Välkommen till

UNIX Grundkurs

Solaris/Aix/SCO/Linux/BSD

Kapitel1
Introduktion och översikt.

- ***Vad kommer du att få lära dig?:***
  - **Unix historia och filosofi**
  - **Dokumentation- Unix reference manual**
  - **Hantering av filer**
  - **Vi editorn**
  - **BASH shell**
  - **Använda filterkommandon och pipes**
  - **Reguljära uttryck**
  - **Shellscripts**
  - **Kommunikationsverktyg (mail,talk)**
  - **Användar Systemadministration**
  - **Grundläggande Nätverksinställningar**

Kursbeskrivning;

- Kap 1 Unix historia samt nuvarande position
- Kap 2 Grundläggande filhantering
- Kap 3 Bash – introduktion
- Kap 4 Unix bibliotekssystem
- Kap 5 Pipeline och filterkommandon
- Kap 6 Bash
- Kap 7 Reguljära uttryck
- Kap 8 Shellprogrammering
- Kap 9 Systemadministration för användare
- Kap 10 Kommunikationsverktyg
- Kap 11 X Window systems

Praktisk Information

- Fika tider
- Telefonkiosk
- Toaletter
- Parkeringsplatser
- Telefonnummer

Presentation av Instruktör

- Bakgrund
- Nuvarande position
- Erfarenhet av Unix
- Utbildning

Deltagarpresentation

- Namn
- Vad gör ditt företag
- Nuvarande position (arbetsuppgifter)
- Erfarenhet av UNIX eller andra operativsystem
- Kommande Unix projekt
- Förväntningar

Kap 1 Operativsystem

Ett operativsystem skall:

- **Övervaka, leda, och fördela datorns resurser, processorer, minne, enheter för in/utdata, samt göra det lätt för användaren att umgås med datorn.**
- **Operativsystem är en programvara som gör maskinen användbar.**

Operativsystemet UNIX

- Fleranvändarsystem
- Flera arbeten på samma gång
- Portabelt operativsystem
- Finns på små och stora system
- Leverantörsoberoende
- Har en mängd hjälpprogram
- Moduluppbyggt
- Hierarkiskt filsystem
- X-windows system

Historien bakom UNIX

1999 — Solaris 7.

1998 — UnixWare 7 - UNIX V.5

1996 — SCO köper Unix av Novell

1995

1994

1993 — Novell köper 82 % av USL.

1992 — UNIX SVR4.2 släpps från AT&T. Novell offentliggör planer på att köpa USL.

1991 — Linux 0.0.1.

1990 — USO blir ett eget bolag och kallas USL (Unix System Laboratories). OSF/1 debuterar.

1989 — AT&T flyttar utvecklingen av UNIX till en separat division (USO-Unix System Operations). AT&T presenterar UNIX SVR4 (System V Release 4) i slutet av året. SVR4 är en sammanslagning av de mest populära UNIX-dialekterna.

1988

1987 — AT&T släpper SV.3 och Berkeley släpper UNIX ¤.3 BSD. AT&T och Sun presenterar plan 2 för sammanslagningen av System V, BSD-UNIX och XENIX (PC-UNIX).

1986

1985 — Sun presenterar NFS. AT&T och Sun presenterar planer på att slå ihop System V och BSD. AT&T Släpper SVID.

1984 — X/Open grundas. /usr/groups förslag till standard presenteras. Blir sen POSIX. AT&T släpper UNIX System V.2

1983 — AT&T introducerar UNIX System V. Från Berkeley kommer UNIX 4.2 BSD.

1982 — Novells första produkt lanseras. Det är en UNIXbaserad dator för ett stjärnnät. Novell byter sedan snabbt operativsystem till DOS. Från Berkeley kommer en UNIX-version som kallas UNIX 4.1 BSD.

1981 — Föreningen /usr/group grundas, liksom företaget SUN.

1980

1979

1978

1977 — Första kommersiellt sålda UNIX-systemet installeras på Rand Corp. SCO och Interaktive System grundas.

1976

1975 — Bell Labs licensierar UNIX till olika universitet

1974

1973

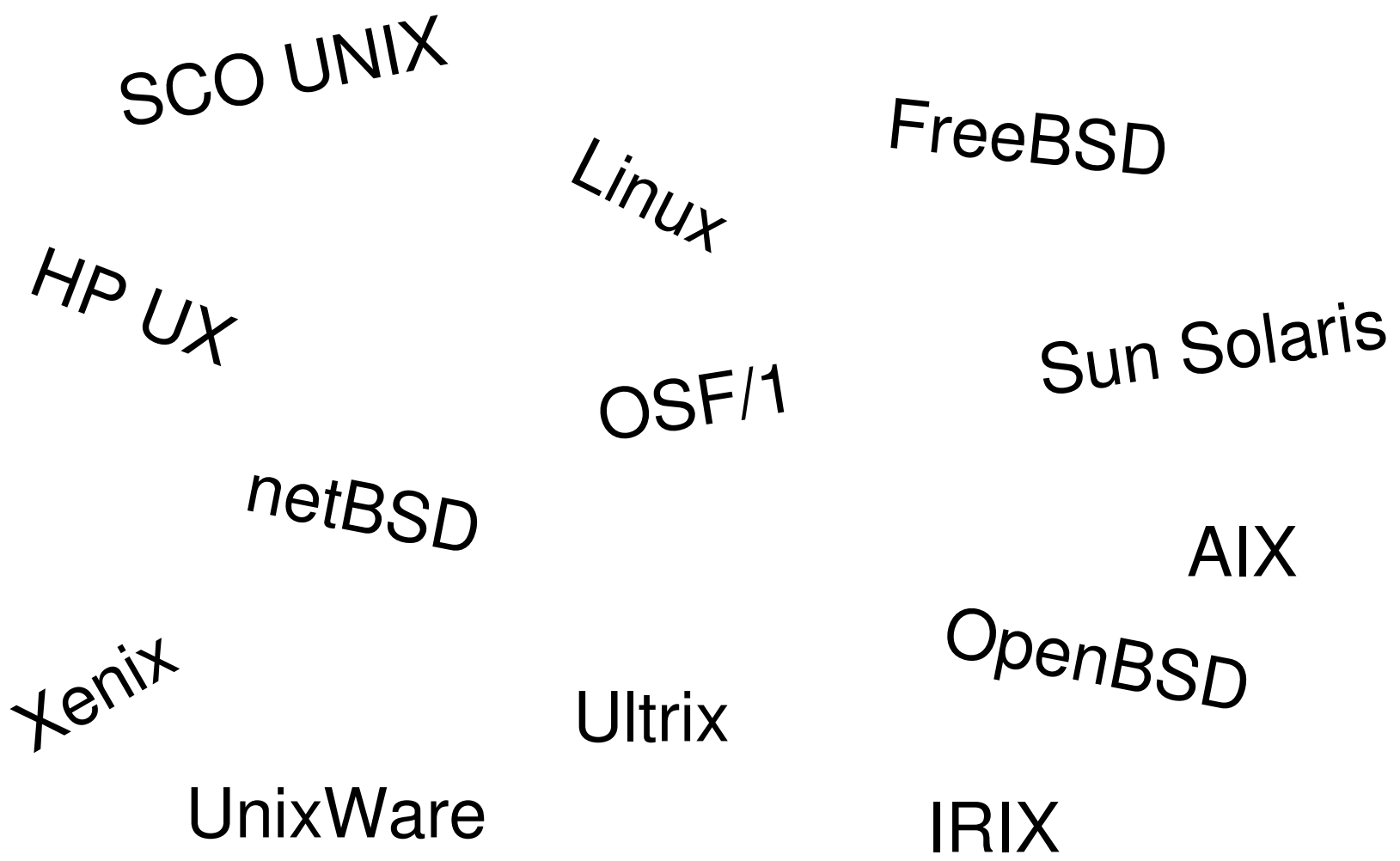
1972

1971 — AT&T:s patentkontor får det första UNIX-systemet som används i drift.

1970 — Operativsystemet döps till UNIX av Brian Kerningham.

Ken Thompson "bootar" den första singel-user UNIX:en.

Olika unixdialekter



Minnesvärda namn

Traditionella Unix dialekter, minnesvärda namn

Ken Thompson,

Dennis Ritchie,

Brian Kerninghan

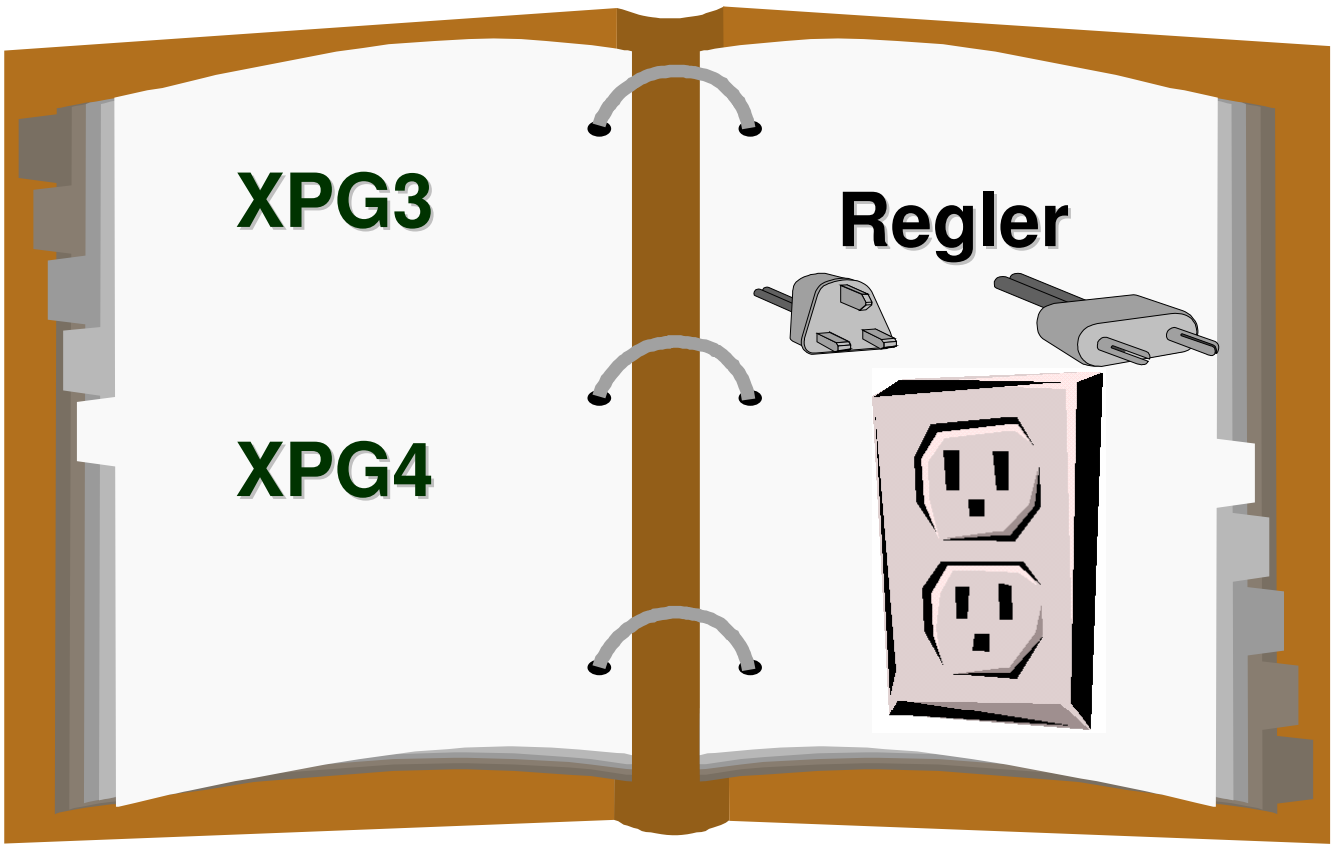
Projekt

Multics som blev UNIX och skrevs om med C kod för att vara portabel.

Linux

Linus Torvalds

X/Open - en standard som anger regler



Unix olika ansikten

Språk i Unix

Shellprogrammering

Awk

Sed

LEX

YACC

Python

Finns tillgängliga

C/C++	**Java**	**Pascal**	**Cobol**
Fortran	**LISP**	**BASIC**	**ADA**

In och Utloggning

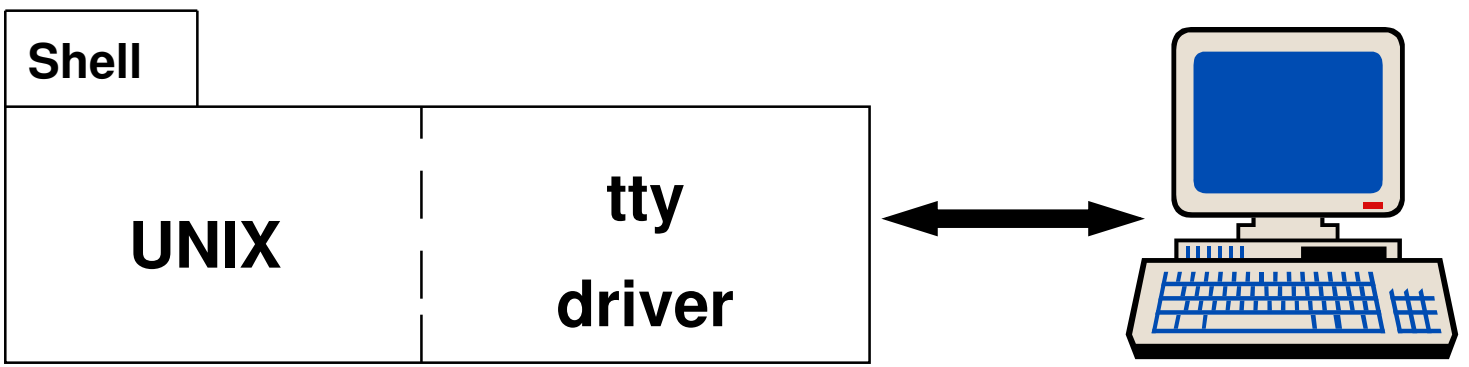
Alla användare har ett användarnamn
Och ett hemligt lösenord

När du loggat in landar du i din hemkatalog
Utloggning sker med kommandot *exit*

tty Driver

- tty driver används för kommunikation mellan Unix och din terminal.
- Så kallade kontrollsträngar styr terminal & applikation
- Benämns tty0 tty1 tty2 osv.
- Kopplingen mellan din terminal och unix
- Kan styras med ”ctrl-knappen” + ”tecken”

  ex) ctrl +c (avbryt) ctrl +d (exit/end of file)



Program och inbyggda kommandon

- De flesta kommandon är program som är lagrade i filer, vissa av dem är inbyggda i shell:

$ date	Visar datum och tid
$ who	Visar inloggade användare
$ rwho	Visar inloggade användare i LAN
$ exit	Loggar ut
$ cd /	Change Directory (byt till root katalog)
$ pwd	Print Working Directory (visar katalogen du står i nu)

Kommando, argument, optioner

- Syntax:
- $cmd [-optioner] [argument]

- -optioner påverkar kommandot.
- Kommandot verkar på sina argument.

Ex visa alla filer i aktuell katalog)

ls –la

Kommando, visa textfiler

- Kommandot *more* visar innehållet av text filer och filernas namn ges som argument.

  $ more filx textfil

- Kommandot *less* visar också innehållet av text filer.

