

Emnekode: PG3401

Emnenavn: Programmering i C for Linux

Innleveringsdato: 10.05.2024

Kandidatnr: **

Programmering i C for Linux:

Oppgave 1.

a)

Programmeringsspråket C regnes i nyere tid som i et lavnivåspråk. Dette betyr at C ligger nær maskinkode, kun ett steg over Assembly-språket, og tillater utviklere større kontroll over utførelsen av koden. Noe som gir utviklere direkte kontroll over komponenter på datamaskinen. Konseptet 'Spirit of C' (Maudal, 2017) bygger på at det skal være tillit til utvikleren, at de er bevisste i deres handlinger. C har derfor, i motsetning til flere høy-nivå språk, færre restriksjoner som kan hindre utførelsen av oppgaver. Dermed gir C-språket utviklere en større kontroll over handlinger som er mindre vanlig i høyere nivåspråk.

C ble opprinnelig designet for UNIX-operativsystemet, men grunnet sin fleksibilitet har C vist seg å være egnet for en rekke generelle formål. Evnen til å gi direkte tilgang til maskinwarefunksjoner, sammen med høy hastighet og ytelse, har gjort språket sentralt i utviklingen av innebygde systemer samt nettverksprogrammering. C har derfor blitt brukt i mange programmer opp gjennom tidene, inkludert i systemkomponenter for Windows. Det er verdt å nevne at C-kompilatoren selv er skrevet i C. I alt har C-programmeringsspråket bred rekkevidde og er et kraftig verktøy.

I tillegg til disse tradisjonelle bruksområdene er C også kjent for sin evne til å kjøre på ulike maskinwareplattformer uten å måtte endre koden, noe som understreker språkets portabilitet. C sin syntaks og konsepter har også påvirket andre programmeringsspråk, inkludert C++ og Java. I tillegg brukes C i utdanningen for å introdusere studenter til grunnleggende systemprogrammering og minnehåndtering.

Kildeliste 1.a

Maudal, O. (2017). History and spirit of C. NTNU.

https://www.pvv.ntnu.no/~oma/HistoryAndSpiritofC_ECC_Sep2017.pdf

Kernighan, B. W., & Ritchie, D. M. (1988). The C programming language (2nd ed.). Prentice Hall.

b)

Linus Torvalds er kjent i IT verden som skaperen av Linux. Utviklingen av Linux startet som et hobby-prosjekt for Linus, mens han fremdeles var student. Intensjonen var å skape et system som var basert på Unix, men tilpasset til akademisk bruk. Siden Linux ble gitt ut som et open-source prosjekt, har resultert i at hvem som helst kan bruke, endre og distribuere produktet, så lenge det forblir open-source. Dette har ført til en rekke forskjellige utgaver og at Linux nå er det mest brukte operativsystemet i verden, spesielt når det gjelder servere og superdatamaskiner (Store Norske Leksikon, 2024).

I tillegg til å skape Linux, har Linus også skapt Git. Git er et versjonkontrollsystem som har blitt standarden for å håndtere endringer i data filer og samarbeid mellom flere personer på disse filene. Git har blitt grunnleggende for utviklere, og er benyttet av alt fra startup selskap, store selskap og open-source prosjekter.

Linus har derfor blitt kjent for sin dedikasjon for å skape verktøy som kan benyttes av alle, og hans oppfordringer til samarbeid innen IT.

Kildeliste 1.b

Store Norske Leksikon. (2024, 4 Mai). Linux. Hentet fra <https://snl.no/Linux>

Store Norske Leksikon. (2024, 4 Mai). Linus Torvalds. Hentet fra https://snl.no/Linus_Torvalds

c)

Stegene fra kildekode til eksekverbar fil består av 5 trinn:

- Kildekode
- Preprocessor
- Kompilering
- Assembler
- Linker

For å kompilere et C program trenger en kildekode. Dette starter med å skrive kode i språket C til en fil som ender med '.c'. Denne koden beskriver hva programmet skal gjøre.

