Datorer och datoranvändning
Föreläsning 1 — Unix/Linux

Mattias Nordahl

mattias.nordahl@cs.lth.se

Föreläsning 1 — Unix/Linux

Förberedelse inför laboration 1.

- Operativsystem, Unix historik
- Filer och kataloger
- Kommandon
- Filskydd
- Kommandotolk
- Processer

Operativsystem

Dator — kör program

Operativsystem — en samling program som gör det möjligt att köra ”vanliga” program

Operativsystemet hanterar:

- de program som körs (program körs ofta parallellt, operativsystemet ser till att programmen får minne och tid att exekvera)
- yttre enheter (tangentbord, mus, skärm, nätverk, . . .)
- lagring av data (filer, . . .)
- skydd, felhantering kommunikation med användaren

Linux, Windows, Mac OS X, Unix, . . .

Unix

Historik:

1960-talet: Multics — Multiplexed Information and Computing System — stort operativsystem för stora datorer. Skulle lösa alla problem.

1969: Ken Thompson (AT&T Bell Labs) började skriva Unix, litet operativsystem för små datorer, avsett för programmerare. Idéer från Multics.

1973: Unix skrevs om i C av Dennis Ritchie, började distribueras.

Senare: mycket utveckling, många varianter (Linux, Mac OS X, NetBSD, A/IX, HP/UX, Irix, Android, IOS...).

Varumärket Unix ägs idag av The Open Group som är ett leverantörs- och teknikoberoende konsortium. Certifierar produkter (Linux ej certifierat).

Kännetecken för Unix

Grundläggande:

- Kärna — liten, grundläggande funktioner
- Processer, tidsdelning
- Filskydd
- Fleranvändarsystem
- Kommandotolk

Unix är:

- Litet, standardiserat, flyttbart
- Inte för nybörjare, fast mycket enklare att lära sig nuförtiden när det finns grafiska användargränssnitt och skrivbordsmiljöer (Unity, Gnome, KDE, Xfce . . .)

Filer i Unix/Linux

I en dator måste man kunna lagra program (Word, Excel, Java-/Scalakompilator, spelprogram, ...) och data (brev, hemsidor, programmeringsuppgifter, ...).