  $ less filx textfil

Manual Sidorna

- *man* kommandot visar UNIX Reference Manual sidorna.
- man ls
- Kommandots manualsida består av åtminstone tre delar:
    - Name: namn och mål med kommandot.
    - Synopsis: syntax- argument, optioner.
    - Description: fullständig beskrivning av kommandot.
- Andra sektioner kan innehålla:
    - Diagnostic
    - See Also
    - Notes
    - Limitation (or Bugs)
    - Files
    - Examples

Forts. Manual sidorna

Manualsidornas indelning i sektioner:

- **man –s <sektion> <manualsida>**
- **UNIX Reference Manual sektioner innehåller:**

Section / Topic	
1	Commands available to users
2	Unix and C system calls
3	C library routines for C programs
4	Special file names
5	File formats and conventions for files used by Unix
6	Games
7	Word processing packages
8	System administration commands and procedures
9	Device drivers

Söka i manualsidorna efter nyckeord:

- **man –k nyckelord**

Kapitel 2

Grundläggande filhantering

- Fil attribut
  - namn, ägare, access rättigheter…

- Filhantering
  - flytta (ändra namn)
  - kopiera
  - ta bort
  - access rättigheter
  - skriva ut fil innehåll
  - jämföra filer
- vi editor

Filnamn

- Alla tecken utom, (slask) är godkända i filnamnet.
- Upp till 255 tecken i filnamnet.
- UNIX skiljer på stora och små bokstäver.
- Filtillägg (.c, .txt .doc) krävs inte av UNIX.
- Filnamn innehållande specialtecken,

  (metatecken) är godkända, men svåra att jobba med.
- Mellanslag kan användas med kvote tecken: ”kalle anka”

Fil attribut

- Utöver data i filen så lagrar UNIX filens attribut som:
  - Filtyp (- l d m s c b t p)
  - Access rättigheter (r w x s S)
  - Ägare (kalle.)
  - Grupptillhörighet (.anka)
  - Storlek (i byte eller block)
  - Ändringstid
  - Senast acessad

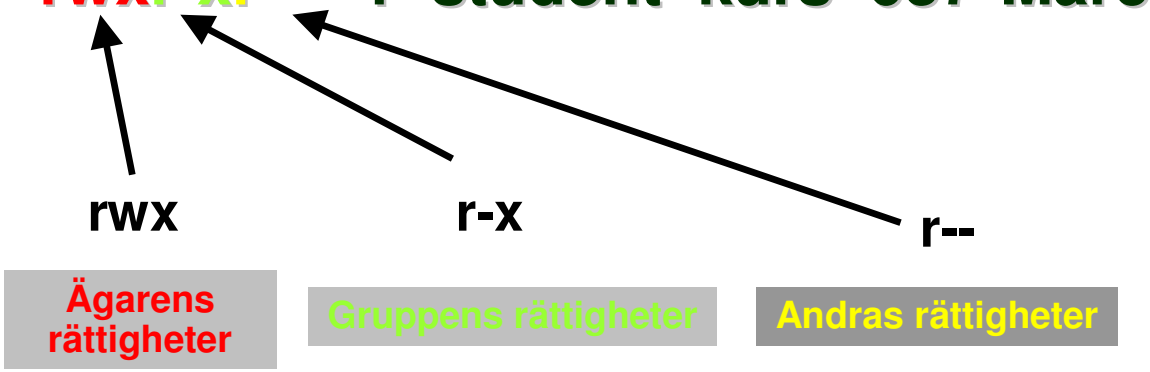
Filtyper

- - vanlig fil
d – katalog fil (directory)
l – symbolisk länk
m – delat minne
b – block fil
c – character fil
p – pipe
s - socket

Filtyper och accessrättigheter

$ ls – list short list visar endast filnamn
$ ls –l long list filnamn + attribut
$ ls –ld listar katalogfilen, ej innehållet

rwxr-xr-- 1 student kurs 037 March10 9:45 xxx



r - läs rättighet
w - skriv rättighet
x - exekverings rättighet
- - ingen rättighet

Kopiera filer

$cp fil1 fil2

Skapar en kopia av fil1 som kallas fil2.

- **Om fil2 redan existerar så skrivs den över.**

Följande krav skall vara uppfyllda:

1/ sökrättighet (x) för samtliga kataloger ovanför fil1 och fil2.

2/ läsrättighet för fil1.

3/ skrivrättighet för fil2 om filen existerar.

4/ *eller* skrivrättighet för katalogen som skall innehålla fil2.

Flytta filer- Ändra namn

$ mv fil1 fil2

- **Ändrar namnet från fil1 till fil2.**
- **Om fil2 redan existerar så tas den bort.**

$ mv fil1 fil2 fil3 dir

- **Flyttar fil1 fil2 fil3 till filkatalogen.**

Länkade filer- hårda länkar

- Varje filnamn kallas länk.
- Länkar kan användas för att:
  - Dela filer mellan olika användare.
  - Tillhandahålla korta namn för långa filnamn.

$ ln fil1 fil1-link

  - Skapar nytt namn fil1-link till existerande fil fil1.
  - Om filen fil1-link redan existerar så tas den först bort.

$ ln samba samba-link

  - Hårda länkar kan inte peka till katalogfiler.
  - Hårda länkar kan inte peka över filsystems gränserna.

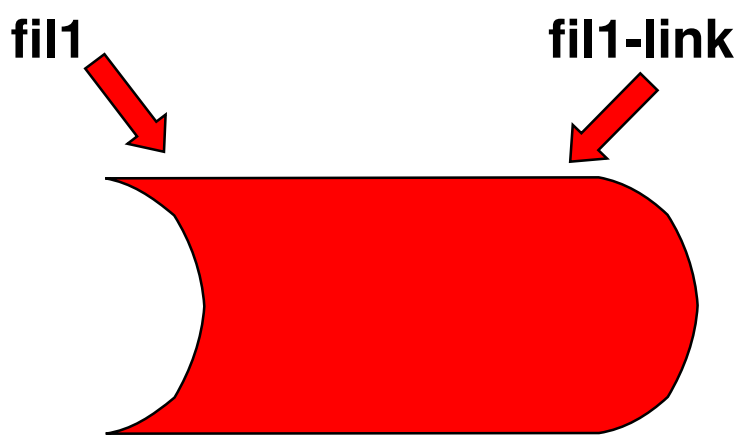
forts. Länkade filer- hårda länkar

- **För att lyckas skapa en länkfil så krävs:**

1/ Sökrättigheter (x) för alla katalogfiler ovanför fil1 och fil1-link.

2/ Skrivrättigheter för katalogen innehållande fil-link.

- **Länkfilen är endast ett nytt namn för den redan existerande filen fil1.**



Fil index (i-node) nummer

- **Alla hårda länkar är likvärdiga. Det finns ingen ”master-link”.**
- **Nya länkar har samma data och attribut som redan existerande länkar.**

$ls –li listar i node nummer

total 379

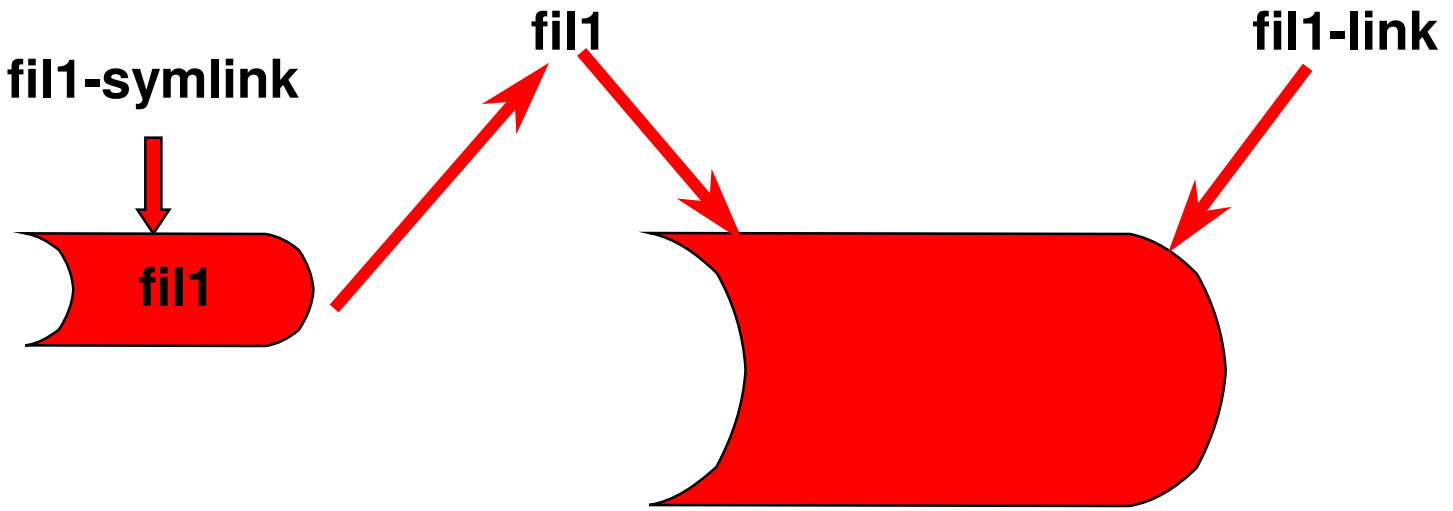
427 rw-r—r– 2 student1 kurs 307 March 10 9:46 samba

427 rw-r—r– 2 student1 kurs 307 March 10 9:46 samba-link

Symboliska länkar

$ ln –s fil1 fil1-link

Skapar en symbolisk länkfil som pekar till fil1.



fil1 är en ”master-link”

fil1-symlink kan peka till katalogfil

fil1-symlink kan peka över filsystemgränserna

Ta bort (avlägsna) filer

$rm fil1…

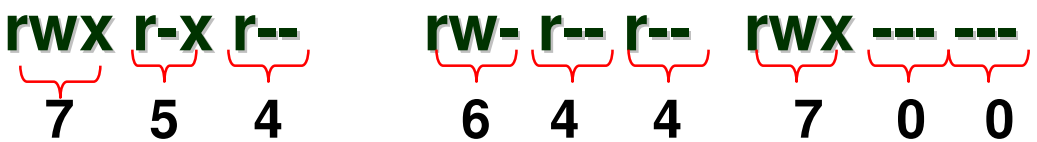
- **Tar bort länkfilen fil1….**
  - **När sista länken försvinner så raderas filen, dvs innehållet är borta.**
  - **Ingen ”unrm” kommando.**

$rm –r dir

- **Tar bort katalogen (dir) och hela strukturen under (filer och kataloger).**

Ändra åtkomst (access)rättigheter

- åtkomsträttigheter är lagrade som bitar:
  - On- åtkomst tillåten
  - Off- ingen åtkomst
- Varje grupp med tre bitar kan representeras oktalt:



- Chmod kommandot ändrar åtkomsträttigheter för filen du äger:

$ chmod 640 brev (text fil)

$ chmod 755 Brev (bibliotek fil)

Ändra åtkomst (access)rättigheter

u –user	+ lägg till	r –läs
g –group	- ta bort	w –skriv
o –other	= absolut värde	x –exekvera
a – alla		

$ chmod ug+w katalog

Fil jämförelse

- **Kommandot *cmp* jämför två filer**
  - **Textfiler, datafiler, programfiler**
  - **cmp visar inget om filerna är identiska.**
  - **I annat fall visar cmp den första skillnaden.**

$ cmp /etc/passwd /etc/passwd-

Fil jämförelse

- **Kommandot *diff* visar skillnaderna per rad mellan två textfiler.**

$diff brev1 brev2

Visar vad du skall göra med brev2 så det blir identiskt med brev1.

Utskrift av filer

Skriv ut:

$ lp brev1 brev2 # system V

$ lpr brev1 brev2 # BSD

Kolla skrivarkö:

$ lpq

Rensa i skrivarkö:

$ lprm <job id>

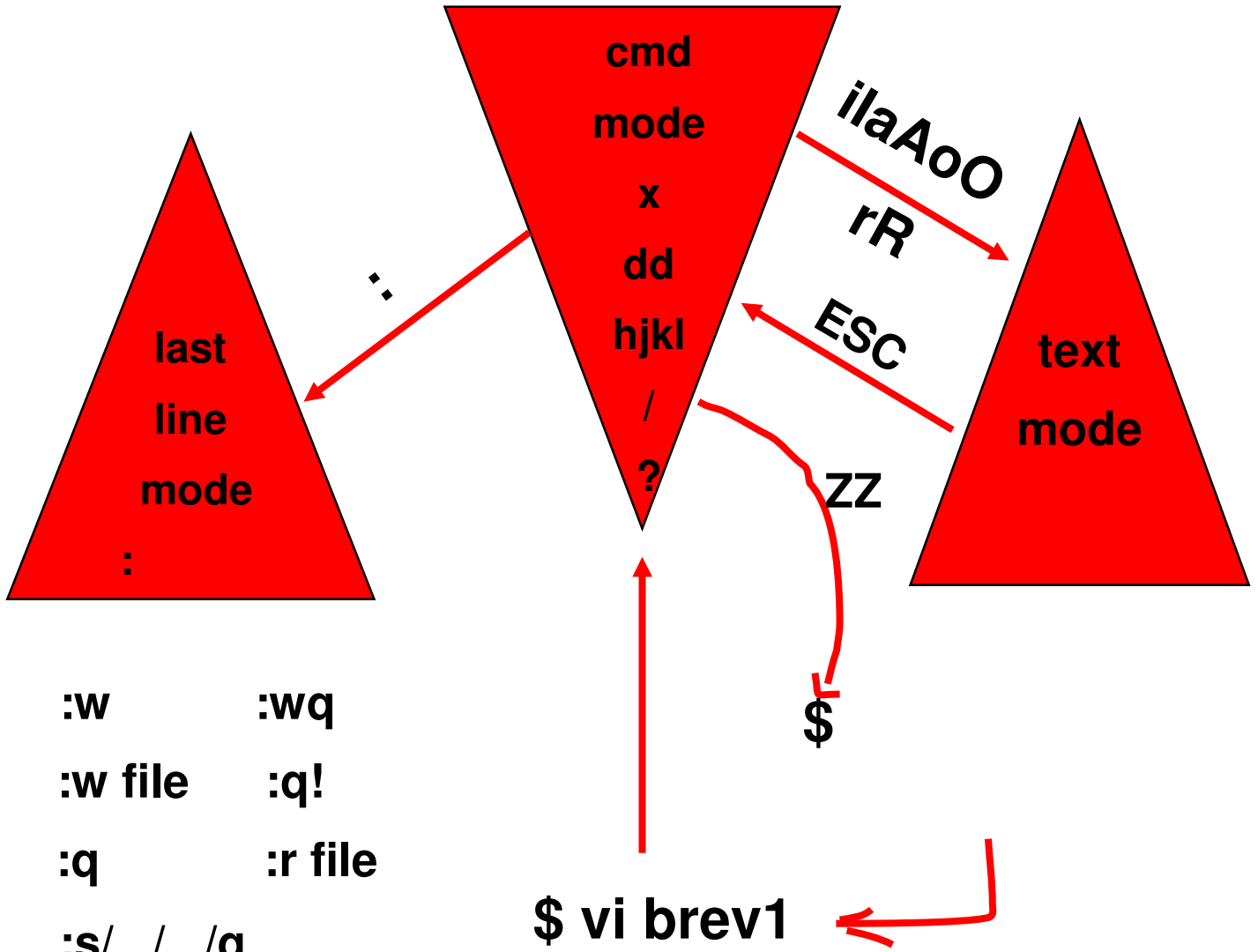
$ cancel <job id>

Kriva ut till viss skrivarkö:

$lp –d printer brev1

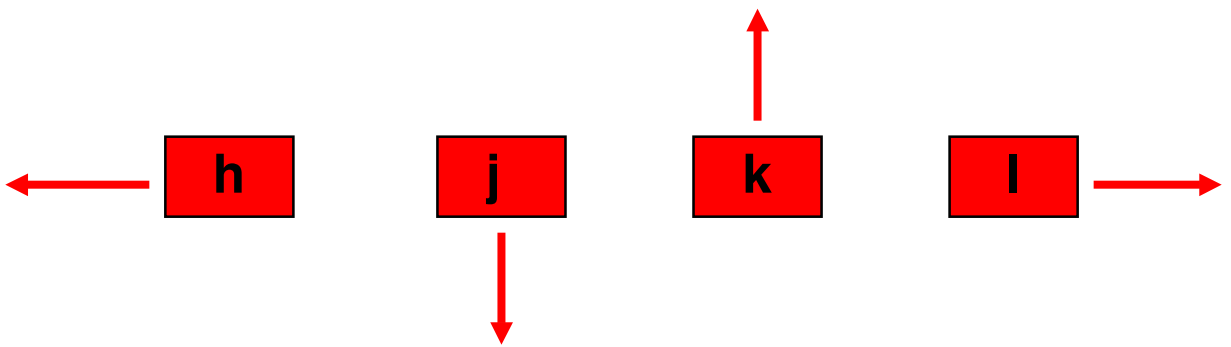
$lpr –p printer brev1

Vi- editor


cmd
mode
x
dd
hjkl
/
?
ilaAoO
rR
ESC
:.
ZZ
last
line
mode
:
text
mode
:w
:w file
:q
:s/.. /.. /g
:wq
:q!
:r file
$
$ vi brev1

Vi-editor

- Pil tangenterna kan användas för att flytta markören.
- När Vi utvecklades så saknade många terminaler piltangenter.



