TTK4235 - Tilpassede datasystemer
Vår 2024

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Øving 1
Introduksjon til Linux

Revisjonshistorie

År	Forfatter
2020	Kolbjørn Austreng
2021	Kiet Tuan Hoang
2022	Kiet Tuan Hoang
2023	Kiet Tuan Hoang
2024	Terje Haugland Jacobsson Tord Natlandsmyr

1 Introduksjon - Praktisk rundt filene

I denne øvingen får dere utlevert en `.c`-fil i `source`-mappen. Tabellen under lister opp filen som kommer med i `source`-mappen samt litt informasjon om dere skal endre på filen eller om dere skal la den bli i løpet av øvingen.

Filer	**Skal filen(e) endres?**
`source/main.c`	Ja
`Main/*`	Nei
`.github/*`	Nei

2 Introduksjon - Praktisk rundt øvingen

Denne øvingen gir en innføring i Linux, slik at resten av labopplegget fremover ikke foregår i et helt ukjent miljø. Seksjon 4 gir en innføring i grunnleggende kommandoer som er nyttige i Linux, og som trengs for å fullføre oppgavene som følger i seksjon 5 . Dere får godkjent øving 1 ved at dere demonstrerer foran en studass at dere har gjort alle oppgavene i seksjon 5 . I tillegg har to appendikser blitt lagt til (se appendiks A og B) i tilfelle det er noen som enten har lyst til å installere Linux på egen maskin med dual booting, eller jobbe på et Unix-liknende omgivelse for Windows med Cygwin.

3 Introduksjon - Litt om GNU/Linux

Linux (Linux kernel) ble først utviklet av Linus Torvalds på 90-tallet da han studerte ved Universitetet i Helsinki. Linus ble interessert i utvikling av operativsystemer gjennom sine studier, men ble frustrert av datidens lisensbegrensninger på tilgjengelige operativsystemer. I 1991 begynte han å utvikle kjernen (kernel) til det som skulle bli til Linux-kjernen som vi kjenner den i dag. En operativsystemkjerne fungerer som et bindeledd mellom maskinvare og programmer. Den er ansvarlig for å starte systemet og å tildele ressurser til programmer. En kjerne er i seg selv ikke særlig nyttig uten resten av operativsystemet, slik som enhetsdrivere, brukerprogrammer, kompilatorer, osv. Heldigvis eksisterte allerede GNU-prosjektet, som også hadde som mål å tilby fri (som i frihet)[1] programvare, og i grunn bare manglet en velfungerende kjerne. Ved å kombinere Linux-kjernen med GNU-programvare ble Linux til et fullverdig operativsystem som mange i dag omtaler som "GNU/Linux".

GNU/Linux kommer i dag som flere ulike varianter, eller distribusjoner, avhengig av hvilken programvare som brukes. Noen av de mest populære distribusjonene er Debian, Fedora Linux og Arch Linux. Datamaskinene dere skal bruke på lab bruker Ubuntu, som er en variant av Debian med et mer begynnervennlig brukergrensesnitt.

Et av kjerneprinsippene til både GNU- og Linux-prosjektet er at programvare, samt tilhørende kildekode, skal være åpent tilgjengelig og respektere brukerens friheter til å bruke, endre og (re)distribuere den. Dette gjelder også utviklingen av Linux. Hvem som helst kan bidra til Linux-kjernen, og kun et fåtall utviklere jobber med å vedlikeholde prosjektet profesjonelt. Projektenes dugnadsånd har vært en stor faktor i GNU/Linux sin utvikling, og holdningen har spredd seg til å bli en hjørnestein i kulturen til dagens programvareutviklere. Dette er også en av flere grunner til at programvareutviklere foretrekker å bruke Linux både til daglig bruk og profesjonelt. GNU/Linux brukes i dag av Android, ChromeOS, majoriteten av tjenere på internett og en stor andel av alle tilpassede datasystemer. SpaceX sin Falcon 9, NASA sin Perseverance og Tesla sine biler er bare noen få av alle nevneverdige Linux-brukere.