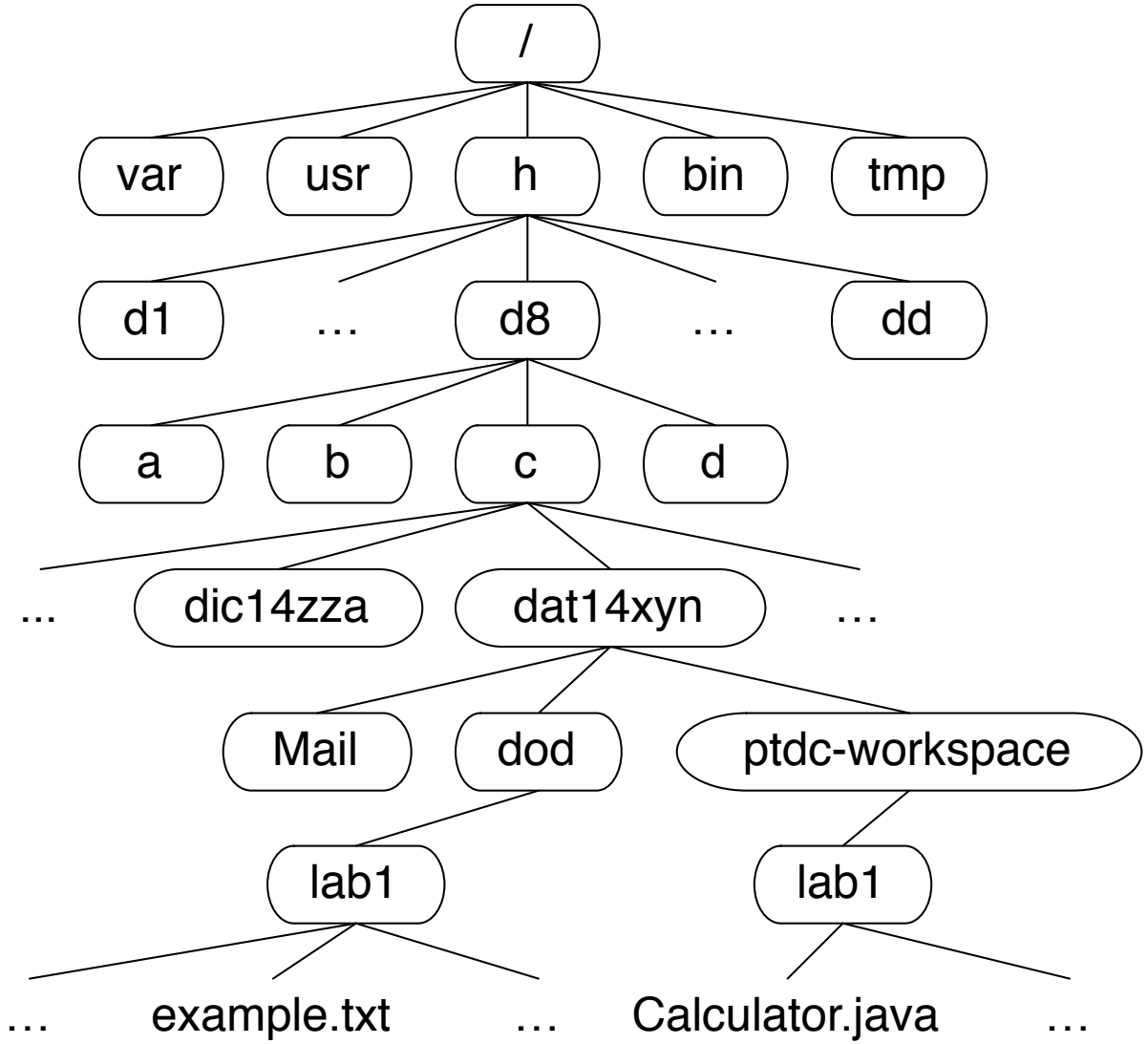
Man lagrar program och data i *filer*. En fil finns normalt på skivminne och har ett namn.

Filnamn i Unix/Linux:

- `namn.tillägg` (`Calculator.java`, `Calculator.class`, `brev.txt`, `test1`, ...)
- tillägget är bara en extra upplysning, bestämmer inte filtypen

Kataloger

Varje fil finns i en katalog. Kataloger kan innehålla underkataloger, så man får en hierarkisk struktur, ett *filträd*:



Filnamn

I praktiken är det inte så rörigt som det ser ut på den förra bilden! Ett fullständigt (absolut) filnamn börjar från rotkatalogen och man räknar upp alla katalogerna med / emellan:

```
/h/d8/c/dat14xyn/dod/lab1/example.txt
```

Men Unix/Linux håller reda på en *aktuell katalog* som man själv kan ändra. Om `lab1` är aktuell katalog så kan man komma åt ovanstående fil med följande namn (*relativt* filnamn):

```
example.txt
```

Om `dat14xyn` är aktuell katalog så kommer man åt filen med namnet:

```
dod/lab1/example.txt
```

Förkortade filnamn

En del filer har förkortade namn:

`~`	(tilde) hemkatalogen för aktuell användare (Alt Gr + ∼)
`~user`	hemkatalogen för användaren med användarnamnet user
`.`	(punkt) aktuell katalog
`..`	(punktpunkt) katalogen ovanför aktuell katalog i filträdet

Antag att `~/dod/lab1` är aktuell katalog. Man kan navigera upp och ned i filträdet enligt följande exempel:

```
~/ptdc-workspace/lab1/Calculator.java
(till hemkatalogen, ner i ptdc-workspace, ner i lab1)

../../ptdc-workspace/lab1/Calculator.java
(upp ett steg, upp ett steg, ner i ptdc-workspace, ner i lab1)
```

Kommandotolk

Man kommunicerar med Unix eller Linux genom att ge kommandon. Ett särskilt program i operativsystemet, kommandotolken, "shell", läser in kommandona och utför det som ska göras.

