: (24 Feb 10:26)

Program          Port          User          #   Hostname [.local]                                          Country          City
No established IPv6-connections

Listening ports: (24 Feb 10:26)

IPv4  IPv6  Program          User          Port#  PortName        Servicerange
-     6     2BUA8C4S2C.com.  cs-pmo               anonymous port  *
4     -     2BUA8C4S2C.com.  cs-pmo               anonymous port  127.0.0.1
-     6     awacsd           root                 anonymous port  *
4     -     Dropbox109       cs-pmo               anonymous port  *
4     -     Dropbox109       cs-pmo               anonymous port  127.0.0.1
-     6     FileMaker Pro    cs-pmo         5003  fmpro-internal  *
4     6     kdc              root             88  kerberos        *
4     6     launchd          root            548  afpovertcp      *
-     6     launchd          root            631  ipp             *
4     6     launchd          root            445  microsoft-ds    *
4     6     launchd          root           5900  rfb             *
4     6     launchd          root             22  ssh             *
4     -     launchd          root            631  ipp             127.0.0.1
```

UNIX-filosofin:

- Skriv program som gör en sak och gör den väl
- Skriv program som fungerar tillsammans
- Skriv program som arbetar med textströmmar, för det är ett universellt interface

Doug McIlroy

Filsystem



Notera att det finns många, många fler kataloger. Dessa är de grundläggande katalogerna!

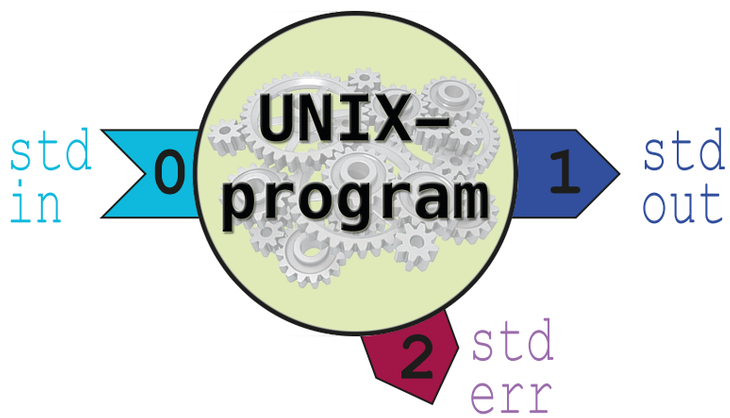
Processer

- UNIX har en familjelik processmodell: endast en existerande process kan skapa en ny process (förälder- och barnprocess). Detta ger ett familjeträd av relaterade processer
- Ett barn kan inte ha högre rättigheter än sin förälder: den första processen som startas vid systemstart är ”gud” (`init` på Linux, `launchd` på OS X)
- Alla processer har: standard-in, standard-out, standard-err som alla är av typen text
- Man skapar en process genom att starta ett program (eller terminalkommando)
- Man övervakar processer med kommandot `ps`
- Man pausar, startar, stänger av processer o.s.v. med kommandot `kill` (TERM, SIGSTOP, SIGCONT, KILL m.m.)

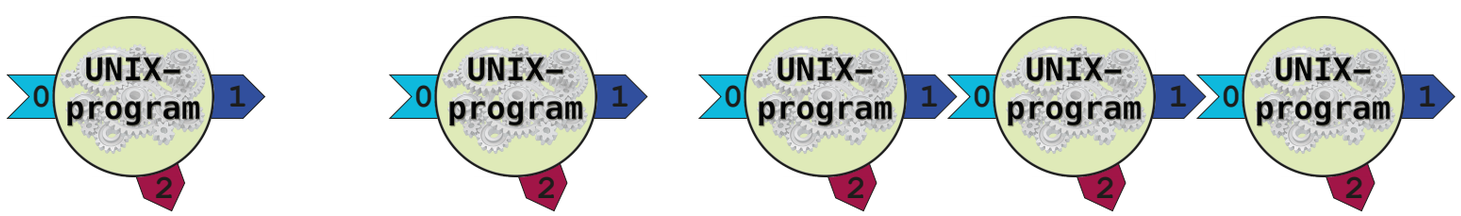
|

program1 | program2 | program3

”Pipe” (rör) är ett sätt att koppla samman program i UNIX. StdOut från ett program används som StdIn för nästa:



På det viset kan man koppla samman flera kommandon i en kedja:




```
less fil.txt | awk '{print $1}’ |  sort  | uniq –c | sort –nr
```

Variabler

- Skalet är programmerbart: alla vanliga konstruktioner finns (lopar, if-satser, funktioner o.s.v.) och även variabler
- Systemet definierar ett antal variabler – användaren kan definiera ytterligare efter sina egna behov
- Variabler är typlösa och kan konverteras efter behov
- Sätt variabler med: `export VAR="Mitt värde"`
- Här en tabell över några vanliga variabler från systemet:

Variabel	Förklaring	Exempel
$PATH	Uppräkning över var binärer finns	`/usr/local/jdk/bin:/usr/local/bin:/usr/local/bin/X11`
$SHELL	Vilket skal man kör	`/bin/zsh`
$PWD	Vilken katalog befinner jag mig i?	`/h/d2/b/peterm`
$EDITOR	Vilken är min default-editor?	`EDITOR=vi`
$LANG	Vad är det för språkinställning?	`LANG=sv_SE.UTF-8`
$0	Det körande scriptets hela rad	`/usr/bin/open_ports.sh -u`
$$	Körande processnumret	`83975`
$?	Resultat från senaste kommando	`0`
$PS1	Promptens definition	`\e[37;41m\d \t ▶ \u@\h:\w$ \e[0m \[\e];\u@\h:\w\a\]`

Ägare

- Alla processer och alla filer har en ägare
- Ägaren tillhör en grupp
- Tre nivåer finns: ägare (u), grupp (g), andra (o)
- För varje nivå finns i filsystemet tre rättigheter: läsa (r), skriva (w), köra (x)

Titta på körande processer:

```
login{peterm}: ps aux | grep "[v]i "
peterm   17642  0.1  0.1 240328 10324 pts/43   Sl+  12:26   0:00 vi .profile
```

Titta på filer:

```
login{peterm}: ls -ls .profile
4 -rwx--x--x 1 peterm cs 3732 Jan 21 12:27 .profile*
```

Hjälp!!

Man får hjälp med kommandona med kommandot `man`: `man ls` visar manualsidan för hur man listar filer.

Alla kommandon har optioner. Exempelvis gör "`-lst`" till kommandot `ls` att filerna visas i en lång listning men den nyaste överst och storleken uttryckt i kByte.

De flesta kommandon är s.k. POSIX-kompatibla och då kan man skriva optionerna i vilken ordning som helst.

Även dessa webbsiter är bra att besöka:
http://stackoverflow.com
http://superuser.com
http://www.commandlinefu.com

Nackdelar??

Konstruktionen med pipe gör kommandomiljön *mycket* kraftfull!

MEN! Kommandona är ofta kryptiska fungerar enligt paradigmet "Remember-and-type". Det finns många hundratals kommandon i ett normalt UNIX-system och det är en utmaning att minnas dem!

Vill man se alla kommandon som finns [i ens PATH] men en kort förklaring kan man köra följande kommandosekvens [i Linux]:

```
echo -n $PATH | xargs -d : -I {} find {} -maxdepth 1 -
executable -type f -printf '%P\n' | sort -u | xargs
whatis 2> /dev/null | egrep '\((1|1m|6|8)\)'
```

För OS X kör man detta:

```
echo ${PATH//:/ } | xargs -J % find % -maxdepth 1 \( -
type f -or -type l \) | xargs basename | sort -u |
xargs whatis 2> /dev/null | egrep '\((1|1m|6|8)\)'
```