4 Innføring - Grunnleggende kommandoer

Denne seksjonen dekker stort sett det som kreves for å bruke Linux i de kommende øvingene/labene. Ubuntu er varianten/distribusjonen som er installert på Sanntidssalen, men alle kommandoene i denne seksjonen gjelder for enhver distribusjon av Linux-baserte operativsystemer.

Ubuntu har i likhet med Windows et grafisk brukergrensesnitt, men det er stort sett raskere å bruke en terminal når bruken er kjent. Man kan åpne terminalen ved å trykke `Ctrl + Alt + T` på tastaturet, eller ved å åpne *Dash* (øverst til venstre)

[1] "Free as in freedom, not as in beer" - Richard Stallman, Grunnlegger av GNU Prosjektet

og søke etter `terminal`.

Terminalen åpner seg i hjemmemappen, og prompten ser slik ut (avhengig av hvem som har gjort endringer på maskinen før, kan den være annerledes):

```
student@Ubuntu:~$
```

Tildesymbolet (~) forteller brukeren at man er i hjemmemappen, og `$` forteller at man er en vanlig bruker - i motsetning til *root* som er en allmektig bruker (symboliseres med `#`).

4 .1 `pwd`

Første kommando som kan være lurt å vite om er `pwd`. `pwd` brukes for å få en oversikt over hvilken mappe man befinner seg i. Om man kaller `pwd` på sanntidssalen, burde man få noe som ligner på dette:

```
student@Ubuntu:~$ pwd

/home/student
```

4 .2 `ls`

Videre, for å vise innholdet i mappen man befinner seg i kan man bruke kommandoen `ls`. Om man kaller `ls` fra hjemmemappen vil man typisk se disse mappene:

```
student@Ubuntu:~$ ls

Documents Downloads Pictures Music
Public Videos Desktop Templates
```

Dette er de samme mappene som kan ses i Ubuntu sin filutforsker (åpnes ved å trykke ikonet til venstre, eller ved å søke `nautilus` i *dash*). I tillegg kan man inspisere innholdet i en spesifikk mappe ved å kalle `ls mappenavn`.

Typisk med kommandoer i Linux er at man kan legge til flagg i kommandoene for å endre oppførselen til kommandoen. Et flagg som er nyttig for `ls` er `-l`. Ved bruk av `-l`, listes det opp hvilke typer filer som befinner seg i mappen, hvem som har brukerrettigheter, hvem som eier filene, filstørrelse, sist modifikasjonsdato, og navn. For eksempel:

```
student@Ubuntu:~$ ls -l

total 48
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Documents
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Downloads
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Pictures
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Music
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Public
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Videos
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Desktop
drwxr-xr-x 2 student student 4096 Nov 18 01:49 Templates
```

Formatet som `ls -l` produserer, er illustrert i figur 1. Den første bokstaven (`d`) betyr at filen er en mappe[2]. Deretter følger noen bokstaver som forteller hvem som har rettighetene til å endre filen. En `r` betyr at man har leserettigheter, en `w` betyr at man har skriverettigheter, og en `x` betyr at man kan kjøre filen.

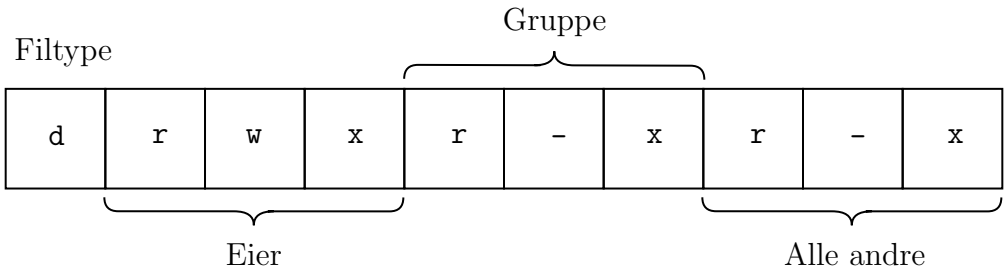

Figure 1: Filegenskaper fra `ls -l`.