- För större förflyttningar så använder man:

ctrl F	- hel skärm framåt.
ctrl B	- hel skärm bakåt.

Vi-editor

- För att söka efter textmönster använd:
  - /textmönster - söker framåt från markörens position
  - n – nästa förekomst (framåt)
  - N – nästa förekomst (bakåt)

- ? textmönster -söker bakåt från markörens position
  - n – nästa förekomst (bakåt)
  - N – nästa förekomst (framåt)

- x – tar bort tecken under markören.
- dd – tar bort raden.
- u – undo (ångra)
- . – redo (repetera)

Kapitel 3

BASH Introduktion

- **BASH underhåller history lista av tidigare exekverande kommandon.**
- ***history*** **visar alla utförda kommandon.**

```
$history
1 cp fil1 fil2
2 ls –l
.
.
.
115 history
```

- **$ history 50 visar de 50 sista kommandona**

Kommando history

BASH stöder direkt bläddring i kommando history-list

kommando editering:

PILTANGENTERNA –flytta upp och ner i history lista.

PILTANGETNERNA – flytta vänster, höger på raden.

Delete, backtab, insert, tangenterna fungerar.

Output (utdata) omdirigering

- tkn > omdirigerar standard output device (skärmen) oftast till en fil.
- utdata från programmet går till en fil.
- Felmeddelanden (om det är några) visas på skärmen.
- > är ett shell metatecken.

$cal 03 1969 > March _69

$more March _69

$history –50 > sista _50

$more sista _50

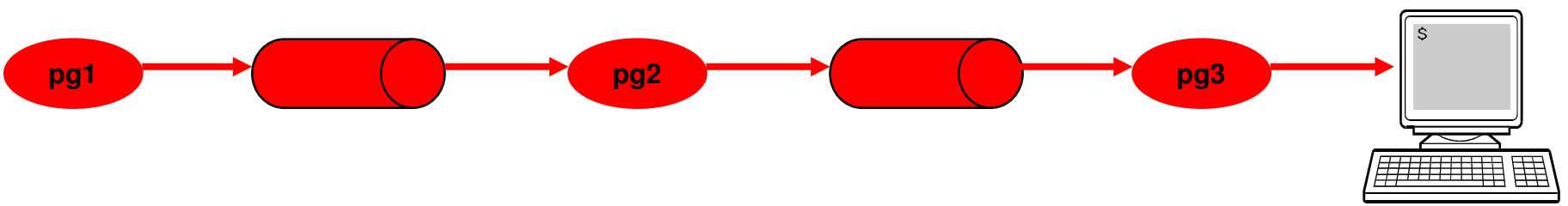
Output (utdata) omdirigering

> - skapar utfil om den inte redan existerar.
- skriver över filen om den existerar.

>> - skapar en utfil om den inte redan existerar.
- lägger till i utfilen om den existerar.

Pipes

- Pipe symbolen | sammanbinder programmen stdout fil med ett annat program, stdin fil.



$program1 | program2 |program3

- program2 och program3 måste kunna läsa stdin (filter)

Filnamn metatecken

- **Speciella tecken i filnamn inkluderar *bl.a:***

  *** - matchar vilka tecken som helst (0 eller mer).**

  ? – matchar ett enda tecken.

  […] – matchar ett tecken i set.

Gruppering och sekvens av kommandon

- Kommandon kan skrivas efter varandra:
  $cd / ; ls ; pwd > where

- Komandon kan grupperas med hjälp av ()
  - $(cd/ ; ls ; pwd) >> where
  - $pwd

- Gruppering av kommandon (pg1;pg2;pg3) startar ett subshell där alla kommandon exekveras före de som står utanför ().

Metatecken

- **Använder du ; ( ) > >> och ; så spelar det ingen roll om du använder mallanslag eller ej.**
- **Använder du ? * [ ] så påverkas resultatet om du inte använder mellanslag.**
- ***Prova:***

  $ls –la > list_fil ; more list_fil

  $ls –la>list_fil ; more list_fil
- ***Prova:***

  $ls [bB]*

  $ls [b B] *

Bakgrundsprocesser 1

- Förgrundsprocesser väntar tills cmd/program avslutas innan prompten visas igen.
- Bakgrundsprocesser startas av shell och prompten visas direkt igen.
- Program som kan startas i bakgrunden är:

  1/ De som använder grafiskt gränssnitt.

  2/ De som inte behöver terminal I/O.
- För att starta ett program i bakgrunden använd & efter programnamn.

```
$sleep 1000 &
 [1] 234
$
```

Bakgrundsprocesser 2

- sleep job nummer och process ID visas och man får prompten tillbaka. 1 = Jobbnummer och 234 är process ID
- Job nummer är tilldelat av shell.
- PID nummer är tilldelat av systemet.
- jobs visar dina bakgrundsjobb
- kill %<job nummer> avslutar bakgrundsjobb ovillkorligen

Process status

- Ett jobb kan ha flera process ID, tilldelade av systemet
- ps utan argument och optioner visas endast aktuellt shells processer:

```
$ps
PID        TTY          TIME         CMD
361        pts/01 0:00          bash
373        pts/01 0:01          sleep
```

- För att se alla pågående processer:

```
$ps –ef          # SystemV
$ps –aux         # RedHat(linux) / BSD med flera
$ps –aux | more  # sidvis visning av listan
```

Avsluta processer

- Användaren kan kommunicera med förgrundsprocesser med hjälp av ctrl-tecken.

  $stty –a - Visar terminalens inställningar.

- Ctrl+C - Skickar en INTerrupt signal.
- Ctrl \ - Skickar en Quit signal.

  $sleep 100

  Ctrl+c

- För att avsluta systemprocesser använd kill kommandot. Flaggan –9 dödar allt. (1-15 finns)

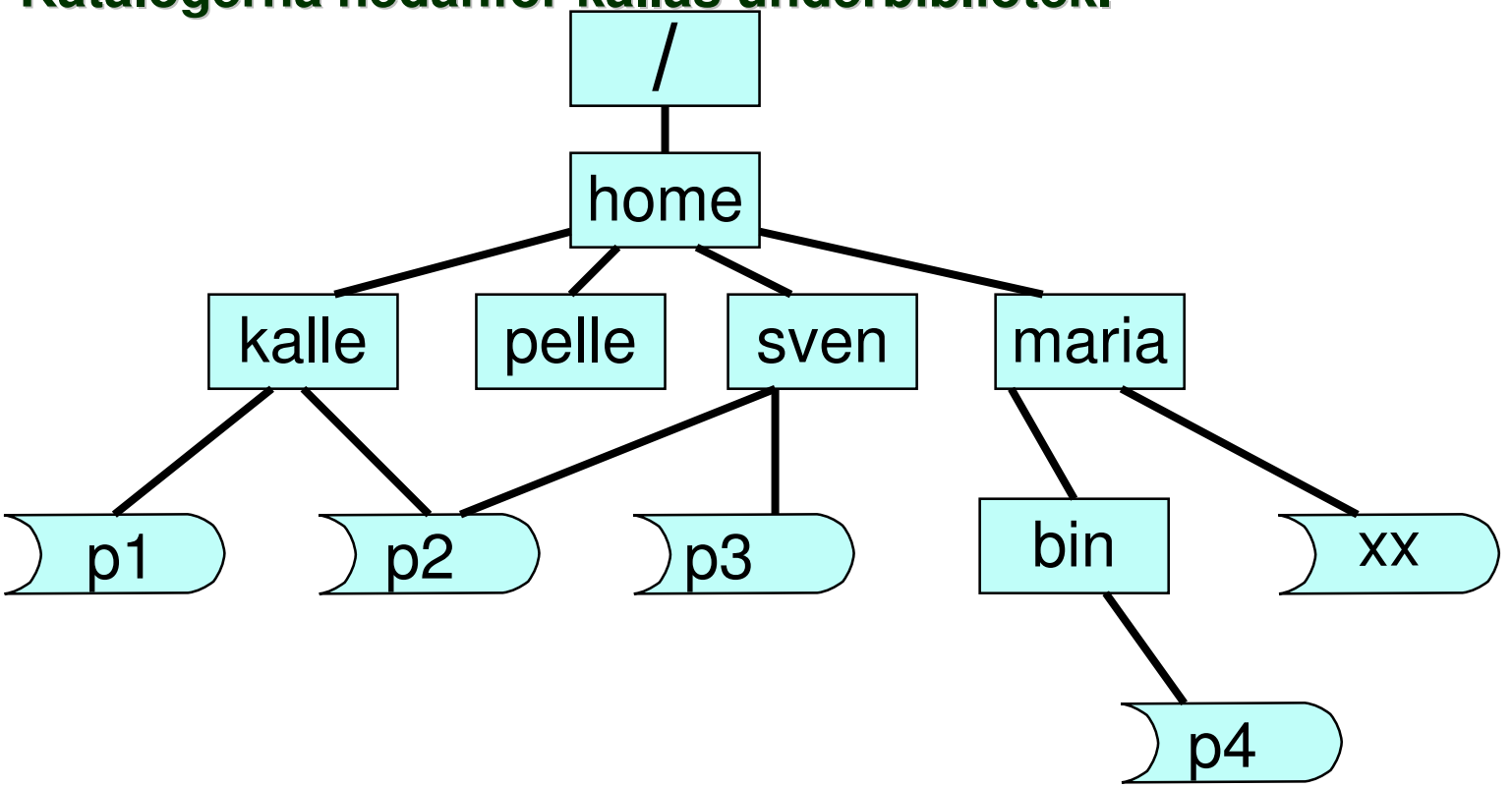
  $ kill –9 PID

  $ kill %jobbnr

Kap 4

UNIX bibliotek struktur

- **Biblioteket är organiserat hierakiskt.**
- **Högst uppe finns root, /, bibliotek (directory).**
- **Katalogerna nedanför kallas underbibliotek.**



Pathname

- Lista av namn separerade med snedstreck (slash) / kallas för pathnamne.
- Pathname som börjar med / kallas för absolut pathname:

Exempel:

/

/home

/home/kalle/p1

- För att se innehållet i filen p1:

  $ more /home/kalle/p1
- För att kopiera p1 till p1.bkp:

  $ cp /home/kalle/p1 p1.bkp

Home och CWD

- **Varje UNIX process har Current Working Directory (cwd).**
- **Varje användare har home directory.**

  $HOME

  cd <retur> **cd ~namn**
- **Pathname som inte börjar med / kallas för relativt path (mot cwd).**
- pwd **visar nuvarande katalog**

Relativt pathname

- Varje bibliotek innehåller två filer:

. är en fil som refererar till aktuellt bibliotek

och

.. är en fil som refererar till biblioteket ovanför (parent directory).

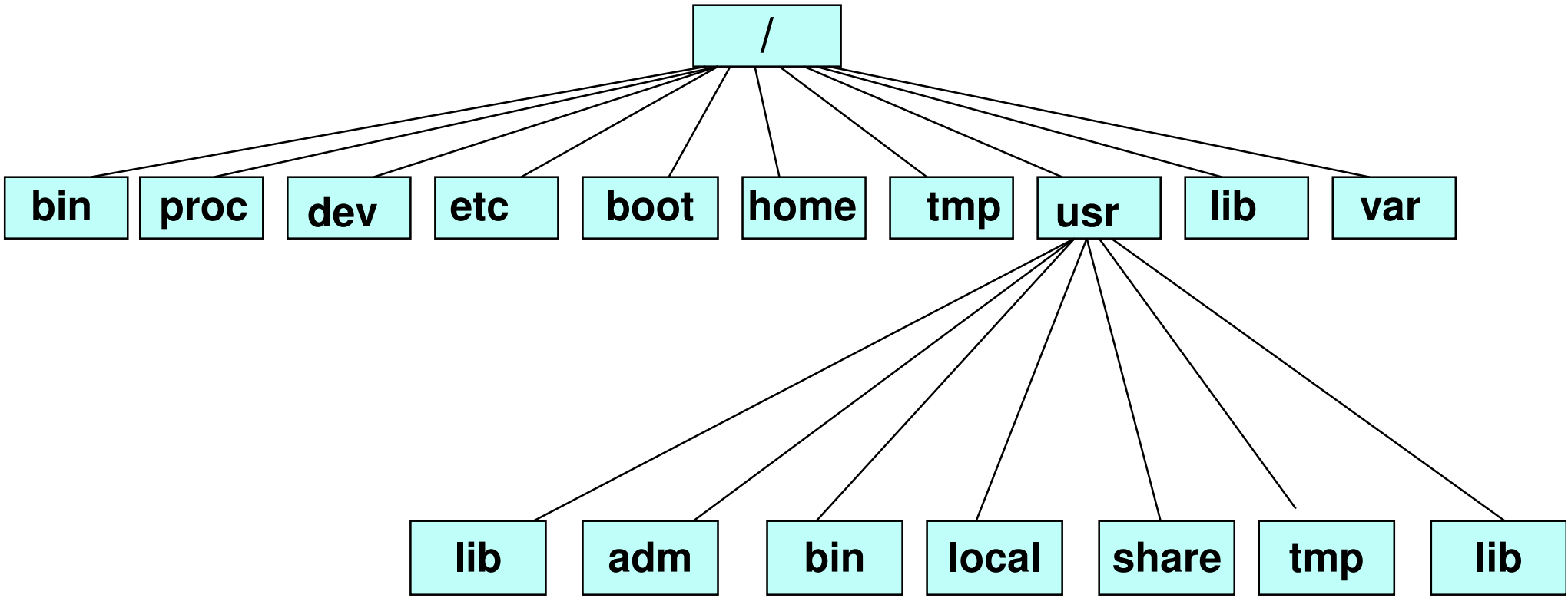
$ ls ../.

$ ls –l ../../kalle

Lista filer i ett bibliotek

- ls kommandot visar namnet på alla filer utom de som börjar med en punkt.
- ls -a option visar alla filer, *tom* .filer
- kan också användas med andra optioner.
- ls –l visar långt listformat
- ls –1 visar kort lista i en rad

Typisk struktur i ett UNIX bibliotek



Innehåll i typiska UNIX bibliotek.

- Unix bibliotek är nästan samma på den första nivån.

/bin -länk till /usr/bin

/usr/bin - användar kommandon

/usr/X - X-Windows (X11)

/usr/local - egna kommandon

/usr/share - delade filer (man sidorna)

/usr/tmp - temporärt utrymme

Forts. innehåll i typiskt Unix bibliotek.

/usr/lib - olika librarys (ex. för C-språk)

/dev - device drivers

/etc - systemadministration and och filer

/export - delade bibliotek och filer

/home - hemma bibliotek

/lib - länk till /usr/lib

/tmp - temporärt utrymme

/var - filer med varierande innehåll

Bibliotek kommandon

- pwd (print working directory) visar CWD.
- cd –change directory.

```
$pwd
/home/kalle
$cd /etc
$pwd
/etc
```

Bibliotek kommandon

$mkdir dir…	- skapar nya dir…
$rmdir dir…	- Tar bort dir…om de inte innehåller filer.
$rm –rf dir…	- Rekursivt tar bort alla dir…, subdirectorys samt alla filer. FARA UTAN ATT FRÅGA!!
$mkdir –p dir/dir/dir	- skapar även mellanliggande dirs…
$mkdir dir/ dir/dir2	- skapar dir och dir2 i dir separat

Inspektera olika typer av filer

- ***More/less*** **kommando visar innehållet för text filer men inte för bibliotek, data eller liknande filer.**
- ***file*** **kommandot ”gissar ” filens innehåll.**
- ***ldd*** **inspekterar en körbar fil och rapporterar beroenden**

Exempel)

$ file . /etc/group brev

.: directory

/etc/group: ASCII text

brev: ASCII text

$ ldd /bin/ls

/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), stripped

Disk usage kommando

- *du* kommando visar antal använda diskblock för filer och bibliotek.
- Utan argument visas antal använda block för filer och bibliotek i CWD.
- du -sh visar summary (total) h=human readable(Megabyte,Gigabyte,Kilobyte..)