Neste steg er Preproccesing. Før kompilatoren starter tar preprosessoren kildekoden og håndterer direktivene, dette er filer som '#include' og '#define'. '#include' forteller preprossesoren å inkludere andre filer, og '#define' lager makroer, noe som kan erstatte deler av kildekoden. Dette klargjør kildekoden for kompilering.

Kompileringen av C koden, som nå er preprossesert, omformer koden til assembly språk. Dette er et lav-nivå språk som ligner mer på maskin-språk, men er fremdeles leselig av mennesker. Her blir koden optimalisert utifra regler som er definert i C.

Nå som det er dannet assembly kode, kan assembling prosessen begynne. Assembleren gjør om koden til maskinkode, i form av objektfiler. Disse filene inneholder binær kode, som en CPU kan kjøre, men er enda ikke et komplett program.

For å danne et komplett program, er vi nødt til å linke. Linkeren tar en eller flere objekt filer som har blitt laget av assembleren og kombinerer dem til en enkel eksekverbar fil. Her vil alle biblioteker som har blitt tatt i bruk også bli linket, og henter koden slik at det kan bli tatt i bruk. Resultatet er at vi får ut en eksekverbar fil, med all maskinkode som er nødvendig.

Kildeliste 1.c

GeeksforGeeks. (2024, 4 Mai). Compiling a C program: Behind the scenes. Hentet fra <https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/>

Oppgave 2 dokumentasjon.

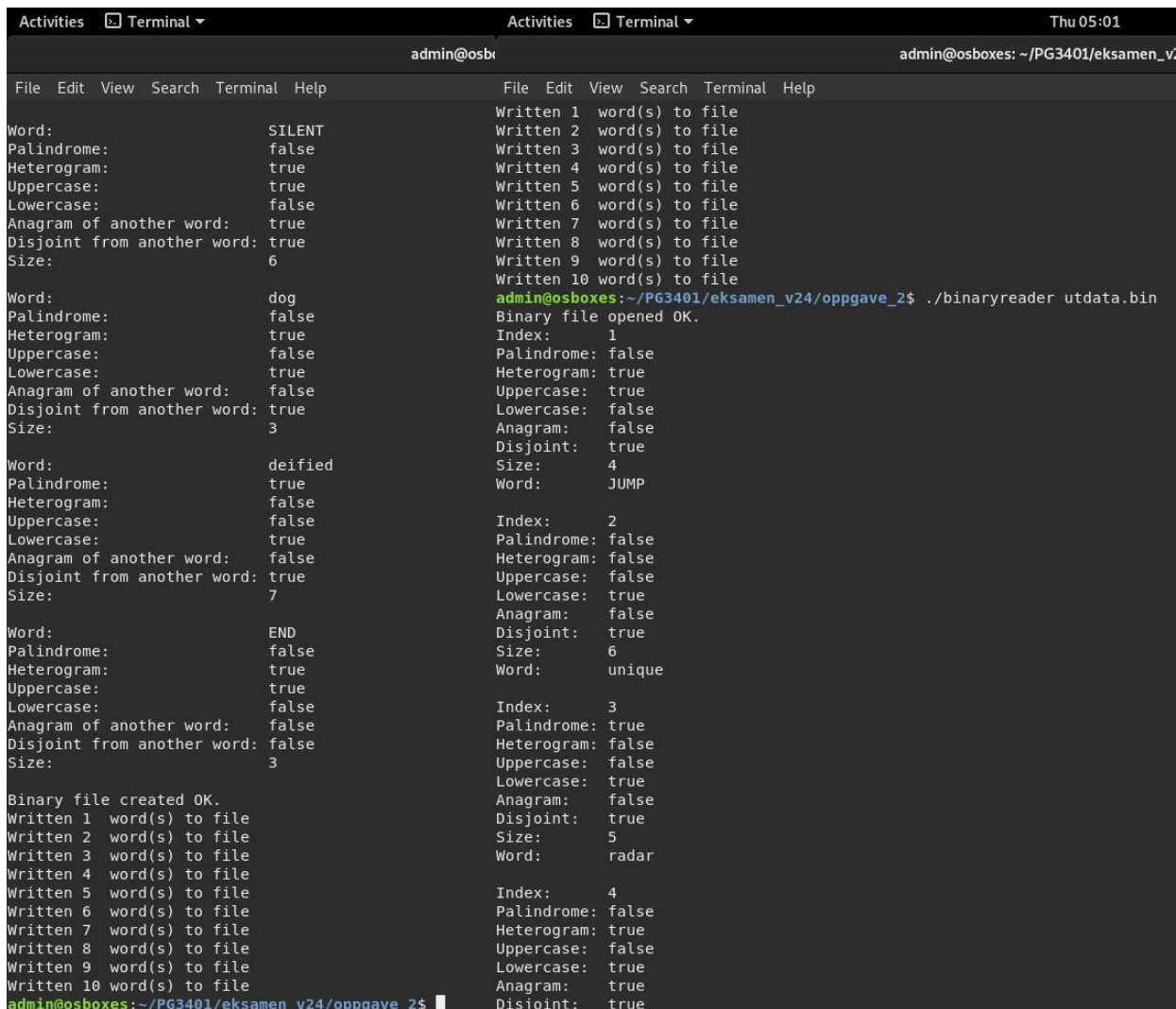
Bildet til venstre viser at programmet, ex2, kjører. Filen som blir behandlet er hardkodet i oppgaven, og den leser først ut alle verdiene, om det er et Palindrom m.m.