I eksempelet i figur 1 kan eieren gjøre hva han eller hun vil, mens gruppen som eier filen og alle andre kan gjøre alt bortsett fra å endre på filen (mangel på `w`).

Etter denne filbeskrivelsen kommer et tall (`2` i eksempelet). Dette er antall *hardlenker* filen har, og har med hvordan filen er lagret på. Deretter følger navn på eieren av filen (`student` i eksempelet), og gruppen som eier filen (også `student` i eksempelet - disse trenger ikke være like). Tallet `4096` som følger hver linje i eksempelet er antall byte som filen okkuperer - men ettersom mapper egentlig bare er pekere som forteller hvilke filer den inneholder, er dette tallet bare størrelsen på pekeren - altså ikke størrelsen på det mappen inneholder. Til slutt kommer en timestamp av når filen sist ble endret, og navnet på filen.

4 .3 `cd`

Kommandoen `cd` er en av de viktigste kommandoene i Linux og blir brukt for å navigere inn- og ut fra mapper. Eksempelvis kaller man `cd Downloads` om man ønsker å bevege seg inn i mappen `Downloads`.

[2]I Linux er mapper også betraktet som filer.

For å bevege seg et nivå opp bytter man ut mappenavnet med to punktum, slik at `cd ..` blir kommandoen. For eksempel om man er i mappen `Downloads` vil `cd ..` navigere brukeren til hjemmemappen.

I Linux har to punktum blitt definert som foreldremappen, mens ett punktum definerer mappen som brukeren befinner seg i. Dette betyr at kommandoen `cd .` flytter brukeren inn i mappen som man allerede befinner seg i.

`cd -` er en annen viktig kommando. `cd -` brukes for å navigere brukeren tilbake til der man sist var.

4 .4 `mkdir`

En nyttig kommando for videre bruk er `mkdir` (*make directory*). Dette er en kommando som brukes for å opprette en mappe. Om man vil opprette en mappe kalt `demo` og bevege seg inn i den kan man eksempelvis kalle:

```
student@Ubuntu:~$ mkdir demo
student@Ubuntu:~$ cd demo
```

4 .5 `touch`

Kommandoen `touch` angir endrings- og tilgangstider for filer. Hvis en fil ikke finnes, opprettes den med standardtillatelser.

```
student@Ubuntu:~$ touch newfile.c
student@Ubuntu:~$ ls

Documents Downloads Pictures Music
Public Videos Desktop Templates newfile.c
```

4 .6 `file`

En annen nyttig kommando er `file`. `file` brukes for å skaffe seg informasjon om en fil. For eksempel:

```
student@Ubuntu:~$ file Downloads
Downloads: directory

student@Ubuntu:~$ file main.c
main.c: C source, ASCII text
```

4 .7 `cat`

Gitt nå at man har lyst til å ta en titt på innholdet i `main.c` som man allerede vet er en `c`-fil på grunn av kommandoen `file`. Da kan man bruke kommandoen

`cat`. `cat` tar innholdet i en fil og skriver det ut i terminalen. For eksempel:

```
student@Ubuntu:~$ cat main.c

#include <stdio.h>

int main(){
    printf("Welcome to TTK4235 - Embedded Systems\n");
    printf("Please be aware of the fact that most animals, especially
        ferrets, are not allowed in Sanntidssalen.\n");
    printf("Smaller birds, e.g. pigeons, may in some very rare cases be
        permitted.\n");
    return 0;
}
```

4 .8 `man`

`man`, kort for *manual page* er den viktigste kommandoen i Linux. Denne kommandoen brukes for å lære om bruken av en hvilken som helst kommando. Et relevant eksempel er kommandoen `man ls` som brukes for å generere en liste over hvilke flagg som `ls` støtter.

I tillegg er det også mulig å kalle `man man` for å få enda mer informasjon om hvordan man kan bruke `man`. Dersom dette blir gjort, kan man se at `man` grupperer manualene i 9 kategorier - hvor kategori 3 er bibliotekkall.

