Vecka 1. Introduktion

Programmering, grundkurs (pgk:dod)

Mattias Nordahl

Datavetenskap, LTH, Lunds universitet
https://lunduniversity.github.io/pgk

EDAB05, Lp1-2, HT2026

Kompilerad den 25 augusti 2026

1 Datorer och Datoranvändning: Operativsystem

- Om Datorer och datoranvändning
- Operativsystem

Om Datorer och datoranvändning

Välkommen till Datorer och datoranvändning (dod)

- Syftet med *dod*-delen?
  - Lär dig grundläggande verktyg.
  - Introduktion inför kommande kurser.
- Vem läser kursen?
  - C-programmet.
  - D-programmet.
- Vad innehåller kursen?
  - Fyra ämnen, med vardera en föreläsning och en laboration
    1. Unix och Linux
    2. Maskinkod
    3. LaTeX
    4. Git och Github

Schema och gruppindelning

- Gruppindelning
  - Gruppindelning, se Canvas.
  - Kapten Alloc
    `https://fileadmin.cs.lth.se/pgk/kaptenalloc/`
- Schema
- Föreläsning alltid samma tid (tisdag 15:15), E:A
- Laborationer
  - En labb per vecka, start på torsdag!
  - Föreberedelseuppgifter, kontrollfrågor.
  - Laboration 1, Linux (Första laborationen denna vecka!)
  - Laboration 2, Maskinkod
  - Laboration 3. LaTeX(LP 2)
  - Laboration 4, Git och Github (LP 2)

Kurshemsidan och Github

Den öppna kurshemsidan:

`https://lunduniversity.github.io/pgk/`

Allt kursmaterial finns tillgängligt på GitHub:

- Allt material är **öppen källkod**.
- Skapa gärna ärenden (eng. *issues*) om du upptäcker problem eller vill föreslå förbättringar.
- Förfrågningar om införande av implementerade ändringsförslag (eng. *pull requests*) är hjärtligt välkomna! Öppna gärna först ett ärende för diskussion av dina ändringsförslag.
- Läs mer i kompendiets inledande kapitel om hur du bidrar till kursmaterialet.

Vid sjukdom / förhinder

Om du blir sjuka eller får annat förhinder:

- Meddela mig (dod) `mattias.nordahl@cs.lth.se`
- Meddela Björn (prog) `bjorn.regnell@cs.lth.se`
- Få hjälp och redovisa på resurstid

Vid frågor

- Ni kan alltid maila mig: `mattias.nordahl@cs.lth.se`
- Discord. Hitta invite-länk på Canvas-sidan.

Operativsystem

Förberedelse Lab 1: `linux`

- Operativsystem, Unix historia, Linux
- Filer och kataloger
- Kommandon
- Filskydd
- Kommandotolk
- Processer

Operativsystem

- Dator — kan köra program
- Program — instruktioner för datorn
- 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
- Exempel:
  - Linux, Windows, Mac OS X, Unix, . . .

Unix historia

- 1960-talet: Multics, ett stort projekt avsedd för stora datorer.
- 1969: Ken Thompson på AT&T Bell Labs utvecklar Unix, ett enklare system för mindre datorer med inspiration från Multics.
- 1973: Unix omskrivs i C av Dennis Ritchie, vilket underlättar distribution och anpassning.
- Utvecklingen fortsätter med många varianter: Linux, BSD, Mac OS X, och Android bland andra.
- Unix varumärket ägs av The Open Group som certifierar Unix-kompatibla system (Linux är ej certifierat).

Linux och GNU

- 1983: GNU-verktygen – fritt Unix-liknande operativsystem.
- 1991: Linus Torvalds skapar Linuxkärnan – hanterar hårdvaruinteraktioner.
- Linuxkärnan + GNU-verktyg = GNU/Linux – skapar en fullständig användarmiljö.
- ”Linux” används ofta som kortform för hela systemet.

Kännetecken för Unix

Grundläggande:

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

Unix är:

- Litet, standardiserat, flyttbart
- (Ursprungligen) Inte för nybörjare. Enklare nu med grafiska skrivbordsmiljöer.

Undersökning OS

Vilket operativsystem använder du?