Deretter oppretter det en binær fil, og printer ut status på hvor mange ord den har skrevet over til den binære filen.

For å sikre meg at filen ble skrevet korrekt, og ikke levere denne oppgaven i blinde har jeg også opprettet en enkel kildefil som kan brukes til å lese utdata.bin filen, med dens metadata.

Denne kan kompiles med kommandoen 'gcc -o binaryreader binaryreader.c'

Deretter kan den kjøres med './binaryreader utdata.bin'. Resultatet er bukket tuk høyre.



```
admin@osbx
File Edit View Search Terminal Help
Word: SILENT
Palindrome: false
Heterogram: true
Uppercase: true
Lowercase: false
Anagram of another word: true
Disjoint from another word: true
Size: 6
Word: dog
Palindrome: false
Heterogram: true
Uppercase: false
Lowercase: true
Anagram of another word: false
Disjoint from another word: true
Size: 3
Word: deified
Palindrome: true
Heterogram: false
Uppercase: false
Lowercase: true
Anagram of another word: false
Disjoint from another word: true
Size: 7
Word: END
Palindrome: false
Heterogram: true
Uppercase: true
Lowercase: false
Anagram of another word: false
Disjoint from another word: false
Size: 3
Binary file created OK.
Written 1 word(s) to file
Written 2 word(s) to file
Written 3 word(s) to file
Written 4 word(s) to file
Written 5 word(s) to file
Written 6 word(s) to file