Disse kategoriene kan brukes for å spesifisere hvilke kommandoer man vil ha mer informasjon om, dersom det er flere programmer eller bibliotekkall med samme navn. For eksempel har man programmet `printf`, som man kan få informasjon om ved å kalle `man printf`. Om man derimot vil ha dokumentasjon på funksjonen med samme navn i `C`, kan man kalle `man 3 printf`.

4 .9 `sudo`

Som vanlig bruker kan man ikke bruke alle kommandoer. Noen kommandoer er forbeholdt brukere med ekstra rettigheter (*root*). Om man får `permission denied` når man kaller en kommando, kan man midlertidig eskalere brukerrettighetene til *root*-rettigheter ved å kalle kommandoen med `sudo` foran (kort for **su**per**u**ser **do**).

4 .10 `apt`

Til slutt er det verdt å vite hvordan man installerer nye programmer og pakker. De fleste Linux-distribusjoner bruker en pakkemanager for å håndtere installerte programmer. På denne måten har man en sentralisert løsning for å installere og oppdatere programvarer.

I Ubuntu bruker man `apt` som pakkemanager. For eksempel kan man installere programmeringsspråket `Ruby` sin interpreter ved å kalle `sudo apt install ruby`. Informasjon om hva programvaren `ruby` inneholder kan fås ved å kalle `apt show ruby`.

For å oppdatere alle installerte programvarer til nyeste versjon som Ubuntu har tilgjengelig kaller man `sudo apt update`, etterfulgt av `sudo apt upgrade`. Kommandoen `update` oppdaterer listen over tilgjengelig programvarer som kan oppdateres, mens kommandoen `upgrade` installerer de nyeste oppdateringene. For mer informasjon om hvilke kommandoer `apt` støtter kan man kalle `man apt`.

En ting som er verdt å vite om er at Ubuntu tester ut nye pakker en stund før de legges til oversikten som `apt` har tilgang til. Med andre ord kan det ta en stund før nyeste versjon av programvare kommer til Ubuntu.

4 .11 `nano`

Når man jobber med GNU/Linux er det ikke alltid gitt at man kan bruke tekstredigeringsprogrammer som *Word*, *Visual Studio Code* eller *Notepad++*. Derfor er det lurt å kunne bruke verktøy som fungerer *uten* et grafisk brukergrensesnitt. `nano` er et enkelt tekstredigeringsvektøy som fungerer i terminalen.

```
student@Ubuntu:~$ nano main.c
```

Nederst i terminalvinduet vil du se enkelte kommandoforslag som `^X` for *Exit*. Kontraintuitivt betyr `^` at vi skal bruke `Ctrl`. Dette betyr at for å gå ut av `nano` må vi taste `Ctrl` og `X` etter hverandre. For å forkaste eller lagre endringene du har gjort i en fil må du deretter taste `Y` for "Yes" eller `N` for "No".

4 .12 Kommentar til Tekstredigering

Det finnes veldig mange populære programmer for tekst- og filredigering for GNU/Linux. Noen av dem er `vim`, `neovim`, `emacs`, `gedit`, `sublime-text`, `notepadqq`, `VSCode`[3] og `nano`. Disse varierer både i grensesnitt og brukervennlighet. For små endringer i filer kan alternativer som `nano` og `vim` være gode programmer. Dersom du ikke har noe tidligere erfaring med GNU/Linux anbefaler vi at du bruker enten VSCode eller open-source programmet VSCodium[4] for mer omfattende endringer. Disse programmene er *source-code editors* skreddersydde for programvareutvikling. Videre i øvingsopplegget kommer vi til å bruke `VSCode` som eksempel på hvordan vi kan bruke verktøyene vi lærer om i et utviklingsmiljø.

5 Oppgaver (100 %) - Grunnleggende Linux

a Naviger til rotmappen (`cd /`), og bruk `ls` og `cd` for å komme tilbake til egen hjemmemappe uten å bruke `cd -`.