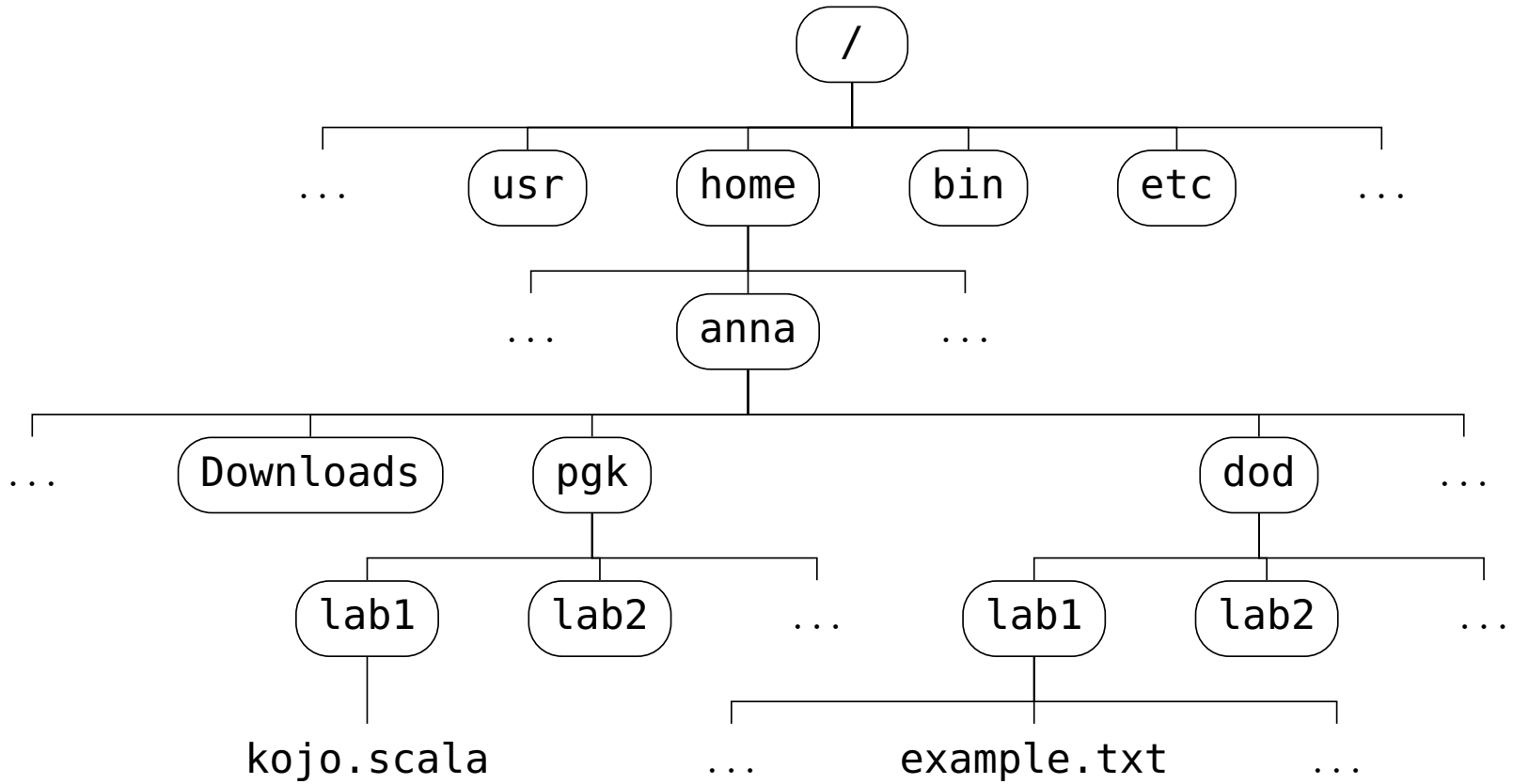
Mentimeter eller handuppräckning

Filer i Unix/Linux

- I en dator måste man kunna lagra
  - program (Word, Excel, Java-/Scalakompilator, spelprogram, . . .)
  - data (foton, dokument, programmeringsuppgifter, . . .)
- Program och data lagras i *filer*, som har ett namn:
  - Filer ligger alltid i en **katalog** (eng. *directory*), även kallad **mapp** (eng. *folder*).
  - Filer och kataloger lagras på ”permanent” minne (hårddisk, SSD, USB-minne, . . .).
- Filnamn i Unix/Linux:
  - `namn.tillägg`
    Exempel: `Calc.scala`, `Calc.class`, `brev.txt`, `README.md`, `...`
  - Tillägget är bara en extra upplysning; filtypen bestäms av det program som läser filen.
  - Filnamn kan sakna tillägg, exempel: `test1`