Hitta filer

- ***find***
  - Genomsöker biblioteks struktur och försöker hitta filer som uppfyller vissa kriteria.

    Ex: namn, ägare, typ, ändringstid, access rättigheter osv.
  - Utför operationer på hittade filer.

    Ex: find / -name brev -Söker i root bibliotek (och i hela strukturen under) efter filer som heter brev samt visar sökvägen till filerna.
  - Ex: find / -type d –name ungk1 –exec chmod u+rwx {} \;

Att hitta filer

- **Syntax**

  find pathname-lista uttryck

  - **pathname listan definierar vilka bibliotek som skall sökas igenom (kan bli flera)**
  - **Uttryck definierar :**

    - vilka attribut som man är intresserad av, ex: -name filer.

    - vad man skall göra med matchande filer, ex: -print.

(Att) hitta filer

- **Attribut som man kan definiera:**

  -name file **-inom n**

  -user namn **-mtime N**

  -type c **-perm octal**

- **Vad skall man göra med hittade filer?**

  - print

  - exec kommando

Extempel på att hitta filer -find

$find /user kalle

$find /home –type l

$find . –mtime –7

$find /usr –perm –002

$find /usr –inom 4752 –exec rm { } \;

$find .. –name brev –exec ls -l { } \;

Att hitta filer- find med flera villkor

- Man kan sätta ihop olika attribut:
  - -name brev –user kalle (and)
  - -name xx.c –o –name brev (or)
  - ! –user root (not)

- -user kalle \(-name xx.c –o –name 'x.c' \)

Kapitel 5

Pipe och filter kommandon

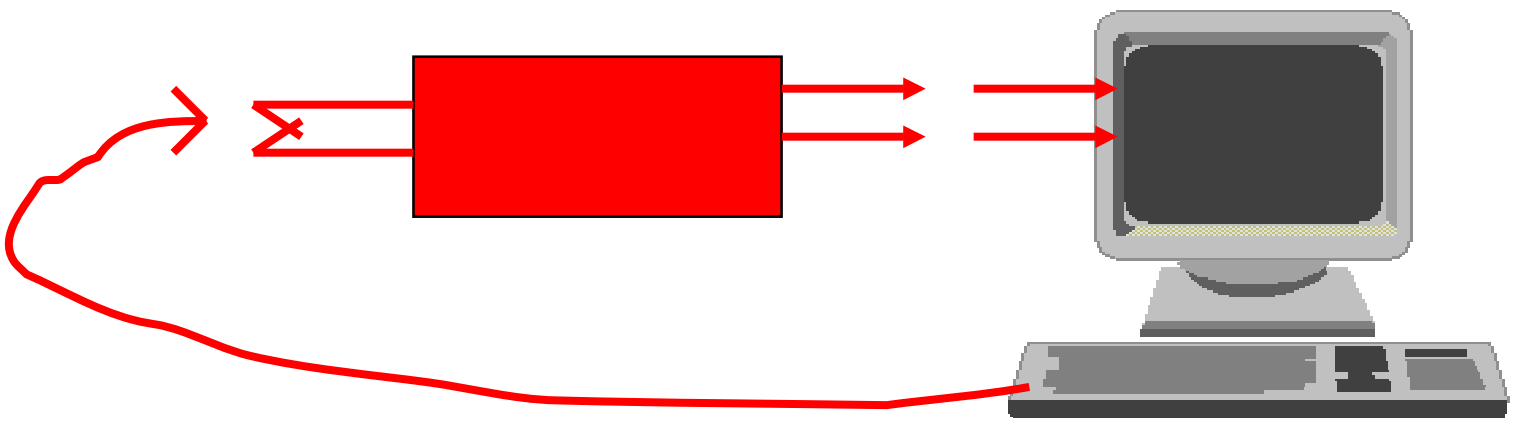
- **Unix filosofi**
  - **Varje program borde göra bara en sak, och göra det bra.**
  - **Komplexa uppgifter kan läsas genom att man kopplar ihop enkla verktyg.**
- **Om möjligt så bör utdata från ett program kunna tas emot som indata till ett annat program.**
- **Filter program bearbetar dataström och kan kopplas ihop med hjälp av pipes.**

Filter kommando

- **Filter**
  - **Läser inputström**
  - **Bearbetar data**
  - **Skriver utström (outputström)**

Standard dataströmmar

- Standard inputsteam 0 - default tangentbord
- Standard outputstream 1 - default skärm
- Standard error stream 2 - default skärm



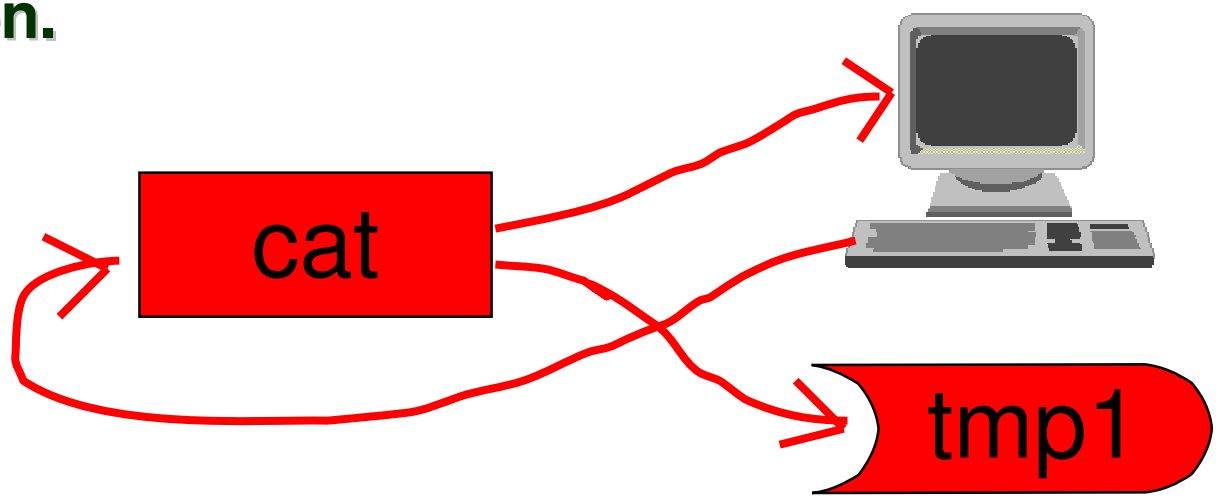
Cat filter – output omdirigering

- *cat* kommandot läser standard input (tangentbord), utför ingen bearbetning och skriver på standard output (skärmen)
- Ctrl-D (eof- end of file) avslutar input stream.

```
$ cat > tmp1
```

Det som användaren skriver på tangentbordet hamnar i filen tmp1. Om filen inte existerar så skapas den.

```
^D
```



Cat filter –input omdirigering

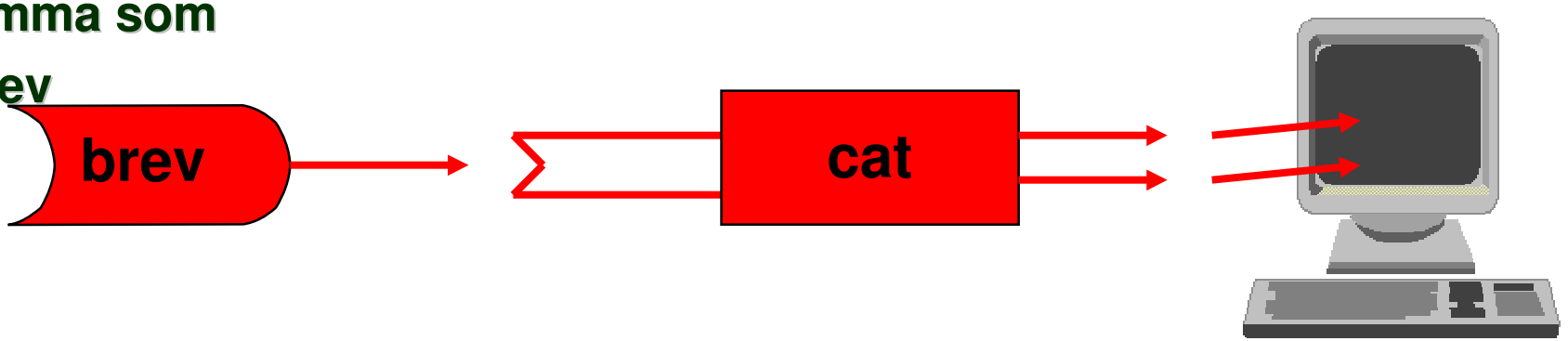
•Omdirigera till bildskärmen är standard för de flesta kommandon

Exempel)

cat < brev

Är samma som

cat brev



Concatenate/Lägg till på slutet av fil

- Append to the end ”>>”
- cat >> tmp1

  Denna text kommer att läggas till i slutet av filen temp1. Om filen inte existerar kommer den att skapas.

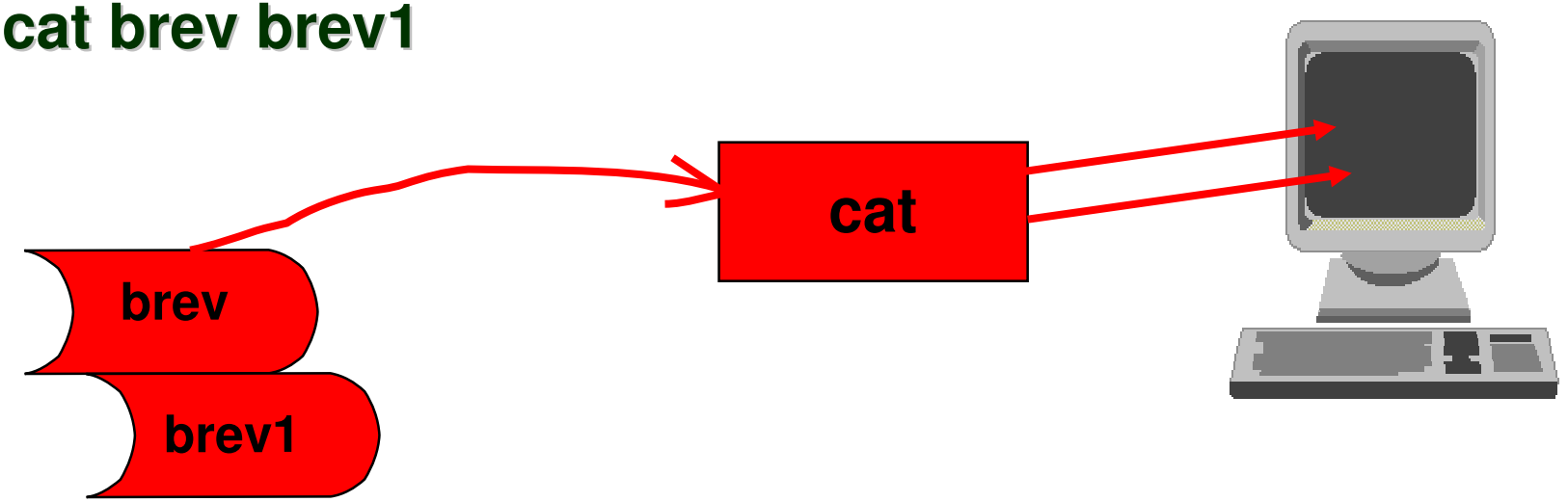
  ^D

  $

Filnamn argument

- Om man skriver fil namn som argument till cat
så läser cat från filen istället för från standard input.

$ cat brev brev1



Slå samman eller bara visa: cat

- Står för catenate – slå ihop.
- Kan användas till att:
  - Visa innehåll i filer.
  - Skapa nya filer.
  - Lägga till info i redan existerande filer.
  - Slå ihop filer.
  - Kopiera filer.

Sort - sortera

- sort – sorterar rader i ASCI ordning och skriver till standard output (skärmen)
- Optioner

  -f - ändra stora bokstäver till små.

  -r - reverse ordning (omvänd ordning)

  -n - sortera numerisk

  -t - fält separatorer

  -o - output fil (utfil)

  + pos -pos börja med fältnr +pos (räknas från 0), sluta med fältnr –pos.

grep- sök efter textmönster

- syntax:

  grep textmönster fil…
- grep söker igenom filer efter rader som innehåller textmönster.
- skriver matchande rader till standard output.

$ grep Kalle Svensson namn

$ grep ’Kalle Svensson’ namn

- -i skiljer inte på stora och små bokstäver.
- -n visar radnummer.
- -v skriv ut ej matchande rader

wc- räkna

- wc räknar rader, ord och tecken.

```
$ wc /etc/passwd /etc/hosts
  97     161    4875 /etc/passwd
126     413    4924 /etc/hosts
223     574    9799 totalt
```

- Optioner

-l - lines (rader)

-w - words (ord)

-c - char (tecken)

Visa början och slutet av filer tail, head

- tail skriver dom sista 10 raderna till stdout.

$ tail brev

- För att skriva dom 25 sista raderna:

$ tail –25 brev

- head skriver dom 10 första raderna till stdout.

$ head brev

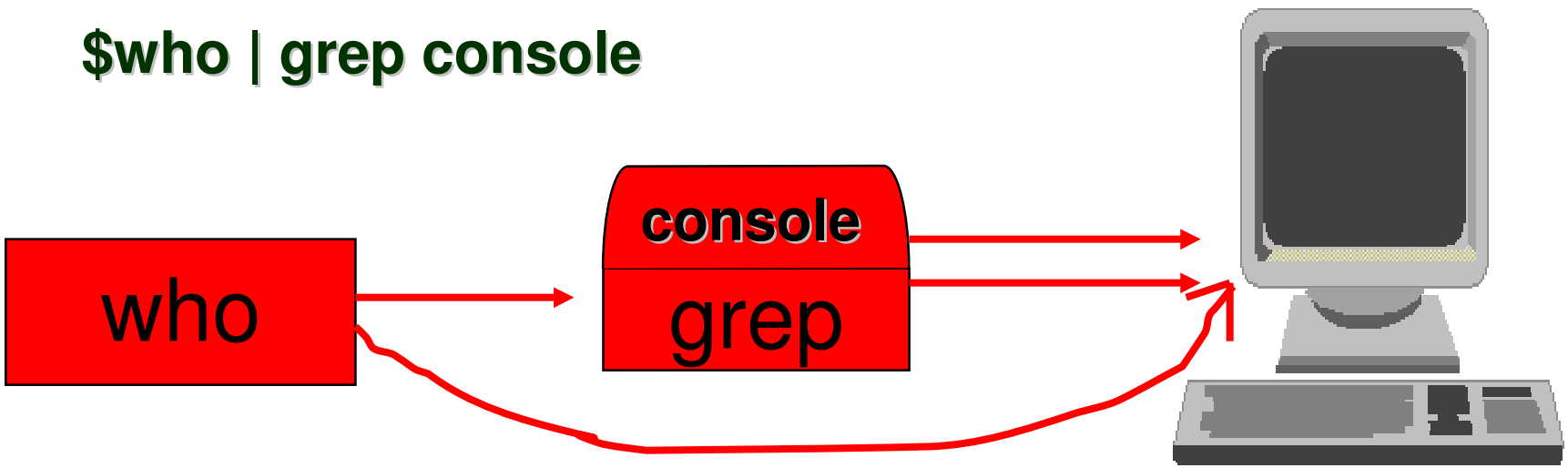
- För att skriva dom 25 första raderna:

$ head –25 brev

Koppla samman kommandon med: Pipe

- Pipe symbolen (|) kopplar stdout filen av föregående kommando med stdin filen av nästa kommando.