[3] Kort for *Visual Studio Code*, laget av Microsoft
[4] VSCode er basert på VSCodium

b I Linux er det mange filer som starter med et punktum i navnet. Ulempen med punktum i navnet er at disse ikke blir vist i filutforskeren eller av `ls` med mindre man spesifikt ber om det. Bruk `man` til å finne hvilket flagg som `ls` krever for å vise alle filene i en mappe.

c Linux kan *pipe* output fra en kommando inn i en annen kommando. Et eksempel på det er `cat main.c | less`, hvor returverdien av `main.c` blir gitt som input til kommandoen `less`. Lær mer om `less` med `man` og prøv `cat main.c | less` med `main.c` i `source`-mappen.

d Filutforskerprogrammet som kommer med Ubuntu heter *nautilus*, og kan startes fra terminalen ved å kalle `nautilus`. Prøv å kalle `nautilus . &` hvor punktum blir brukt som argument, mens `&` blir brukt for at terminalen skal starte *nautilus* i bakgrunnen. Bruk `man` til å finne hvilke flagg som kreves for å avslutte `nautilus`.

e Opprett en ny fil med `touch` og *pipe* innholdet fra `main.c` inn i den nye filen. Bekreft at du har kopiert innholdet fra `main.c` over til den nye filen ved å bruke tekstredigeringsverktøyet `nano`.

f Erstatt en av tekststrengene i `printf` med *Linux er lett!* i filen du opprettet i oppgave **e**. *Hint:* Bruk `nano`.

g Oppgrader alle pakkene på datamaskinen på sanntidssalen med `apt`.

A Appendiks - Installasjon av Linux

Det er mulig å installere Linux på egen maskin i tillegg til å bruke maskinene på Sanntidsalen. Dette gjøres enten ved å slette operativsystemet man har nå for så å installere Linux, eller gjennom en ordning kalt *dual booting*. Sistenevnte er spesielt populært og betyr ganske greit at man har to eller flere operativsystemer installert på en datamaskin, så velger man hvilket operativsystem man vil *boote* inn ved oppstart.

Installasjonsprosessen er litt forskjellig fra maskin til maskin, og Linux-variant, men her er en generell beskrivelse av installasjons-prosessen.

A.1 Last ned operativsystemet

Det første man trenger er et *image* av operativsystemet man vil installere. Om dette er første gang man er borti Linux før, er Ubuntu anbefalt. Fra https://ubuntu.com/download kan man laste ned nyeste LTS-utgave (**L**ong **T**erm **S**upport). Dette kommer som en `iso`-fil som må overføres til et oppstartsbart medium - typisk en minnepenn.

A.2 Kopier `iso`-filen til et oppstartsmedium

For å transformere minnepennen til et oppstartsmedium bruker man enten programmer som UNetbootin dersom man kommer fra Windows eller Mac. Om man har tilgang på en annen Linux-maskin kan også kommandoen `dd` brukes:

1. Koble minnepennen i datamaskinen og kall `lsblk` for å se hva minnepennen ble registrert som - typisk som `sdb`.
2. Kall `dd if=~/Downloads/ubuntu[...]amd64.iso of=/dev/sdb bs=4M status=progress`. Om isofilen ligger i en annen mappe enn i `Downloads` endrer man på `if`-adressen. Tilsvarende endrer man på `of`-adressen om minnepennen ble registrert som noe annet enn `sdb`.

A.3 Start fra oppstartsmediet

Etter at man har laget et oppstartsmedium, kan man starte datamaskinen fra denne. Hvordan dette gjøres, varierer sterkt, men ofte kommer man inn i maskinens *boot meny* ved å trykke `F12`, `F8`, eller `F2` mens maskinen *booter*.

På nyere utgaver av Windows kan maskinen også ha *UEFI*-beskyttelse, som først må skrus av før man kan starte fra et oppstartsmedium som ikke først er godkjent av Microsoft. Linux er *ikke* godkjent av Microsoft, så vi må skru av *UEFI*-beskyttelsen.

A.4 Installasjon av Ubuntu med dual booting

Når man først får startet fra oppstartsmediumet, er det bare å prøve ut systemet som man vil, eller installere ved å følge *wizarden* som dukker opp.