Det finns flera olika kommandotolkar. På Linuxdatorerna i E-huset används kommandotolken `bash` ("Bourne-Again SHell"). Andra vanliga kommandotolkar är `sh`, `zsh` och `csh`.

Kommandotolken läser kommandon som ges i ett terminalfönster på skärmen. Så här arbetar tolken:

```
Upprepa i all oändlighet:
    Läs ett kommando
    Om kommandot är ett riktigt kommando:
        utför det som ska göras,
    annars:
        skriv ut "XXX: command not found"
```

Kommandoformat

Varje kommando skrivs på en rad:

```
kommandonamn -option1 -option2 ... argument1 ...
```

- Kommandot talar om vad som ska göras.
- Argumenten är (oftast) filer eller kataloger som påverkas av kommandot.
- Optionerna modifierar kommandot på något sätt.

Exempel:

`javac Calculator.java`	Kompilerar `Calculator.java` med Java-kompilatorn, skapar `Calculator.class`
`cp -i report.tex old.tex`	Kopierar `report.tex` till `old.tex`. `-i` betyder att systemet frågar om `old.tex` ska skrivas över om den redan finns

Kommandon för filhantering

Exempel på kommandon för att hantera filer:

`cp orig kopia`	Kopiera filen `orig` till `kopia`.
`less fil`	Skriv ut `fil` på skärmen, en sida i taget.
`ls [-la] kat`	Skriv ut en innehållsförteckning över katalogen `kat` (aktuell katalog om ingen katalog ges). `-l` skriv i ett längre format, `-a` skriv också ut punktfiler.
`mv fil1 fil2`	Döp om `fil1` till `fil2`. Om `fil2` är en katalog så flyttas filen till den katalogen.
`rm fil1 ...`	Tag bort de angivna filerna.

Kataloghantering

Exempel på kommandon för att hantera kataloger:

`cd kat`	Ändra aktuell katalog till `kat`. Utan argument blir det hemkatalogen (samma som `cd ~`).
`mkdir kat`	Skapa underkatalogen `kat` i den aktuella katalogen.
`pwd`	Skriv ut namnet på aktuell katalog.
`rmdir kat`	Tag bort `kat`. Man måste först ta bort alla filerna i katalogen.

Filskydd 1

Varje fil har en ägare. Ägaren identifieras med användarnamn och grupp, till exempel `dat14xyn` i gruppen `students`.

Ägaren kan skydda filer så att andra inte kan komma åt dem eller öppna filer så att andra kan komma åt dem.

Filskydd och ägare (och storlek och datum) skrivs ut om man gör `ls -l`:

```
hacke-3{dat14xyn}: ls -l
-rw-r-----  1 dat14xyn  students  4940 aug 21 11:14 example.txt
  skydd        ägare     grupp
```

Filskydd 2

```
-rw-r-----  1 dat14xyn  students  4940 aug 21 11:14 example.txt
  u   g   o
```

Tre kategorier av användare:

- `u` (user) filens ägare
- `g` (group) medlemmar i samma grupp som ägaren
- `o` (others) alla andra

Tre olika rättigheter:

- `r` (read) tillstånd att läsa
- `w` (write) tillstånd att skriva
- `x` (execute) tillstånd att exekvera program (för kataloger betyder `x` tillstånd att titta på innehållet i katalogen)

Filskydd 3

Kommando för att ändra filskydd:

```
chmod skydd fil1 fil2 ...
```

Filskydd kan anges symboliskt, till exempel `u=rw,o=r`, men också numeriskt, där man anger skydden som är satta med ettor, de som inte är satta med nollor. Till exempel:

```
x = 001, w = 010, r = 100, rx = 101, rw = 110, rwx = 111
```

Siffrorna tolkar man sedan som binära tal, vilket ger:

```
x = 1, w = 2, r = 4, rx = 5, rw = 6, rwx = 7
```

Fullständiga numeriska filskydd blir till exempel:

```
604 <=> u=rw,o=r     705 <=> u=rwx,o=rx
```

bash-finesser 1

Man kan editera den aktuella kommandoraden:

DELETE	Tag bort tecknet till vänster om markören.
CONTROL-U	Radera hela raden.
$\leftarrow$ $\rightarrow$	Flytta sig på raden.
TAB	Filnamnskomplettering (man behöver bara skriva början av ett långt namn).

