

Aarhus University

4. Semesterprojekt

## **IKN OPGAVE 12**

*Author:*

Søren Landgrebe (201508295)  
Stinus Lykke Skovgaard  
(201401682)  
Daniel Tøttrup (201509520)

*Supervisor:*

Torben Gregersen

May 16, 2017



# TABLE OF CONTENTS

1	Opgaveformulering	1
2	Link::send	3
3	LINK::recieve	4
4	Transport::send	5
5	Transport::recieve	6
6	Applikationlaget::client	7
7	Applikation::server	8
8	Samlet	9

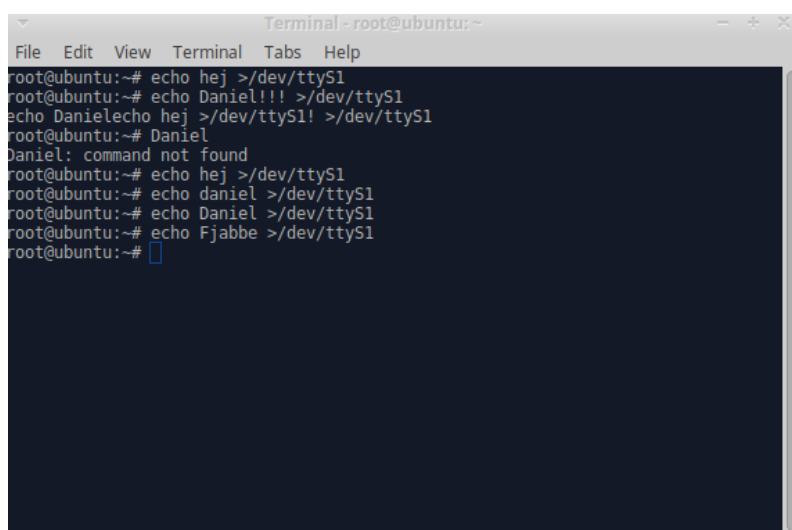
# LIST OF FIGURES

1.1	Test vha minicom v. maskine1 . . . . .	1
1.2	Test vha minicom v. maskine2 . . . . .	2

## OPGAVEFORMULERING

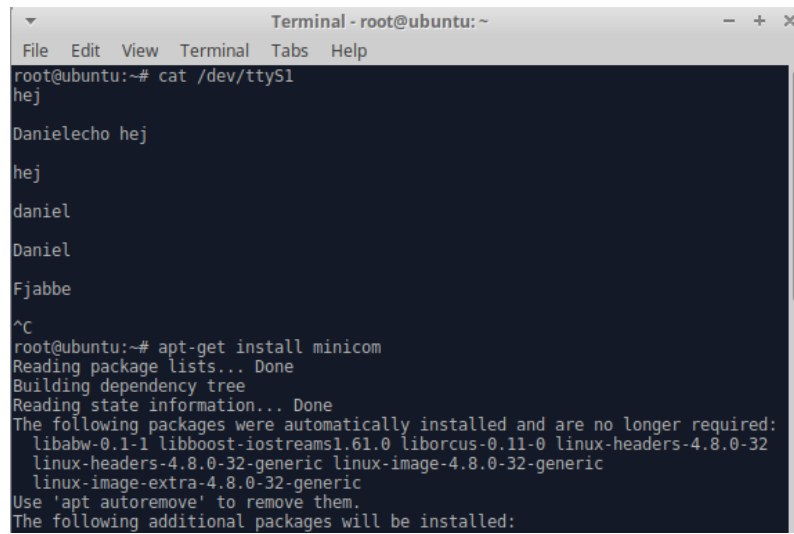
I denne opgave skal vi udvikle en protokol stack, som kan overføre en fil vha. den serielle port i vores virtuelle maskine. Vi bruger til øvelsen RS-232, som er en seriel digital datakommunikation. I opgaven, skal vi udvikle både en client og en server som hhv. skal stå for at sende og modtage en bestemt fil. Vi skal opbygge hele protokollen og derved hele laget, som en TCP-protokol. Dette indebærer linklaget, transportlaget og applikationslaget, som skal implementeres på både client og server. Hertil skal vores link lag, gøre brug af SLIP-protokollen. Vi bruger opgave 7 som grundlag for, at lave et applikationslag, som vi kun behøver at modificere for at kunne bruge til denne opgave.