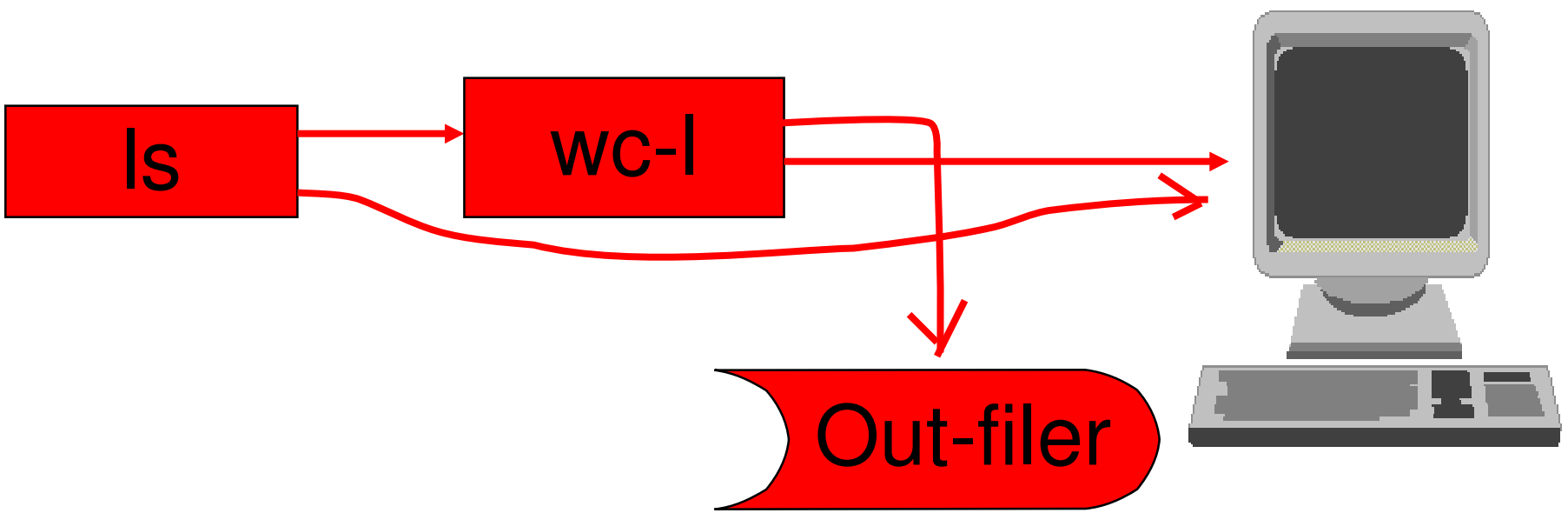
$who | grep console



Pipe

- **Utdata från pipeline kan omdirigeras till en fil:**

$ls | wc –l > out-filer



sed- stream editor

- ***sed*** **kopierar stdlin eller filer till stdout och implementerar editeringskommandon för definierade rader.**

- **sed innehåller en mängd kommandon:**

q – guit	**p- print**
d – delete	**i - inserte**
s – substitute	**g- global**

sed –stream editor

- substitute kommandot byter textmönster mot ett annat textmönster.
  $ sed s/Unix/UNIX/ U_history

- för att byta alla förekomster på raden:
  $ sed s/Unix/UNIX/g U_history

Fältbearbetning - awk

- ***awk*** **betraktar varje inrad som en sekvens av fält separerade med mellanslag (default).**

- **Fält kan selekteras med nummer:**

$awk ’{ print $2 }’

Bill Gates

Gates

Bil Gates, Jr.

Gates,

^D

Fältbearbetning - awk

- awk används ofta för att omformattera utdata från kommandon.

$ date

$ date | awk '{ print $2 $3 $6 }'

Appril192001

$ date | awk '{ print $2, $3, $6 }'

April 9 2001

$ date | awk '{ print $2, $3, ",", $6 }'

April 9, 2001

$ awk –F: '{ print $5 "login namn : " $1 }' /etc/passwd

awk

- awk är ett fullständigt programmeringsspråk.
  - Stöder inbyggda och definierade variabler.
  - Sträng och aritmetiska operationer.
  - Loopar
  - Funktioner
- Awk används ofta för selektering och ändring av fältordning.

- Boken: ”The AWK Programming Language” av Aho, Weinberger, Korningham innehåller en fullständig beskrivning av awk.

Kapitel 6

BASH del 2

- **Bash fungerar på följande sätt:**
  - **Visar prompt.**
  - **Läser kommandoraden.**
  - **Utför tolkning av speciella tecken.**
  - **Exekverar kommandon.**
  - **Visar prompt igen.**

Tilde tecknet

- Tilde tecknet (~) tillhandahåller referens till olika biblioteks (directorys) pathnamn.

~ - referens till mitt hembibliotek.

~kalle – referens till kalles hemma bibliotek.

$echo ~ ~kalle

$ls –l ~kalle

$cmp ~kalle/brev ~/brev

Kommando ersättning

- **Om man vill söka igenom de senast tio ändrade filerna efter rader innehållande Unix:**

$ls –1t | grep Unix | head

Vill man söka efter nyckelord kan man skriva:

$ls –1t | grep Unix | grep ungk1

- **Det finns mer praktiska vägar och mer resurssnåla.**

Kommando ersättning

Back-kvot tecknet ` används för att exekvera kommandon i söksträngar.

- **grep Unix ` ls –1t | head `**
- **Shell exekverar kommandon i backquote (` `) först och ersätter det med dess värde.**
- **echo ”Antal filer: `ls –1 | wc –l`” skriver ut:**

Antal filer: 118

Detta är mer resurssnålt än att sända till grep.

Dessutom aningen snabbare

Kommando ersättning

- **Standard output av vilket kommando som helst kan ersättas med dess värde och tillhandahållas som argument till ett annat kommando.**

$ls –l

$echo I dag är **`date`**

$echo I dag är **`date | awk '{ print $2, $3, $6 }'`**

Shell variabler

- **Syntax**

  Variabelnamn= variabelvärde

  db=/usr/local/bin

- **OBS!**
  - **Inget mellanslag varken före eller efter =.**
  - **$variablenamn används som reference till variabelvärde.**
  - **Vit tilldelning av variabelvärde används inte $**

  cd $db

  pwd

  /usr/local/bin

Speciella variabler

- Använd set kommandot för att se shell variabler.

$set | more

- Många variabler har speciell betydelse för BASH shell.

HOME – hemma bibliotek

LOGNAME – användarens login namn

PATH – lista av bibliotek som skall sökas igenom efter kommandon/program.

PS1 – första prompten

PWD – innehåller aktuellt bibliotek

SHELL – Shell programmet

Speciella variabler

- **PATH – innehåller en lista av biblioteken som skall sökas igenom efter program, och bestämmer dessutom ordningen av sökningen.**

echo $PATH

- **Man kan ändra PATH:**

PATH=$PATH:/usr/local/bin:

- **Man kan även modifiera PS1:**

PS1=' $PWD >'

Globala variabler

- **Shell har tillgång till både lokala och globala variabler.**
- **Endast globala (evironment) variabler är tillgängliga för pgm/subshell som startas från shell.**
- **Kommandot *env* visar globala variabler:**
- **$ env | sort | more**

Globala Variabler

- **Globala variabler innehåller bl.a:**

MAIL – e-mail ”box”

TERM – en typ av terminal

MANPATH – talar om för *man* kommandot vilka

bibliotek som skall sökas igenom. (finns inte i alla unixdialekter.

- **export kommandot gör om en lokal variabel till en global.**

Metatecken- borttagning av kommandotolkning

- Inom ’ ’ förlorar alla tecken sin speciella betydelse.
- Inom ” ” förlorar alla utom $, \ ,` och ` sin speciella betydelse.
- \<tkn> –tar bort den speciella betydelsen för nästkommande tecken, tkn.
- Shell matchar quotes från vänster till höger och i par.
- Prova)

$ echo ”’”,”’,

$ echo *

$ echo *

Några Exempel

```
$ cd
$ echo $LOGNAME b*
$ echo ’ $LOGNAME b* ’
$ echo ”$LOGNAME b* ’
$ echo \$LOGNAME b*
$ echo \’ \”\\
$ echo ’ ” ’ ” ’ ”
$ awk –F: ’{ print $1 ” ’\’ ’s shell:” $7 }’  /etc/passwd
Kalle’s shell : /bin/ksh
```

Interna optioner med set

- Interna optioner påverkar hur shell skall fungera
  + sätter en option till
  - stänger av en option
- För att se uppsättning av optioner:

  $ set –o
- För att se alla shell/miljö variabler:

  $ set
- För att definiera en option:

  $ set –o option

  $ set –o emacs

  $ set –o vi
- För att ta bort definitionen av en option:

  $ set +o option

  $ set +o macs

Kortkommandon till kommandon: Alias

- Alias tillhandahåller ett kort namn för en längre kommando sekvens.

  - För att definiera alias:

    $ alias namn=värde

  - Använd ’ ’ om metatecken förekommer

    $ alias ll =’ls –la’

    $ alias h=history

Alias

$ alias namn - visar värde av alias namn

$ unalias namn - tar bort alias definition

$ alias - visar alla alias

- BASH innehåller många fördefinierade alias:

$ alias <enter> # Listar alla alias

$ alias rm

alias rm=’rm -i’

$ alias ll

alias ll=’ls -l --color=tty’

Funktioner, mer komplexa kortkommandon

- Shell funktioner associerar ett namn med en lista på kommandon.
- Syntax

    function name { lista; }

Exempel)

```
function ant_anv  { who | wc –l; }
$ ant_anv
8
```

Funktioner och argument

- **Argument är indata till en funktion/kommando**

  ls –al /etc -al och /etc är argument $1 och $2

- **Argument till shell funktion/script kan nås via dom positionella parametrarna $1…$9**

- **Funktionen kan definieras på flera rader:**

```
$ function agare {
> ls –ld $1 | awk '{ print $3 }'
> }
$ agare /usr/bin
root
```

Visa funktioner: set

- *set* kommandot visar definierade funktioner.

```
$ set
agare ( ) {
ls –ld | awk ’ { print $3 } ’
}
$ _
```

Uppstart scripts bygger användarmiljön

- **Variabler, optioner, alias samt funktioner som är definierade interaktivt i shell:**
  - **förloras om du avslutar shell.**
  - **är inte tillgängliga från andra shell.**
  - **Förloras om man startar vissa program och applikationer.**

För att ändra på detta, definiera allt i uppstartsfilerna.

Uppstart scripts

•BASH läser två uppstartsfiler/scripts vid inloggning:

- /etc/profile - ett/system
- $HOME /.bash_profile - en/ användare

• Om ENV variabel är definierad i $HOME/.bash_profile:

ENV =$HOME/.bashrc ; export ENV

• Då läser /exekverar BASH även $HOME/.bashrc

Övriga subshell eller program/cmd läser enbart ENV fil.

Uppstart scripts

- Allt man vill definiera en gång vid inloggningen skall definieras i .bash_profile filen.
- PATH, ENV, MANPATH, TERM, …

- Allt som inte ärvs av subshell skall definieras i ENV filen.
- erase, kill, intr och andra tty driver tecken
- Shell optioner (set –o option)
- alias
- funktioner
- Shell variabler (ej globala)

Kapitel 7

Regulära uttryck

- Reguljära uttryck RE är korta och exakta definitioner av textmönster.
- De flesta UNIX program som letar efter textmönster använder regulära uttryck.

- grep - textmönster argument är reg. uttryck
- sed - sökning och byte
- vi - sökning och byte

- Andra program som använder reguljära uttryck:
  - egrep (extended grep)
  - awk - lex
  - csplit -ecpr

Reguljära uttryck

.	- Matchar ett enda tecken.
^	- Matchar början av raden.
[abc]	- Matchar a, b eller c.
*	- Matchar 0 eller mer föregående reg. uttryck.
$	- Matchar slutet av raden.
\	- Tar bort specialbetydelsen av nästkommande tecken.

grep och reguljära uttryck

- grep´s namn kommer från kommando *ed*

  g/reg – exp/p

- grep söker igenom input rader som innehåller reg-uttryck och skriver ut de rader som matchar reg-uttryck.

- Exempel:
  - $ grep t.e brev
  - $ grep \. brev
  - $ grep ’ \.’ brev

grep och reguljära uttryck

- vilka rader kommer att matchas i följande fall:

  - $ grep ’^B’ brev
  - $ grep ’t$’ brev
  - $ grep ’^$’ brev
  - $ grep ’[A-Z]’ brev

grep och reguljära uttryck

- $ grep ’^[A-Z]’ brev
- $ grep ’[aeiou][aeiou]’ brev
- $ grep ’[aeiou][aeiou]*’ brev
- $ grep ’a.*e.*i.*o.*u.*’ brev
- $ grep ’^ *[A-Z]’ brev

Flera reguljära uttryck

[^abc] – ej a, b eller c

\ <…\> - isolerade ord (vi, BSD grep)

+ - 1 eller mer föregående reguljära uttryck

? - 0 eller 1 föregående reguljära uttryck

| - eller (or)

egrep, sed och uttryck

• egrep (extended grep) kan söka efter flera textmönster och är ofta snabbare än grep.

```
$ egrep ’[lL] iten | [sS]tor’  brev

$ sed  ’s/:.*//’ /etc/passwd

$ sed  ’s/.*://’ /etc/passwd

$ grep ’^kalle:’ /etc/passwd | sed /.*://’
```

vi och reguljära uttryck

- vi´s kommandon / och ? söker efter reguljära uttryck.

- vi har substitute kommando med samma syntax som sed.

:s/liten/stor/

- Man kan definiera vilka rader som skall ändras och även använda g (global) kommando.

:1,$s/liten/stor
:1,$s/liten/stor/g

Kapitel 8

Shell programmering

- Unix shell är också ett interpreterande programmerings språk.

- Varje Unix shell användare är programmerare.

- Ofta använda kommandon kan lagras i filer – shell scripts.

- Kan exekveras som vanliga program.

Enkla script exempel

```
$ ls – l | grep brev

$ cat > ch-file
ls – l | grep brev
^D
$ chmod 755 ch-file

$ ./ch-file
```

Enkla script exempel

```
$ cat > ch-file
ls – l | grep $1
^D
$ ch-file brev
$ ch-file nyttbrev brev namn
$ cat > ch-file
for fil in $*
do
ls – l | grep $fie
done
^D
$ ch-file nyttbrev brev namn
```

Användar definierade variabler

- **BASH variabler kan ha textsträngar som värde:**

$ m=~kalle/brev

$ echo $m

/home/kalle/brev

- **Vad händer om vi vill använda variabeln m för att göra en kopia av filen brev?**

$ cp $m $m_bak

Användar definerade variabler

- för att slå ihop variabelns värde med en textsträng:

  använd { } måsvingar

```
$ echo $m_bak   # m-bak existerar inte

$ echo  ${m}_bak
 /home/kalle/brev_bak

$ cp $m ${m}_bak   # fungerar
```

Speciella variabler

- $1, $2,……$9 - argument 1- 9
- $* - alla argument från 1
- $# - antal argument från 1
- $$ - shell process ID
- $? - sista cmd exit status

  Kommandots felkod:

  0 = allt ok

  >0 nåt e fel!