Written 7 word(s) to file
Written 8 word(s) to file
Written 9 word(s) to file
Written 10 word(s) to file
admin@osboxes:~/PG3401/eksamen_v24/oppgave_2$

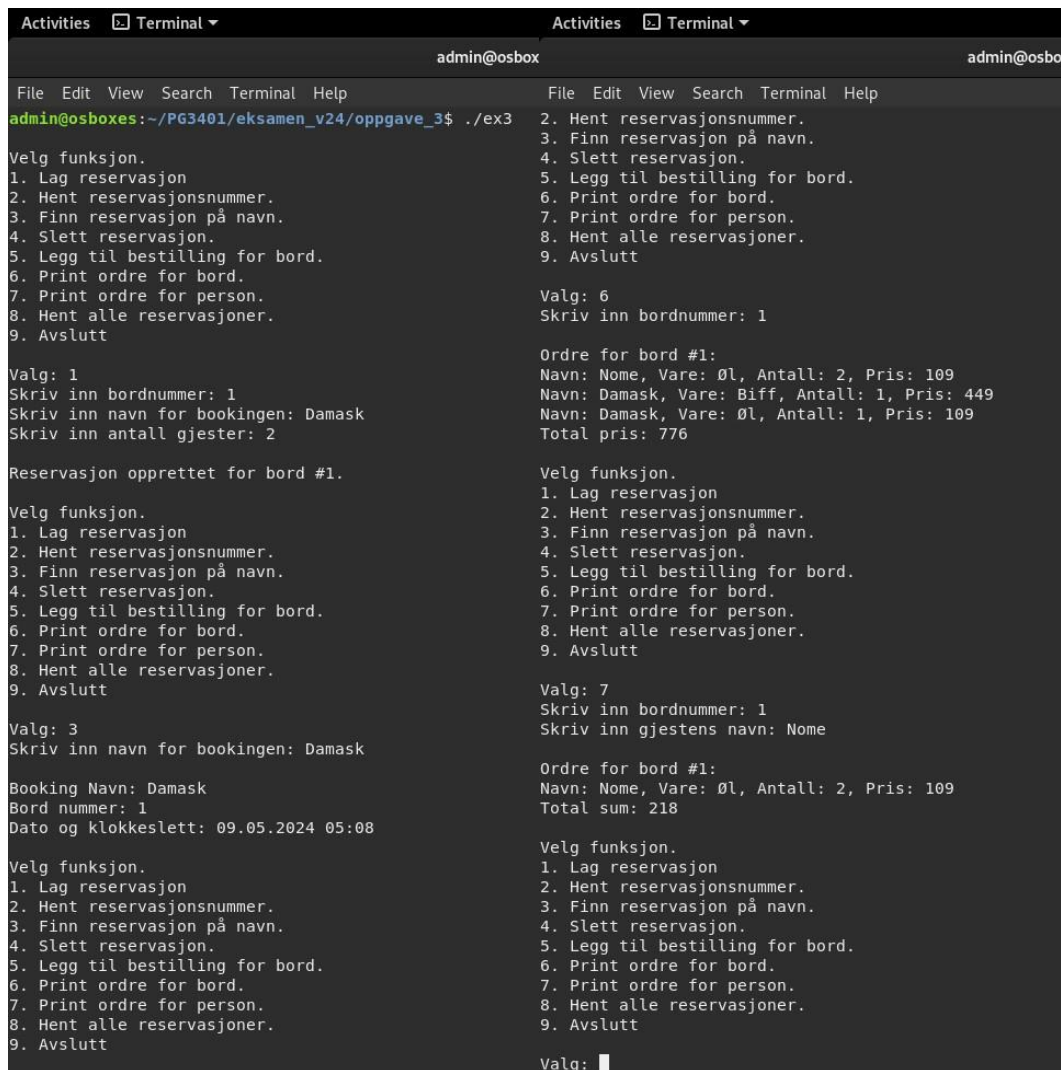
admin@osboxes: ~/PG3401/eksamen_v
File Edit View Search Terminal Help
Written 1 word(s) to file
Written 2 word(s) to file
Written 3 word(s) to file
Written 4 word(s) to file
Written 5 word(s) to file
Written 6 word(s) to file
Written 7 word(s) to file
Written 8 word(s) to file
Written 9 word(s) to file
Written 10 word(s) to file
admin@osboxes:~/PG3401/eksamen_v24/oppgave_2$ ./binaryreader utdata.bin
Binary file opened OK.
Index: 1
Palindrome: false
Heterogram: true
Uppercase: true
Lowercase: false
Anagram: false
Disjoint: true
Size: 4
Word: JUMP
Index: 2
Palindrome: false
Heterogram: false
Uppercase: false
Lowercase: true
Anagram: false
Disjoint: true
Size: 6
Word: unique
Index: 3
Palindrome: true
Heterogram: false
Uppercase: false
Lowercase: true
Anagram: false
Disjoint: true
Size: 5
Word: radar
Index: 4
Palindrome: false
Heterogram: true
Uppercase: false
Lowercase: true
Anagram: true
Disjoint: true
```

Oppgave 3 dokumentasjon.

Oppgave 3 vises kjørt på bildene under. For å ikke 'bloate' PDF besvarelsen med bilder, valgte jeg å inkludere 2 bilder som viste noen av de essensielle funksjonene til programmet.