Vi tester først vores serielle port, for at sikre os vi har opsat porten korrekt og at vi kan skabe en forbindelse mellem de to virtuelle maskiner.



```
Terminal - root@ubuntu:~  
File Edit View Terminal Tabs Help  
root@ubuntu:~# echo hej >/dev/ttyS1  
root@ubuntu:~# echo Daniel!!! >/dev/ttyS1  
echo Danielecho hej >/dev/ttyS1! >/dev/ttyS1  
root@ubuntu:~# Daniel  
Daniel: command not found  
root@ubuntu:~# echo hej >/dev/ttyS1  
root@ubuntu:~# echo daniel >/dev/ttyS1  
root@ubuntu:~# echo Daniel >/dev/ttyS1  
root@ubuntu:~# echo Fjebbe >/dev/ttyS1  
root@ubuntu:~#
```

Figure 1.1: Test vha minicom v. maskine1



```
Terminal - root@ubuntu: ~
File Edit View Terminal Tabs Help
root@ubuntu:~# cat /dev/ttyS1
nej
Danielecho nej
nej
daniel
Daniel
Fjabbe
^C
root@ubuntu:~# apt-get install minicom
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libabw-0.1-1 libboost-iostreams1.61.0 liborcus-0.11-0 linux-headers-4.8.0-32
  linux-headers-4.8.0-32-generic linux-image-4.8.0-32-generic
  linux-image-extra-4.8.0-32-generic
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
```

Figure 1.2: Test vha minicom v. maskine2

## LINK::SEND

I vores link send, vil vi implementere SLIP-protokollen som går ud på at vi starter og slutter vores data, altså fram'er hele dataprotokollen med A, for at undgå støj, og derved sikre at vi kender et start og slut punkt for data-bufferen. Vi vælger først at sætte den første plads i vores nye data-array til 'A', så vi ved vi starter med et A. Herefter erstatter vi alle steder vi har et A, med et BC, og et B med et BD. Vi bliver dog nød til at flytte det hele en plads i arrayet, for at få et ekstra karakter ind, dette gøres ved at tælle bufferen op. Alt dette gør vi, da vi kun vil bruge A som start og slut bit, og derfor bliver nød til at erstatte det fra inputbufferen. Hvis vi får et 0 fra inputbufferen, er besked slut, og vi tilføjer derfor et A til bufferen. Vi sikre dog den ikke tilføjer uendelige A, ved at lave en Acount, som bliver brugt som et flag, der bliver sat højt, når vi har lavet den sidste bit om til et A. Hvis vi modtager andet end fra vores inputbuffer end A eller B, bliver det placeret i den nye buffer.

```
void Link::send(const char buf[], short size)
{
    int j = 0;
    buffer[0] = 'A';
    int Acount = 0;

    for(int i = 0; i < size-1; ++i)
    {
        ++j;
        if(buf[i] == 'A')
        {
            buffer[j] = 'B';
            ++j;
            buffer[j] = 'C';
        }
        else if(buf[i] == 'B')
        {
            buffer[j] = 'B';
            ++j;
            buffer[j] = 'D';
        }
        else if (buf[i] == 0 && Acount == 0)
        {
            buffer[j] = 'A';
            Acount = 1;
        }
        else
            buffer[j] = buf[i];
    }
    v24Write (serialPort, (unsigned char *)buffer, strlen(buffer));
}
```

## LINK::RECEIVE

I vores link send, vil vi implementere SLIP-protokollen som går ud på at vi starter og slutter vores data, altså fram'er hele dataprotokollen med A, for at undgå støj, og derved sikre at vi kender et start og slut punkt for data-bufferen. Vi vælger først at sætte den første plads i vores nye data-array til 'A', så vi ved vi starter med et A. Herefter erstatter vi alle steder vi har et A, med et BC, og et B med et BD. Vi bliver dog nød til at flytte det hele en plads i arrayet, for at få et ekstra karakter ind, dette gøres ved at tælle bufferen op. Alt dette gør vi, da vi kun vil bruge A som start og slut bit, og derfor bliver nød til at erstatte det fra inputbufferen. Hvis vi får et 0 fra inputbufferen, er besked slut, og vi tilføjer derfor et A til bufferen. Vi sikre dog den ikke tilføjer uendelige A, ved at lave en Acount, som bliver brugt som et flag, der bliver sat højt, når vi har lavet den sidste bit om til et A. Hvis vi modtager andet end fra vores inputbuffer end A eller B, bliver det placeret i den nye buffer.

```
void Link::send(const char buf[], short size)
{
    int j = 0;
    buffer[0] = 'A';
    int Acount = 0;

    for(int i = 0; i < size-1; ++i)
    {
        ++j;
        if(buf[i] == 'A')
        {
            buffer[j] = 'B';
            ++j;
            buffer[j] = 'C';
        }
        else if(buf[i] == 'B')
        {
            buffer[j] = 'B';
            ++j;
            buffer[j] = 'D';
        }
        else if (buf[i] == 0 && Acount == 0)
        {
            buffer[j] = 'A';
            Acount = 1;
        }
        else
            buffer[j] = buf[i];
    }
    v24Write (serialPort, (unsigned char *)buffer, strlen(buffer));
}
```



**TRANSPORT::SEND**

**TRANSPORT::RECIEVE**

**APPLIKATIONLAGET::CLIENT**

**APPLIKATION::SERVER**

## SAMLET

## Bibliography