Exit koder

- alla processer, kommandon och program returnerar en exit kod som visar om processen lyckades eller misslyckades.

exit kod 0	- process lyckades
exit kod ej 0	- process misslyckades

for loop

- **syntax:**

```
for var in 1 2 3 4 nizze kalle pang
do
kommandon
Kommandon
…..
…..
done
```

- **Listan genomlöps och upprepningarna slutar när listan är slut. Miljövariablen var tilldelas det nästföljande element i listan vid varje upprepning.**

case sats

•syntax

case ord in

mönster) kommandon ;;

mönster) kommandon;;

esac

•ord är vanligtvis referens till variabelinnhållet.

case ${name} in

- mönster använder vanligtvis metatecken för filnamn och före eller efter |.

case sats

en case sats kan ingå i for loopen

Vi kikar i term_type

```
$ cat term_type
for i in $*
do
case ${i} in
tty [o-3])            echo $ {i} HDS te???;;
tty [45]|ttya)  echo ${i} DEC te??;;
*)              echo Okänd;;
esac
Done
```

Vi provkör:

```
$ term_type ttya tty3 tty07
  ttya: DEC term
  tty3: HDS term
  tty07: Okänd
$
```

test kommando

- test kommando undersöker ett uttryck och returnerar 0 om uttrycket är sant och returnerar något annat om uttrycket är falskt.
- Syntax

  test uttryck

  eller

  [uttryck]
- med test kommando kan man testa filer

- f file # sant om file exist och är vanlig fil
- d file # sant om file exist och är dír fil
- x file # sant om file exist och är exekverbar.
- s file # sant om file exist och är >0

test kommando

- man kan testa textsträngar

  S1 = S2 # sant om S1 identisk med S2

  S1 != S2 # sant om S1 inte är identisk S2

- man kan testa numeriska värden

n1 –eg n2 # =

n1 -ne n2 # !=

n1 –gt n2 # >

n1 -lt n2 # <

n1 –le n2 # <=

n1 –gl n2 # >=

If /else satsen

- **syntax :**

  if [uttryck];

  then kommando

  [elif [uttryck]; then kommando]…

  [else kommando]

  fi

- **exit kod från uttryck bestämmer om:**

  0 - uttrycket sant (omvänd logik)

  ej 0 – uttrycket falskt

While loop

- **Upprepa så länge sannt!**
- **Syntax :**

```
while [ uttryck ]
do
      kommandon
done
```

Exempel;

```
while true  #  evighetslop
do
  who
  sleep 30
done
```

While loop

- **Om while loopen testar en kommando**
  lista så är det sista kommandot exit
  kod som kontrollerar loopen. I detta fall sleep 30

```
while who | grep kalle >/dev/null
do
   sleep 30
done
echo ”Kalle har loggat ut !”
```

Exit satsen

- **exit n avslutar shell script med exit koden,**
  0 är förbestämd
  if error
  then
  echo ”error inträffade”
  exit 1
  fi
  ….
  ….
  exit

read satsen

- **Inbyggd (i shell) read satsen läser standard input (tangentbord) rad och tilldelar den till variabler.**

```
$ read var
Detta är en test
$ echo $var
Detta är en test
$ read var var1
Detta är en test
$ echo $var
Detta
$ echo $var1
är en test
```

script exempel

```
$ cat bovar
   awk – F: ’{ print $1, $6 }’ /etc passwd |
   while read user home
   do
   if echo $home | grep ungk >/dev/null
   then
   echo `du –s $home | awk ’{ print S1 }’ ` $user
   fi
   done | sort  –nr | head | awk ’{ print $2 ”:”, $1 }’
$
```

- Komplexa scripts kräver ofta avlusning $sh -x bovar

Vilken interpretator?

- Om man inte definierar vilket shell som skall tolka innehållet så blir det Bourne shell.
- andra shell kan användas som interpretatorer

```
#!fullständig_path_till_shell

#!/bin/sh

#!/bin/csh

#!/local/bin/bash
```

Kapitel 9

Användar systemadministration

- systemadministration kräver speciella privilegier.
- super-user (ID=0 i /etc/passwd) kan öppna/modifera

  vilken fil som helst.
- För att bli super-user (root) logga in som root eller använd su (switch user) kommando.

  $ su

  Password:

  $ su –

  Password:

Super-user ROOT

- Som super-user kan du utföra operationer som är otillgängliga för andra.
- öppna/modifiera filer.
- Ändra ägarskap, grupptillhörighet, access rättigheter för filer.
- Stoppa/döda vilka som helst processer.
- Skapa nya filsystem..

OBS!

- Var försiktig när du arbetar som root.!!!

  # rm –r /tmp tar bort bara /tmp katalogen

  # rm –r / tmp tar bort hela systemet / !!!

Boot procedur / ta upp systemet

- moderna system laddar UNIX kernel in i minnet och

exekverar den när man trycker på ON/OFF knappen.

- init process (PID=1) startas efter laddningen av kernel filen.

-ansvarar för system konfigurering.

-startar andra processer.

/etc/inittab är konfigurationsfilen i Sys-V och Linuxdialekter.

/etc/rc är konfigurationsfilen i BSD-system.

Boot procedur / ta upp systemet

En rad saker händer när systemet startar, kortfattat:

1. **Power good ”nätdelen”**
2. **Pre post test ”bios”**
3. **Boot device ”bios”**
4. **Boot sektor med boot-strap programmet ”/boot/boot.b”**
5. **Sched, planläggaren /boot/initrd och kernel**
6. **init programmet tar systemet till default runlevel**
7. **Aktuellt run-level script /etc/rc #runlevel startas**
8. **Slutligen är systemet helt laddat..**

Daemon processer

- Bakgrundsprocesser tillhandahåller hjälp för /eller tillgång till tjänser och kallas för daemon processer.
- UNIX system använder en mängd standard daemon processer för att tillhandahålla system tjänser – mail, printing osv.
- Deras startscript återfinns i /etc/rc.d/init.d

- sendmail - mail deamon
- lpd - printer deamon BSD
- lpsched - printer deamon Sys V
- inetd - internet ”serves” deamon
- rshd - BSD remote shell deamon

Starta och stoppa bakgrundsprogram och demoner

Det finns flera sätt att starta och stoppa bakgrundsprogram.

- /etc/init.d/sendmail status

  sendmail (pid 1189) running..

Får vi se mailserverns status, om den kör, vilka process ID den har erhållit.

- /etc/init.d/sendmail stop
- /etc/init.d/sendmail start

Flera instruktioner kan finnas såsom restart, reload med mera.

Detta gäller de flesta bakgrundsprogram

Starta och stoppa bakgrundsprogram och demoner

Andra bakgrundsprogram kan sakna start-script och startas då genom att man skriver kommandot, prompten kommer tillbaka efter eventuella felkoder, det kan också hända att de bara kommer i loggböckerna som finns i katalogen /var/log

- gnugkd <retur>
- gated <retur>
- Viktigaste loggboken /var/log/messages

Att avsluta dessa bakgrundsprogram görs oftast med kill kommandot.

Leta upp bakgrundsprogrammet: ps –e | grep gated

Döda kill –9 <PID>

Cron Daemon

cron är ett kronograf ur, en daemon, den exekverar kommandon i crontab filen vid vissa tidpunkter.

- /var/spool/cron/ #RedHat linux
- Demonen heter ”crond”
- format för crontab filen:

  minut timme dag månad veckodag cmd

Varje tidsfält innehåller en siffra, eller en lista av siffror med kommatecken, eller en intervall av siffror med bindestreck, eller * (varje).

$ crontab –l # Visar din cronlista

$ crontab –e # Öppnar din cronlista för redigering

$ crontab –r # Raderar hela din cronlista

Cron Daemon

Exempel)

0,10, 20, 30 * 10-20 * * date >/dev/console

- **denna rad kommer exekveras klockan 1:05**

  varje söndag och onsdag.

  5 1 ** 0,3 min prog

- **resultatet av körningen av programmet kommer cron att skicka till mig som ett e-mail.**
- **/etc/cron.hourly/**
- **/etc/cron.daily/**
- **/etc/cron.weekly**
- **/etc/cron.monthly/**

Ta ner UNIX system

- shutdown kommandot är det enklaste sättet att ta ner ett UNIX system.

  - syntax och optioner är systemspecifika.
- Shutdown

  - skriver meddelande till aktiva terminaler om inkommande nedtagning.

  - tillåter inte nya inloggningar ca 5 min före procedurens start.

  - dödar (tar ner) aktiva processer.

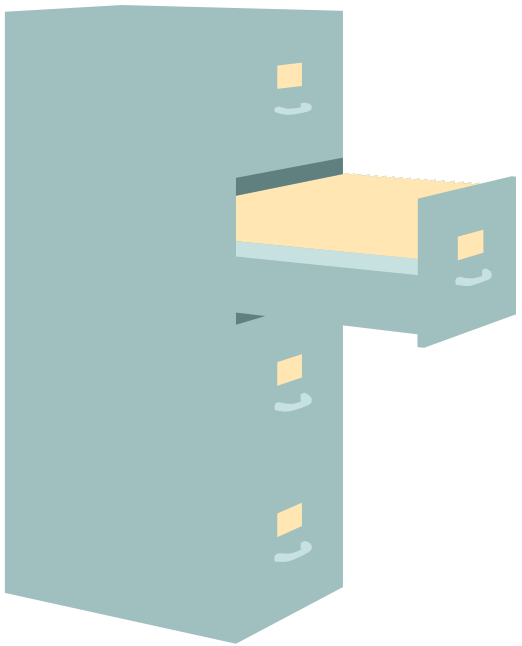
  - stoppar andra processer.
- halt är också komfortabelt om man vill ta ner systemet
- reboot är ett alternativ om man vill starta om systemet

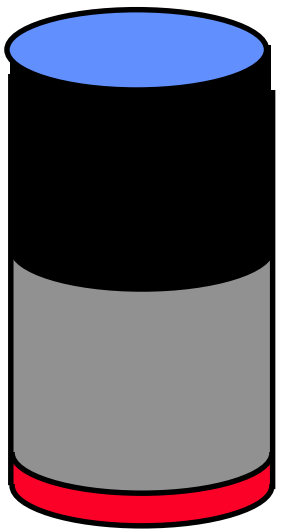
Administrativa verktyg

- De flesta UNIX System har ett grafiskt eller ett menybaserat administrativt verktyg.
- webmin http://localhost:10000

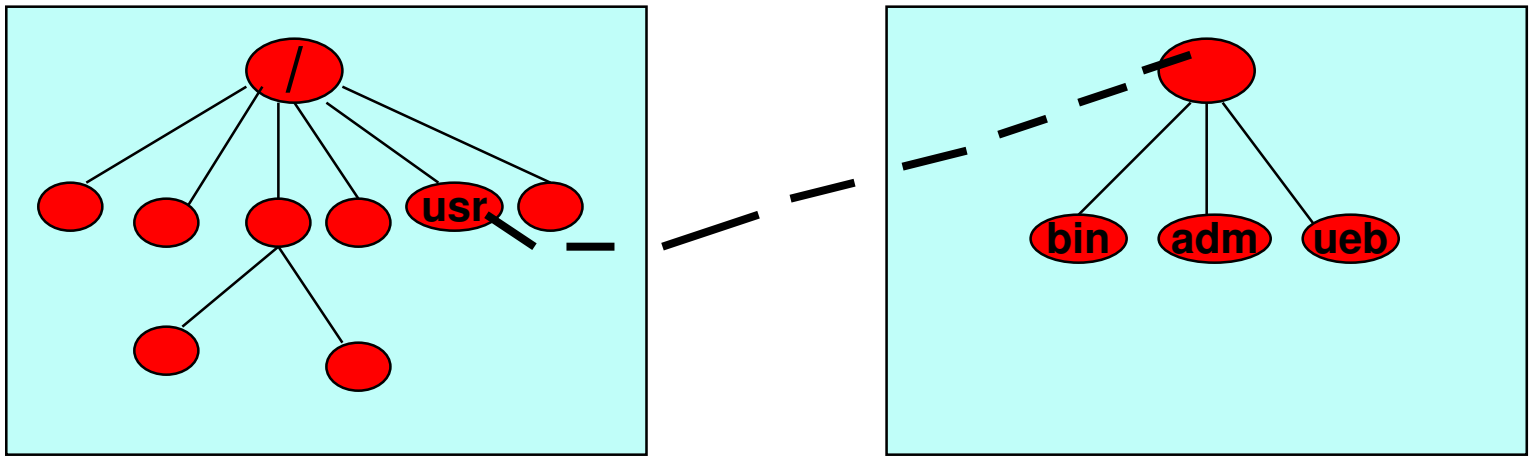
  (starta först webmin /etc/rc.d/init.d/webmin start)
- setup från kommandoraden
- Med dessa verktyg kan man:
  - Hantera nya och existerande användare.
  - Hantera nya och existerande grupper.
  - Installera och ta bort program.
  - Aktivera och stänga av seriella portar.
  - Hantera skrivare.
  - Starta/stoppa definerade nätverkskort.
  - Utföra system säkerhetskopiering.
  - Samt mycket mera.
- Alla verktyg stöder alla dessa, men kan ha andra liknande funktioner.

Filsystem och Partitioner



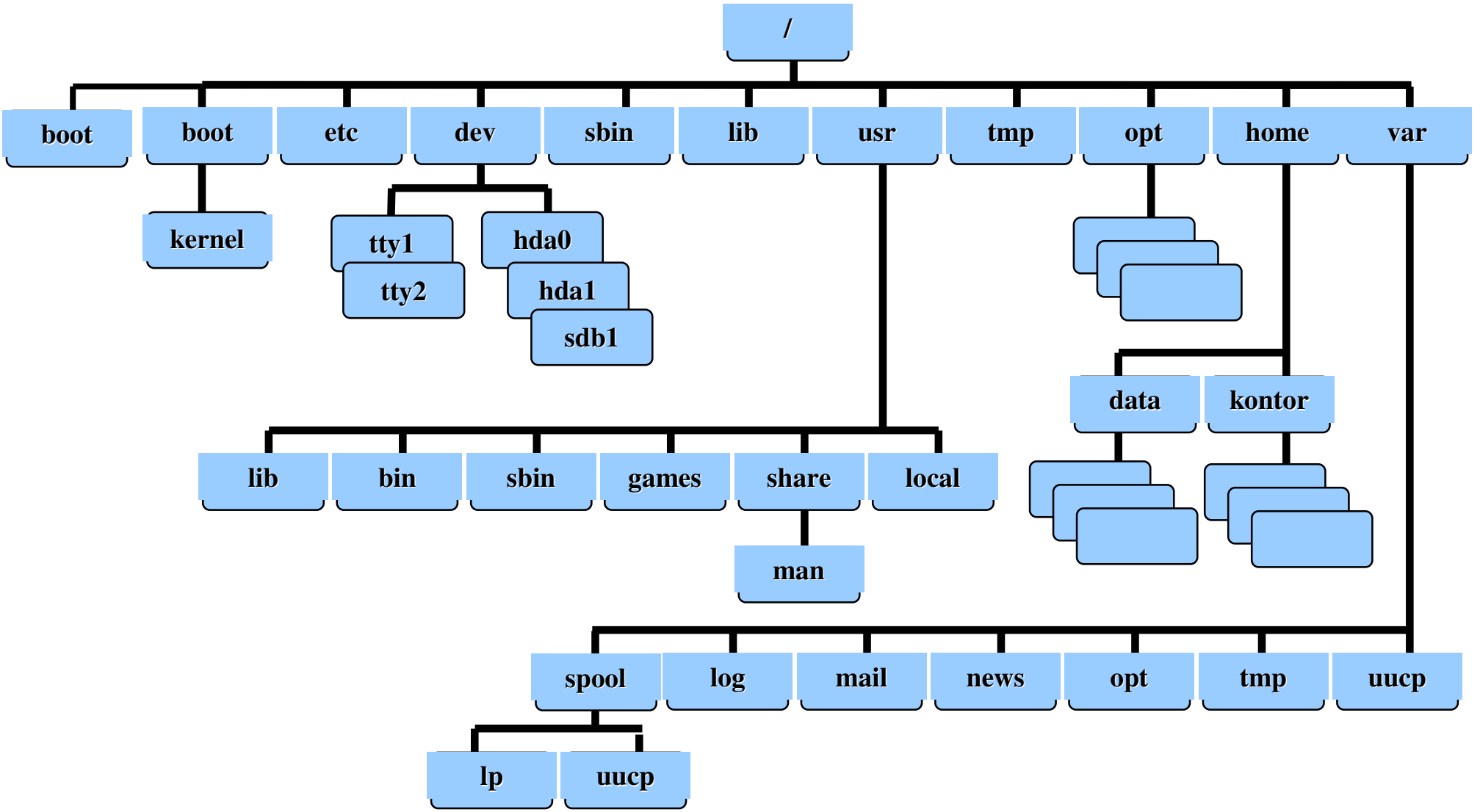


Filsystem och Partitioner



- mount kommandot används för att montera ett filsystem.
- du kommandot används för att se antalet block(512/1025 bytes) som behövs för att lagra olika filer, även biblioteksfiler.