Kataloger

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



Filnamn och sökvägar

Termer:

- Sökväg (eng. path) – vägen till en fil
- Absolut sökväg – från rotkatalogen
- Relativ sökväg – från aktuell katalog
- Aktuell katalog – *där du är i filträdet*

Absolut filnamn:

```
/home/anna/dod/lab1/example.txt
```

Relativt filnamn:

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

På skolans datorer ser de översta delarna av filträdet annorlunda ut; en absolut sökväg för en användare med student-id `an1337sv-s` kan se ut ungefär såhär:

`/h/d5/r/an1337sv-s/dod/lab1/example.txt`

Förkortade filnamn

Förkortningar:
Vissa filnamn har speciella betydelser:

`~`	Hemkatalog (tilde)
`~user`	Användarens hemkatalog (tilde user)
`.`	Aktuell katalog (punkt)
`..`	Överordnad katalog (punktpunkt)

Exempel:
Om `~/dod/lab1` är aktuell katalog:

```
~/pgk/lab1/kojo.scala
```

Betydelse: till hemkatalogen, ner i pfk, ner i lab1

```
../../pgk/lab1/kojo.scala
```

Betydelse: upp ett steg, upp ett steg, ner i pgk, ner i lab1

Kommandotolk (shell, ”skal”) och terminal

Termerna *kommandotolk* och *terminal* används ofta synonymt, men har olika betydelser:

- **Kommandotolk**: För kommunikation med operativsystemet via kommandon
- **Terminal**: Program som låter användaren skriva kommandon till kommandotolken
- **Shell**: Engelska för kommandotolk, *skal* på svenska

Exempel på kommandotolkar:

- `bash` (Bourne-Again SHell)
- `sh`, `zsh`, `csh`

Kommandotolk (shell, ”skal”) och terminal

Så här fungerar en kommandotolk:

```
Upprepa i all oändlighet:
  Läs ett kommando
  Om kommandot är ett riktigt kommando:
    utför det som ska göras
  annars:
    skriv ut felmeddelande
```

https://youtu.be/zQ-e9SuoWXo

Bash vs. Zsh för Mac-användare

Bash och Zsh:

- Bash är standard i de flesta Linux-system.
- Zsh är standard i macOS från Catalina.
- Båda är kraftfulla skal i Unix-system.
- För grundläggande användning, t.ex. denna kurs, fungerar båda bra.

Att byta till Bash (valfritt):

- Öppna Terminal och skriv `bash` för att temporärt byta till Bash.
- Installera senaste Bash via Homebrew: `brew install bash`

Skillnader:

- Zsh upplevs av många som mer användarvänligt. Erbjuder fler funktioner och anpassningsmöjligheter, men kan vara överväldigande för nybörjare.

Unix-shellalternativ för Windows-användare

Git Bash:

- Enkel Bash-tillgång, minimal konfiguration.
- Installeras med Git för Windows.

WSL (Windows Subsystem for Linux):

- Kör fullständig Linux i Windows.

Cygwin:

- Ger bred Unix-vertygsuppsättning.
- Mer konfiguration.

Vilket ska jag välja?

- **Git Bash**: Grundläggande Bash och Git. (Rekommenderas)
- **WSL**: Fullständig Linux-utveckling. (Rekommenderas)
- **Cygwin**: Omfattande Unix-funktionalitet. (Avråds)

Kommandoformat

Varje kommando skrivs på en rad:

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

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

Exempel:

`scala run Calc.scala`	Kompilera och kör `Calc.scala`
`scala run .`	Kompilera och kör alla Scala-filer i aktuell katalog och alla dess underkataloger
`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. Hakparenteserna indikerar att optionerna är valfria.
`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 1/3

Filägande:

- Ägare identifieras via användarnamn och grupp (t.ex., `ma1337no-s` i `students`)

Filsäkerhet:

- Ändra åtkomsträttigheter för att skydda eller dela filer

Visa detaljer:

```
hacke-3{ma1337no-s}: ls -l
-rw-r-----  1 ma1337no-s  students  4940 aug 21 11:14 example.txt
(skydd)       (ägare)     (grupp)
```

Filskydd 2/3

```
-rw-r-----  1 ma1337no-s  students  4940 aug 21 11:14 example.txt
- u  g  o
```

Första tecknet är antingen - (vanlig fil) eller d (katalog).
Därefter, tre kategorier av användare:

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

Som vars har 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 3/3

Kommando för att ändra filskydd:

```
chmod skydd fil1 fil2 etc
```

Filskydd kan anges symboliskt, till exempel `u=rw,o=r`

Eller numeriskt, enligt:

```
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
```

Exempel numeriska filskydd och motsv. symboliska:

```
chmod 604 fil1 fil2          chmod u=rw,o=r fil1 fil2
chmod 755 fil1 fil2          chmod u=rwx,g=rx,o=rx fil1 fil2
```

bash-finesser 1

Man kan editera den aktuella kommandoraden:

← →	Flytta markören på raden.
`Control-K`	Radera hela raden.
`Control-U`	Radera allt till vänster.
`Control-L`	Töm terminalfönstret.
`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:

↑ ↓	Bläddra bland tidigare kommandon.
`!!`	Gör om senaste kommando.
`!abc`	Gör om senaste kommando som inleds med `abc`.

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
```

Då kan vi till exempel skriva:

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

Initieringsfiler

Om initieringsfiler:

- Används ofta för att konfigurera program.
- Filnamn börjar med punkt och är dolda, s.k. *punktfiler*.

`bash` initieringsfil:

- bash läser `.bash_profile` (från hemkatalogen) när ett nytt terminalfönster öppnas.
- Nyttiga alias för att förhindra oavsiktliga filoperationer:

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

Varning:

- Kopiera inte initieringsfiler utan att förstå innehållet!

Vad är egentligen ett kommando?

Typer av kommandon:

Inbyggda Kör direkt i kommandotolken (t.ex. `cd`, `pwd`, `ls`).

Vanliga program Kommandotolken söker efter programfiler i `PATH`.

Programexekvering:

- `PATH` definierar var program letas upp (`export PATH=...` för anpassning).
- Ex: `/usr/bin/javac` finns i `PATH`, så `javac X.java` går att köra.

Köra egna program:

```
./programnamn
```

Använd `./` för program i den aktuella katalogen, eftersom `.` inte ingår i `PATH`.

Processer

Grundläggande om Processer:

- Varje program körs som en egen process, oberoende och ”parallellt” med andra.
- Processväxling sker kontinuerligt, hanteras automatiskt av operativsystemet.

Exekvering i Terminal:

- Standard: Kommandotolken väntar på programmet. Terminalen framstår som upptagen tills programmet avslutas.
- Bakgrundskörning: Använd & för att köra programmet frikopplat från terminalen.

Exempel:

```
gedit example.txt &
```

Öppnar `gedit` i bakgrunden, terminalen blir omedelbart tillgänglig för nya kommandon.

Fönstersystem, skrivbordsmiljö

Fönstersystem:

- I Unix och Linux är fönstersystemet inte inbyggt (Unix utvecklades långt innan det fanns grafiska skärmar).
- Ett äldre fönstersystem är X Window System (ofta kallat *X*).
- Ett nyare fönstersystem är Wayland (används i Ubuntu).

Fönsterhanterare och skrivbordsmiljö:

- Fönsterhanteraren bestämmer utseende och funktion hos fönstren.
- Skrivbordsmiljön bygger vidare på fönsterhanteraren för en fullständig användarupplevelse.

Lära sig skrivbordsmiljön:

- För många av er är Linux nytt. Bästa sättet att bli bekant med en skrivbordsmiljö är att prova sig fram och träna!
- I Ubuntu: Tryck på windowsknappen och skriv ”help” så får du fram **Ubuntu Desktop Guide** med en översikt av skrivbordsmiljön.
- Lär dig gärna kortkommandon för funktioner du använder extra ofta, se ”Tips & Tricks -> Useful keyboard shortcuts” i desktop-gajden.