`bash` håller reda på de senaste kommandona som utförts, och man kan få tillbaka gamla kommandon:

$\uparrow$	Återkalla senaste kommando (kan upprepas).
`!!`	Gör om senaste kommando.
`!abc`	Gör om senaste kommando som inleds med `abc`, till exempel `!javac`

bash-finesser 2

När man refererar till filer kan man utnyttja jokertecken (wildcards):

- `?` motsvarar *ett* godtyckligt tecken
- `*` motsvarar 0–flera godtyckliga tecken

Antag att vi har följande filer i vår katalog: `Test1.java`, `Test2.java`, `Test23.java`, `Final.java`, `Test1.class`, `Test2.class`, `Final.class`, `FinalReport.tex`, `Outline.tex`.

```
javac Test?.java (javac Test1.java Test2.java)
rm *.class       (rm Test1.class Test2.class Final.class)
gedit F*.tex     (gedit FinalReport.tex)
```

Initieringsfiler

Många program läser en initieringsfil när de startas. Initieringsfilen innehåller inställningar för programmet. Namnen på initieringsfilerna inleds med punkt, så man ser normalt inte dessa filer när man gör `ls`.

`bash` läser initieringsfilen `.bash_profile` när ett nytt terminalfönster skapas. Bra rader att ha i denna fil (gör att man alltid får frågor om man verkligen vill ta bort/skriva över filer):

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

Varning: kopiera inte andras initieringsfiler om du inte begriper vad som står i dem!

Vad är egentligen ett kommando?

Kommandon är av två slag:

"inbyggda" små kommandon som exekveras direkt i kommandotolken, till exempel `cd`, `pwd`, `exit`

"vanliga program" kommandotolken letar efter ett program med samma namn som kommandot

Kommandotolken letar efter program i de kataloger som anges i sökvägen (operativsystemvariabeln `PATH`, som man själv kan sätta med kommandot `export`). När man till exempel skriver `javac X.java` exekveras programmet `/usr/bin/javac`, eftersom katalogen `/usr/bin` finns i sökvägen.

När man ska exekvera ett eget program som finns i den aktuella katalogen skriver man:

```
./programnamn
```

`./` är nödvändigt eftersom `.` (den aktuella katalogen) inte finns i sökvägen.

Processer

Varje program som körs i Unix körs som en egen process som exekverar självständigt, ”parallellt” med andra processer. Varje process körs några millisekunder, sedan får nästa process exekvera, osv. Detta hanteras av operativsystemet.

När man exekverar ett program från ett terminalfönster ”låser” programmet fönstret tills det avslutas — så vill man ofta att det ska fungera.

Men man kan också köra program ”i bakgrunden” dvs frikopplade från terminalfönstret. Det gör man genom att skriva & sist på kommandoraden, till exempel:

```
gedit example.txt &
```

Fönstersystem, skrivbordsmiljö

I Unix och Linux är fönstersystemet inte inbyggt (Unix utvecklades långt innan det fanns grafiska skärmar).

De flesta Unix- och Linuxsystem som kör på persondatorer använder fönstersystemet X Window System, som oftast kallas bara X. X gör det möjligt att ha fönster på skärmen, att flytta dem, ikonifiera dem, osv.

Exakt hur fönstren ska se ut och hur man arbetar med dem bestäms av *fönsterhanteraren*, som också är ett separat program. Ovanpå fönsterhanteraren finns skrivbordsmiljön.

Det bästa (enda) sättet att lära sig att använda en skrivbordsmiljö är att träna!