Filsystem och Partitioner



Backup hantering
tar

- **Fördelar:**
  - **Enkel att använda.**
- **Nackdelar:**
  - **Kan ej backa upp specialfiler.**
  - **Kan ej backa upp tomma filer eller bibliotek.**

- **Syntax:**
  tar [flaggor] filer …

Exempel;

tar cvf /tmp/home.tar /home

tar xvf /tmp/home.tar

tar tvf /tmp/home.tar

Backupexempel
cpio

- **find . -print | cpio -ov > /dev/fd0**
  find och cpio skriver ned alla filer från det aktuella dir. till backupmediat /dev/fd0

- **cpio -itv < /dev/fd0**
  cpio listar filerna på bakupmediat /dev/fd0

- **cpio -iv kundfil < /dev/fd0**
  cpio skriver tillbaka filen kundfil.

Komprimera

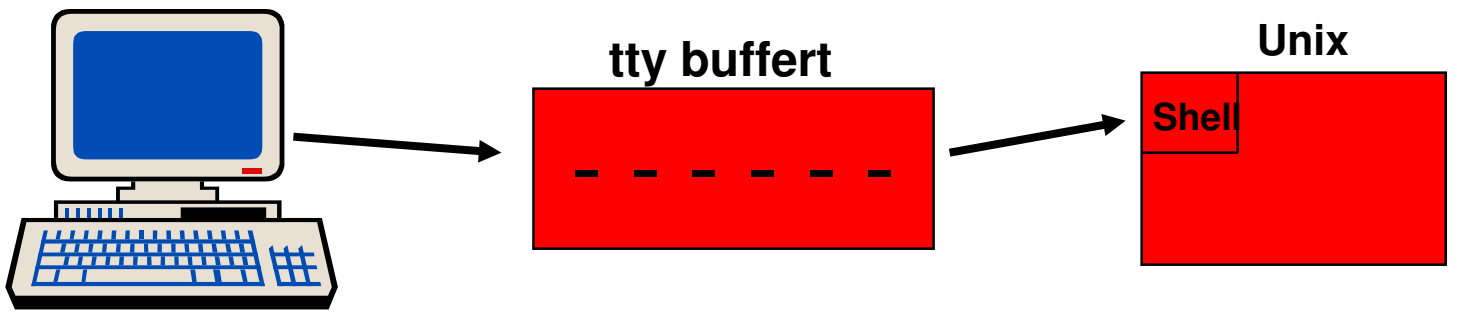
- **Kommandot compress komprimerar filer.**
- **uncompress återställer filerna.**
- **zcat kan läsa komprimerade filer.**

Modernare komprimeringsverktyg som är 50% mer effektiva än compress

- **Gzip komprimera**
- **gunzip packa upp**
- **bzip(2) bnu zip**
- **bunzip(2)**

tty driver repetition

- Stty kommandot visar eller konfigurerar tty driver parametrar.



- stty –a # visar alla parametrar
- stty erase ’^h’ # ändrar

Linux Virtuella konsoler

Alla linux har som standard 7 virtuella konsoler, de är en form av textterminaler

-> Konsol 7 är den grafiska x-terminalen

-> Ctrl+Alt+F<nummer> hoppar till konsol.

-> I filen /etc/inittab kan du ändra terminalerna

Exempel på en sådan konsol/terminal rad:

1:2345:respawn:/sbin/mingetty tty1

Tag helt enkelt bort de terminaler du inte vill ha, den grafiska terminalen får nästa <nummer> ovanför den högsta.

Användaradministration

Add lägga till och ta bort användare i Redhat linux är enkelt.

- useradd kalle
- userdel kalle
- usermod kalle

Den som vill kan naturligtvis editera /etc/passwd manuellt och skapa katalogerna

- webmin är ytterst bekvämt

Användaradministration

Alla användare bor i som standard /home katalogen. Det behöver inte vara så utan man kan ha /home1 /home2 osv.

Root användaren bor i katalogen /root

Många administratörer organiserar /home med olika underkataloger såsom wheel, users, members osv.

useradd –m –d /home/wheel/kalle –G wheel kalle

Skapar kalles hemkatalog (-m) och användaren kalle som blir medlem av gruppen wheel.

Alla lokala Unix användare finns i /etc/passwd

- **Innehåller en rad för varje användare som har konto på systemet.**

  kalle:x:500:500:Kalle Svensson:/home/kalle:/bin/bash

  Använder : som fältseparator.

Fälten i turordning:

1. **”kalle” Användar namn**
2. **”x” Krypterat password eller x om utökad säkerhet.**
3. **”500” User ID nummer (UID)**
4. **”500” Grupp ID nummer (GID)**
5. **”Kalle Svensson” Text fält, kommaseparerad.**
6. **”/home/kalle” Hemma bibliotek.**
7. **”/bin/bash” shell**

/etc/passwd är läsbar för alla!

/etc/shadow -file

- Unix efter 1995 implementerar /etc/shadow filen som innehåller krypterat password (lösenord).
- Förbättrad säkerhet – filen är endast läsbar av root (superuser).
- /etc/passwd har x i password kolumnen som referens till /etc/shadow filen.
- Gör det svårare att gissa password och genomföra krypterade strängar.

NIS / LDAP- alternativ till /etc/passwd

- I stora nätverk av Unix datorer är underhåll av identiska /etc/passwd väldigt krävande.
- I sådana miljöer implementerar man NIS/NIS+/LDAP
  - Lokala maskiner /etc/passwd behåller endast ett litet antal system och ”privata” konton.
  - NIS/LDAP servers /etc/passwd (samt andra dbfiler) innehåller konton för LAN.

KAP 10

- **Användarkommunikation**
  - **Mail**
  - **Write**
  - **Talk**
  - **Internet åtkomst**

Att läsa brev – mail

- Mail -programmet kan användas för att läsa och skriva brev.
- Utan argument undersöker mail inkommna brev.
- Om brev/breven har kommit visar mail följande:
  - $ mail
  - ”/var/mail/kalle” , 3 messages 3 new
  - > N 1 pelle Wen Okt 13:55 11/226 test
  - N 2 maria Wen Okt 20:05 125/937 raport
  - N 3 bettina Wed Okt 21:45 45/1890 Semester
  - ?

Att läsa brev – mail

- Vid prompten (?) tryck Enter för att läsa aktuellt brev
- Eller skriv kommando och sedan Enter
  - Följande kommandon finns

    . Visa aktuellt brev

    + visa nästa brev

    - visa förgående brev

    3 visa brev 3

    d ta bort aktuellt brev

    h visa lista av brev

    s file spara brev i filen file

    q avsluta, spara inte lästa brev

    x avsluta, spara allt: lästa, olästa, och borttagna brev

    ? visar hjälp

Skicka brev - mail

- För att skicka brev till en eller flera användare på samma system (maskin) skriv deras namn som argument:

```
$ mail pelle kalle
Subject: test
Detta är en test av mailsystemet
.
$
```

Skicka brev - mail

- För att skicka ett mail till användare inloggade i LAN använd namn@maskin som argument
- $ mail kalle@pilsner pelle@dab
  Subject: raport
  .
  $
- Om du har Internet tillgång använd:
  användarnamn@iternet maskin adress
- $ mail mrx@ing-steen.se
  Subject: hej på dig
  .
  $

Tilde kommandon i mail

- **Man kan använda tilde(~) kommandot för att ändra eller visa ett nytt meddelande.**
  - **Tilde kommandot måste skrivas i början på en ny rad.**

  ~p visar meddelande samt listan av användare som skall få meddelandet samt subject.

  ~v startar vi editor och läser in meddelandet.

  ~h modifierar header informationen, Subject och lista av mottagare av meddelandet.

  ~? listar tillgängliga tilde kommandon.

Använda mail program

- Många andra kommersiella och public-domain mail program är tillgängliga:
- elm electonic mail, klassiker
- pine klassiker inom linux
- xmail
- EMACS rmail inbyggd mail program för EMACS
- CC:Mail från Lotus Development
- CDE Mailer i Comon Desktop Enviroment mailtool grafisk verktyg på Sun maskiner

Mail program och shell script

- Subjekt av ett mail meddelande kan definieras med –s optionen och själva meddelandet kan skickas via pipe eller omdirigering av standard input.

  $mail –s ’report from meeting’ kalle < report
  $(date ; who) | mail –s `inloggade användare` kalle pelle
- Kommando ersättning (substitution) kan användas för att skapa en lista av mottagare:
  $mail –s ’hej på er’ `who|awk ’{ print $1 }’ ` < meddelande
  $mail –s ’hej’ `awk –F: ’{ print $1 }’ ` /etc/passwd < till_alla

Skicka meddelande till en annan inloggad användare
write

- **Syntax:**
  write user [tty]
  - **tty - används för att specificera porten om användaren är inloggad på flera ställen.**

Stänga av / på meddelande funktionen med mesg

- Visar om man tar emot meddelande eller ej, samt slår på eller stänger av funktionen.
- Syntax:
  mesg [-n |-y]

Skicka meddelanden till alla inloggade användare.
wall

- **Syntax:**
  wall [file]
  - **file - anger att meddelande skall läsas från filen file istället för från standard input.**

Prata med en användare inloggad i LAN

- Talk kommandot kan användas för att ”ringa” upp en annan användare:
  $talk kalle@pilsner
- Den användaren måste då ”svara” genom att skriva kommandot talk samt adressen:

  $talk pelle@dab
  CTRL-D avslutar samtalet.
- Efteråt delas fönstret i två delar, den ena tillhör den första användaren och det andra den andra användaren.

Internet Kommandon

- **Internet service inkluderar:**
  - **remote login rlogin**
  - **remote fil kopiering rcp**
  - **Mail och news service**
  - **Allmän databas service**
- **Grafisk WWW browsers är mycket populära, *ex:***
  - **Netscape navigator**
  - **Explorer**
  - **Opera**
  - **Mozilla**

Internet Kommandon

- Många rad orienterade kommandon finns och används fortfarande:

telnet - remote login program.
ftp - file transfare protocol program.
gopher - söker för Internet information.

rn, trn, nn, tin, etc – news programs
finger, whois – talar om vem användaren är.

Anslut till nätverket

För att din dator skall fungera med nätverket måste den ha:

- Nätverkskort
- hostnamn
- IP adress
- Subnetmask
- Defaultrouter
- Nameserver

Man kan ställa in nätverkskonfigurationen via wizards med:

- setup –från konsolen
- webmin -med netscape
- netconfig -från kommandoraden / netcfg x-win

Anslut till nätverket

Kolla om nätverkskorten är igång och anslutna till media:

$ ifconfig –a # visar alla nätverkskort

Leta efter UP och RUNNING samt inet addr

Saknas ip adresser och eller korten är nere prova med att starta om aktuellt kort:

$ ifconfig eth0 down ; ifoconfig eth0 up

$ ifconfig eth0

Kolla igen!

Anslut till nätverket

Har nätverkskortet erhållit någon IP adress ?

- **Kör ni DHCP, är alla kablar anslutna**
- **Körs ”dhcpcd” på din klientdator**
- **Kontrollera /etc/resolv.conf**
- **Kontrollera /etc/nsswitch.conf**
- **Kontrollera /etc/host.conf**

Nätverksinställning i ordning ?

- **Kolla om dhcp client demon körs på din dator om ni använder dhcp:**

  /etc/init.d/dhcpcd status

- **Kolla om nätverket är aktiverat:**

  ifconfig –a

- **Kolla om mediakontakt finns:**

  arp –a

- **Kolla om default gateway finns:**

  netstat -rn

Namnserver filen

- **Kolla i filen /etc/resolv.conf, Namnservrar ?**

  nameserver 127.0.0.1

  nameserver 80.84.32.10

  nameserver 80.84.32.12

  search server.se

Namn-bindningsordning

- Filen /etc/nsswitch.conf styr i vilken ordning man letar efter hostar/domäner med mera:

  #hosts: db files nisplus nis dns

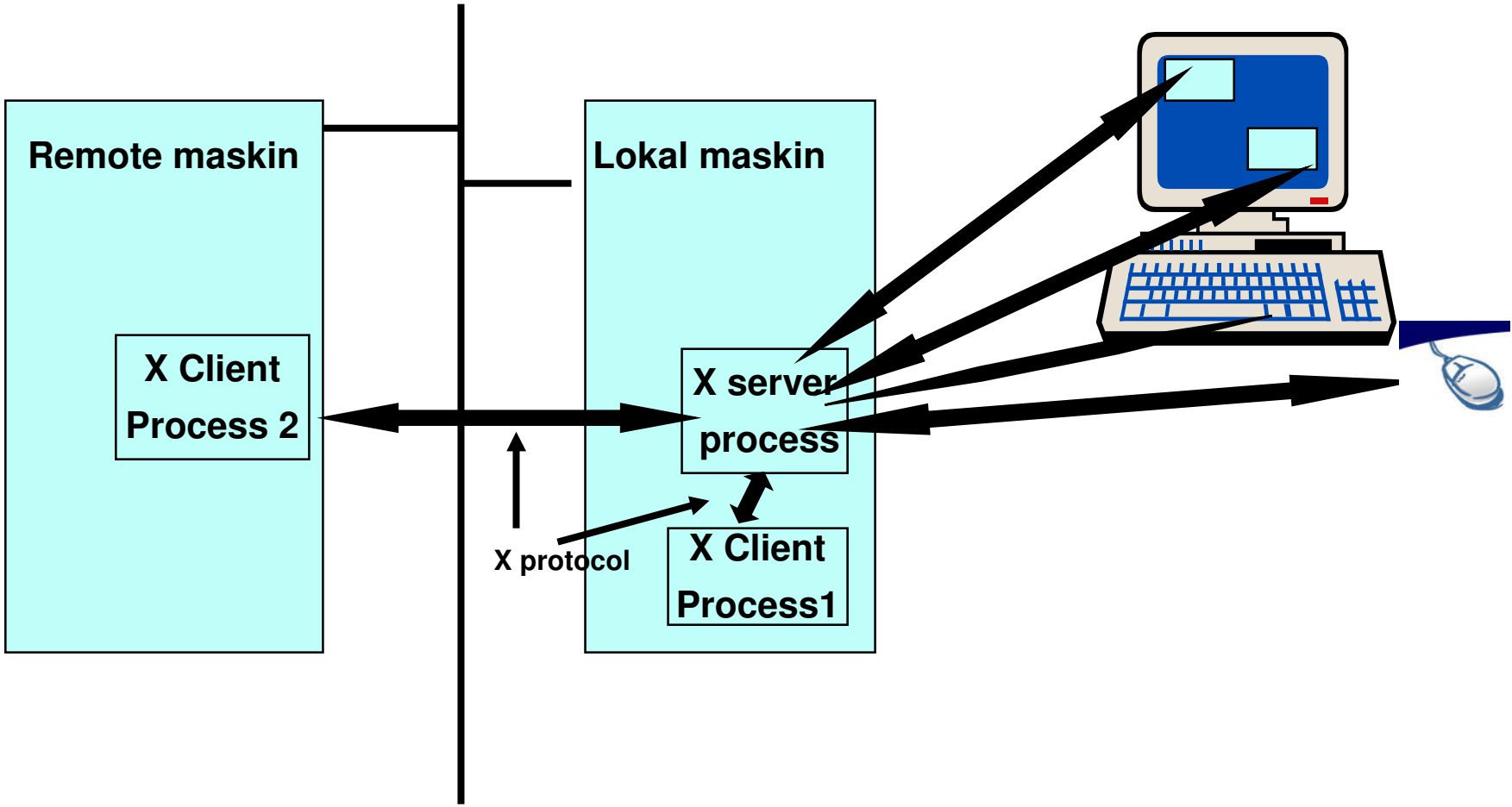
  hosts: files dns nisplus
- Sist filen /etc/host.conf som bestämmer i vilken ordning man letar upp adresser till hostar.