B Appendiks - Cygwin

Cygwin er et alternativ til dual booting av Linux dersom man vil fortsatt jobbe på en Window maskin. Dette er en stor samling av GNU og Open-Source verktøy som gir en funksjonalitet som likner veldig på en Linux-distribusjon på Windows.

B.1 Last ned Cygwin

For å installere Cygwin, må man først laste ned installasjonsprogrammet ved å gå inn på Cygwin sin nettside (https://www.cygwin.com/). Installasjonsprogrammet heter `setup-x86.exe` (for 32 bits Windows) eller `setup-x86_64` (for 64 bits Windows). Det kan være lurt å lagre installasjonsprogrammet, ettersom den kan brukes til å oppdatere Cygwin senere.

B.2 Installasjon av Cygwin

For å installere Cygwin, må man kjøre installasjonsprogrammet. Selve installasjonsprosessen er veldig enkel, det eneste man trenger å passe på er å velge **Install from Internet**. Etter dette er det bare å trykke **Next**, **Next**, **Next** og **Next**. Deretter har man muligheten til å velge distribusjonsstedet. I teorien kan man også bare presse **Next** på dette steget. Når man har gjort dette, kommer det opp et vindu. For å installere pakkene man trenger, kan man enten søke etter pakken (se figur 2), eller så kan man bare lete nedover ved å ekspandere de ulike kategoriene. For å informere Cygwin om å installere en pakke, må man dobbelttrykke på **Skip**, helt til man får en versjon opp (i figur 2 installer man `make` ved at man inkluderer `make version 4.3.1`). Deretter trykker man **Next** og **Finish**.

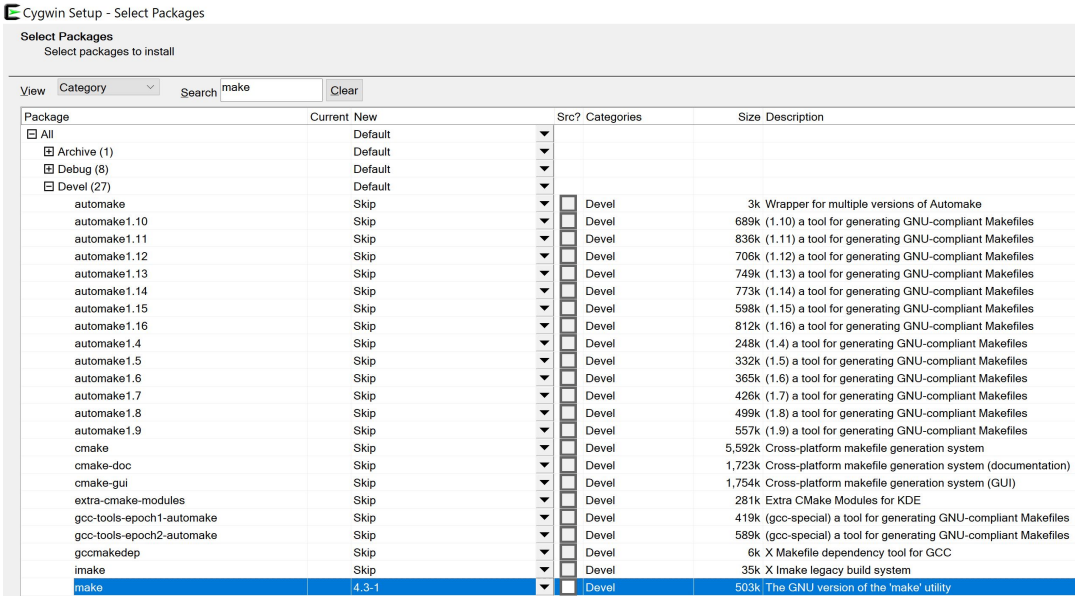

Figure 2: Eksempel på installasjon av pakken `make` i Cygwin.

B.3 Oppdatering av pakker etter installasjon

Det er fort gjort å glemme å installere en pakke. Da kjører man simpelthen installasjonsprogrammet igjen. Like enkelt er det å oppgradere pakker. Installasjonsprogrammet holder styr på det man allerede har installert, og sammenlikner det med det som ligger på det distribujsonsstedet som man velger.