  order hosts, bind

KAP 11

- **X Windows System**
  - **Koncept**
  - **Kommando optioner**
  - **DISPLAY variabler**
  - **X resurser**

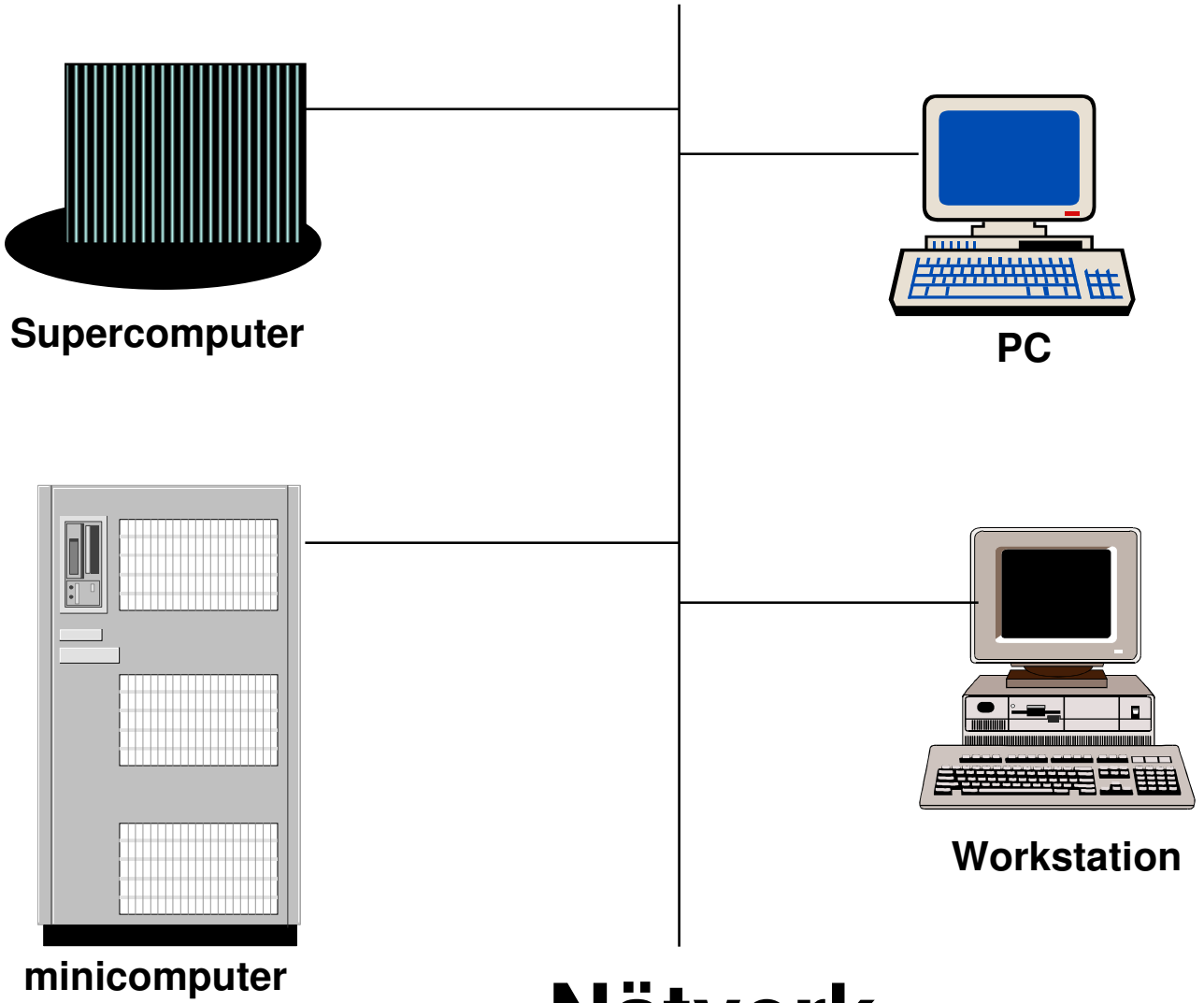
Client/Server Arkitektur



Client/Server Arkitektur

- Xserver process underhåller (tar hand om) en X display – skärm,tangentbord samt mus.
  - Tar emot indata från tangentbord och mus.
  - Kontrollerar utdata till skärmen.
- X client processer kommunicerar med X server process med hjälp av X Protocol.
  - Client processerna kan köras på lokala eller andra maskiner i nätet.
  - Användarna märker ingen skillnad (network transparency).

Hårdvaru oberoende



Hårdvaru oberoende

- X servers kommunicerar med X clienter via X protocol som är **maskin oberoende**
- Xclient program som exekveras på en maskin kan därför utnyttja skärmen på andra maskiner.
- Till och med OS kan bli olika (OS oberoende)
  - Det finns X servers för MS-Windows VMS OS/2 osv.

X Windows mål

- X tillhandahåller fönsterhantering, text och grafikhantering, men ställer inga krav på utseende eller hur det känns att jobba med X Windows – ”look and feel”
- Därför blev X allmänt accepterat.
- De två mest kända ”look and feel” hanterare är:
  - Motif utvecklad av Open Software Foundation
  - CDE är baserad på Motif
  - Open Look uvecklad av AT&T och Sun

Att starta X

- På många system startas X redan när man bootar systemet.
  - Sedan loggar man in i X session
  - /etc/inittab (initdefault runlevel 5=xwindows)
- På andra system måste man starta X efter inloggningen.
  - Startkommandot kan variera
    $ startx
    $ startkde
    $ xinit

Terminal emulator

- xterm är en av många terminal emulatorer för X.

- Andra som har andra copy-and-paste, fönsterhiss och andra faciliteter är:
  - zterm - från MIT
  - cmdtool - från Sun
  - decterm - från DEC
  - aixterm - från IBM

Window Managers

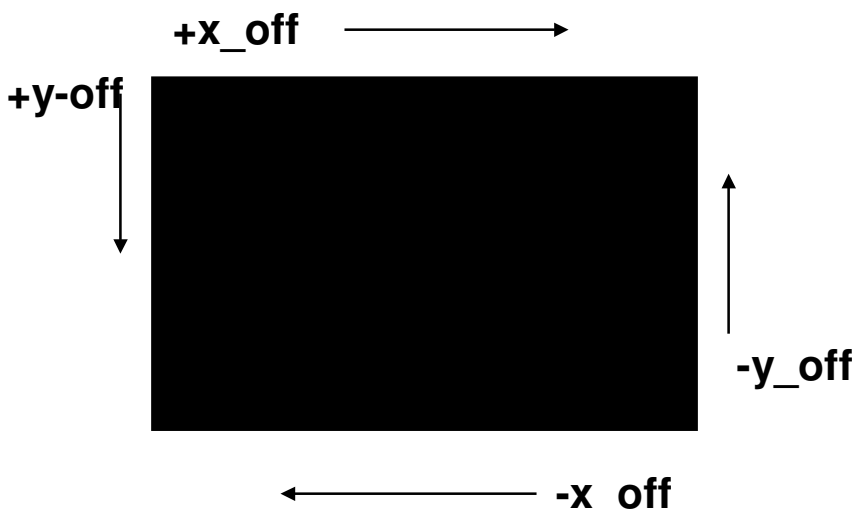
- Window Managers är en X client program som tillåter användaren att:
  - Flytta och ändra storlek för fönster
  - Ändra fönster till icon och tvärtom
  - Utföra program
  - Etc.
- Window manager är ”först bland clienterna”
  - Dekorerar andra clienters fönster
  - Talat om hur de skall ”uppföra sig”
  - Etc.

Window managers

- **Bland de mest kända finns:**
  - **Dtwm : Desktom Window Manager (CDE)**
  - **Gnome: GNU**
  - **KDE :GNU**
  - **Enlightment: GNU**
  - **Mwm : Motif Window Manager (OSF)**
  - **Olwm : Open Look Window Manager (Sun & AT&T)**
  - **Twm : TabWindow Manager (MIT)**
  - **Fvmv Motifliknande WM (Linux)**
  - **Många andra.**

X positions optioner

- De flesta X applikationer har positions optioner
  - -geometry width x hight [+-] x_off [+-] y_off
  - Width och higth är vanligtvis i pixels men ibland också i rader eller tecken.
  - x_off och y_off är in pixels och definerar avståndet från skärmens kanter.



X positions optioner

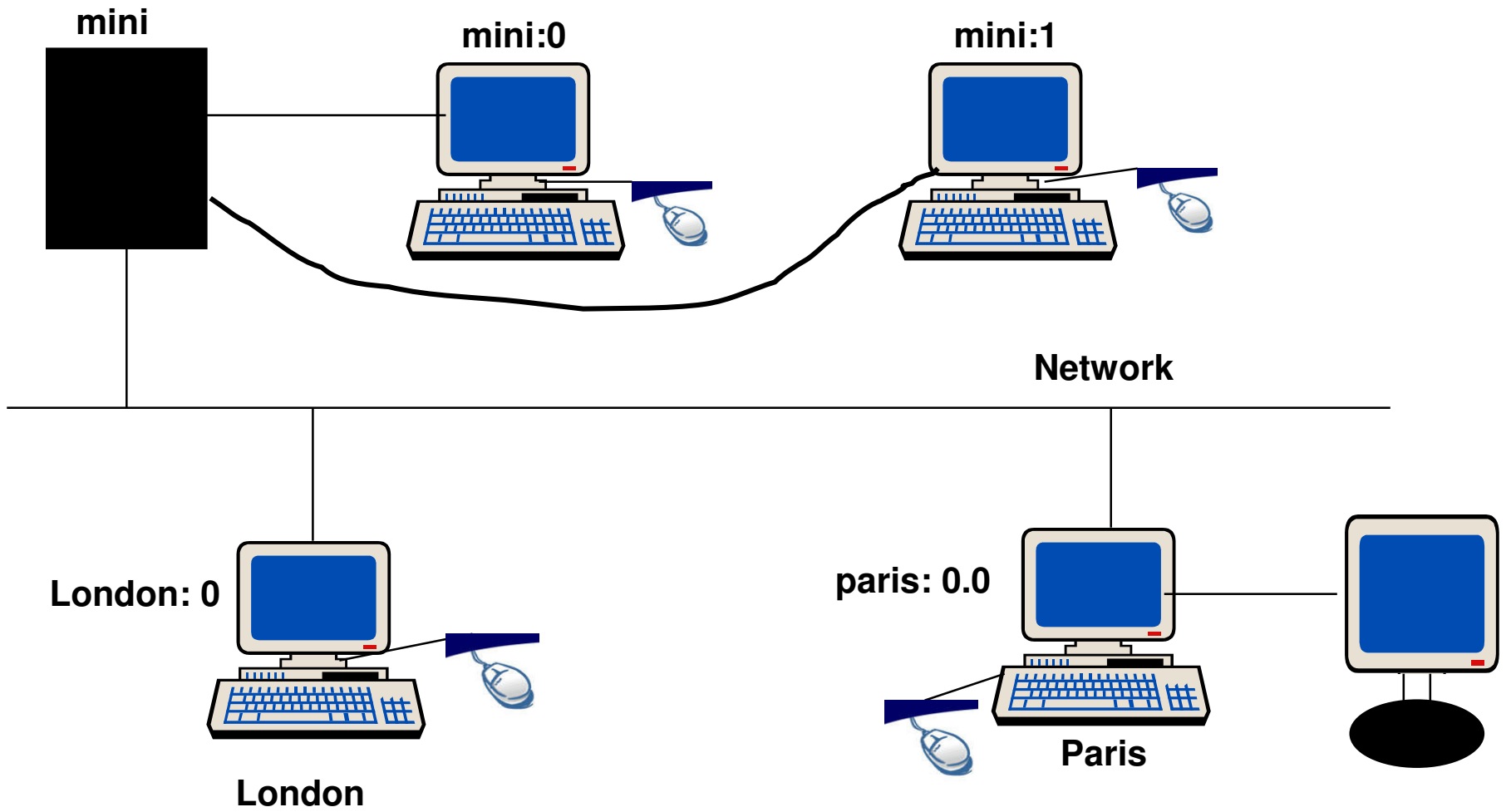
$ xterm –geometry 80x40-0-0 &

- **Skapar en 80 tkn bred 40 rader hög *dtterm* i nedre högera hörnet av skärmen.**

$ xclock –geometry 200x200+20-20 &

- **Skapar en 200x200 pixel xclock lite från nedersta vänstra hörnet av skärmen.**

X Display option



X Display option

- Xdisplay består av tangentbord, mus, och en eller flera skärmar
- Man kan definiera vilken maskin, display och skärm som skall användas av en program med –display option x_app –display hostname:display.screen

$ xclock –display paris:0.1 &

X color option

- Förgrund (text och grafik) samt bakgrundsfärger kan definieras med
- –fg color och –bg color.

$ dtterm –fg red –bg yellow $

- Tillgängliga färger finns i filen rgb.txt, eller kan listas med:
- $ showrgb | more

Andra X optioner

- Några andra X standard optioner:
- font font_namn
- iconic startar pgm som en icon
- title sträng
- rv revers video

- Vissa applikationer kan ha egna optioner:

$ xclock –digital –fg blue –bg pink $

X Display variabel

- Global DISPLAY variabel används av alla X program.
  - Identifierar maskin, display, skärm samt vart programmet skall visas.
  - Man behöver då inte definiera –display.

$ env | grep DISPLAY

DISPLAY=0.0

dvs display 0, screen 0 på aktuell maskin.

X Display variabel

- Man kan omdefiniera DISPLAY på lokal maskin med:

$ DISPLAY=0.0 ; export DISPLAY

- efter inloggning via nätverket till en anna maskin med rlogin kommando:

$ DISPLAY=lokal_maskin:DN.SN;export DISPLAY

DN= displaynummer

SN= skärmnummer

X Resurser

- X program kan anpassas genom att man använder X resurser.
- Den enklaste definieringen av resurser är:

  object* recource_name: value

  class* recource_name: value
- En applikations objekt har namnet (xclock) och tillhör en klass (Xclock)
  - För klass förklaring undersök manualsidorna.
  - Applikationen kan ändra objekt namnet med: -name option, men inte klass.

X Resurser

- Object resurs definition är ”starkare” än class resurs definition, tex:

xclock * background : green

Xclock * background : red

$ xclock$ # namn clock ; Xclock class ; green background

$ xclock –name minklocka $ # namn minklócka; Xclock class; red background

$ xclock –name xclock $ # namn xclock ; Clock class ;green background

X Resurser

- Dom flesta X miljöer har resurser på X server sidan.
- xrdb program kan användas mot servrars resurs databas.

$ xrdb –query # visar aktuella resurser

$ xrdb –merge file # slår ihop resurser definierade i file med serverdatabasen.

- Resurs definitioner kan lagras i vilken fil som helst.

$ HOME/.Xresources är vanlig

- Äldre X miljö kan använda:

$ HOME/.Xdefaults

X Resources

- **CDE /Motif använder en mängd av resurser *bl.a:***

background
foreground
fontList
activBackground
activForeground

osv **.**

Kapitel 12
Installera Linux

Man kan installera Redhat från många olika typer av media:

- CD-Skivor
- Web
- NFS (med start på diskett eller boot från nätverkskort)
- SMBFS (med start på diskett eller boot från nätverkskort)
- FTP
- Disketter
- Hårddisk

Under denna grundkurs fokuserar vi helt på installation från CD-skivor.

Har du det som behövs för installationen

RedHat klarar sig med mycket blygsam hårdvara:

- 386sx16
- 4Mbyte RAM
- 40MByte Hårddisk
- VGA grafik

Men vill man göra mera än bara titta på systemet:

- Pentium III 733MHz
- 64MByte RAM (gärna runt 2Gigabyte för en server)
- 4Gigabyte Hårddisk
- SVGA grafik med accelerator