I bildet til venstre blir en reservasjon opprettet (valg 1), deretter søker vi etter reservasjon på navnet (valg 3).

Bildet til høyre viser reservasjoner gjort på et bord (valg 6), og individuelle bestillinger for et bord (Separat regning, valg 7).



```
admin@osboxes:~/PG3401/eksamen_v24/oppgave_3$ ./ex3
Velg funksjon.
1. Lag reservasjon
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

Valg: 1
Skriv inn bordnummer: 1
Skriv inn navn for bookingen: Damask
Skriv inn antall gjester: 2

Reservasjon opprettet for bord #1.

Velg funksjon.
1. Lag reservasjon
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

Valg: 3
Skriv inn navn for bookingen: Damask

Booking Navn: Damask
Bord nummer: 1
Dato og klokkeslett: 09.05.2024 05:08

Velg funksjon.
1. Lag reservasjon
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

admin@osboxes:~/PG3401/eksamen_v24/oppgave_3$ ./ex3
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

Valg: 6
Skriv inn bordnummer: 1

Ordre for bord #1:
Navn: Nome, Vare: Øl, Antall: 2, Pris: 109
Navn: Damask, Vare: Biff, Antall: 1, Pris: 449
Navn: Damask, Vare: Øl, Antall: 1, Pris: 109
Total pris: 776

Velg funksjon.
1. Lag reservasjon
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

Valg: 7
Skriv inn bordnummer: 1
Skriv inn gjestens navn: Nome

Ordre for bord #1:
Navn: Nome, Vare: Øl, Antall: 2, Pris: 109
Total sum: 218

Velg funksjon.
1. Lag reservasjon
2. Hent reservasjonsnummer.
3. Finn reservasjon på navn.
4. Slett reservasjon.
5. Legg til bestilling for bord.
6. Print ordre for bord.
7. Print ordre for person.
8. Hent alle reservasjoner.
9. Avslutt

Valg: 
```

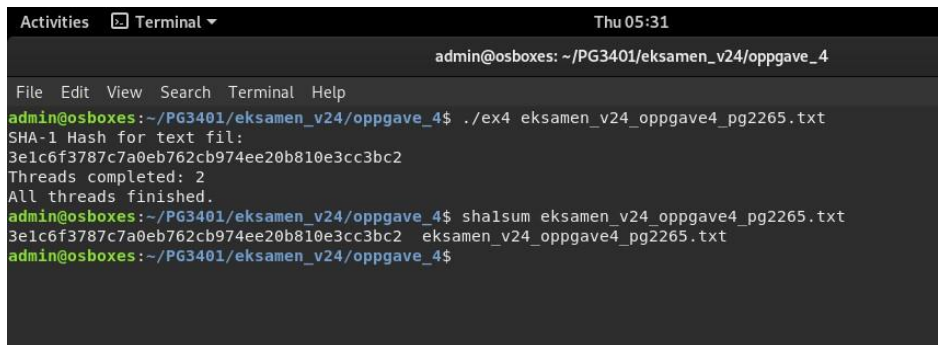
Oppgave 4 dokumentasjon.

Oppgave 4 oppretter 2 tråder, hvor tråd A skal lese en fil, og overføre denne til tråd B. Trådene har en minneplass på 4096 byte, og siden filen er større overføres data i flere omganger.

Det er også lagt inn en funksjon for å sjekke SHA1 Hash verdien til filen. Dette gjøres ved at tråd B regner ut verdien av dataene den har mottatt, printer dette til skjermen (som vist i bildet under, hvor programmet kjører), før den avslutter tråd B.

Dersom dette har skjedd uten feil, vil både SHA1 hash være printet til skjerm, og vi vil få opplyst hvor mange tråder som er fullført og om alle er ferdig.

For å sikre meg at trådene behandlet dataene riktig og regnet ut hash verdi korrekt, dobbeltsjekket jeg med kommandoen 'sha1sum' kjørt mot målfilen.



```
Activities  Terminal  Thu 05:31
admin@osboxes: ~/PG3401/eksamen_v24/oppgave_4

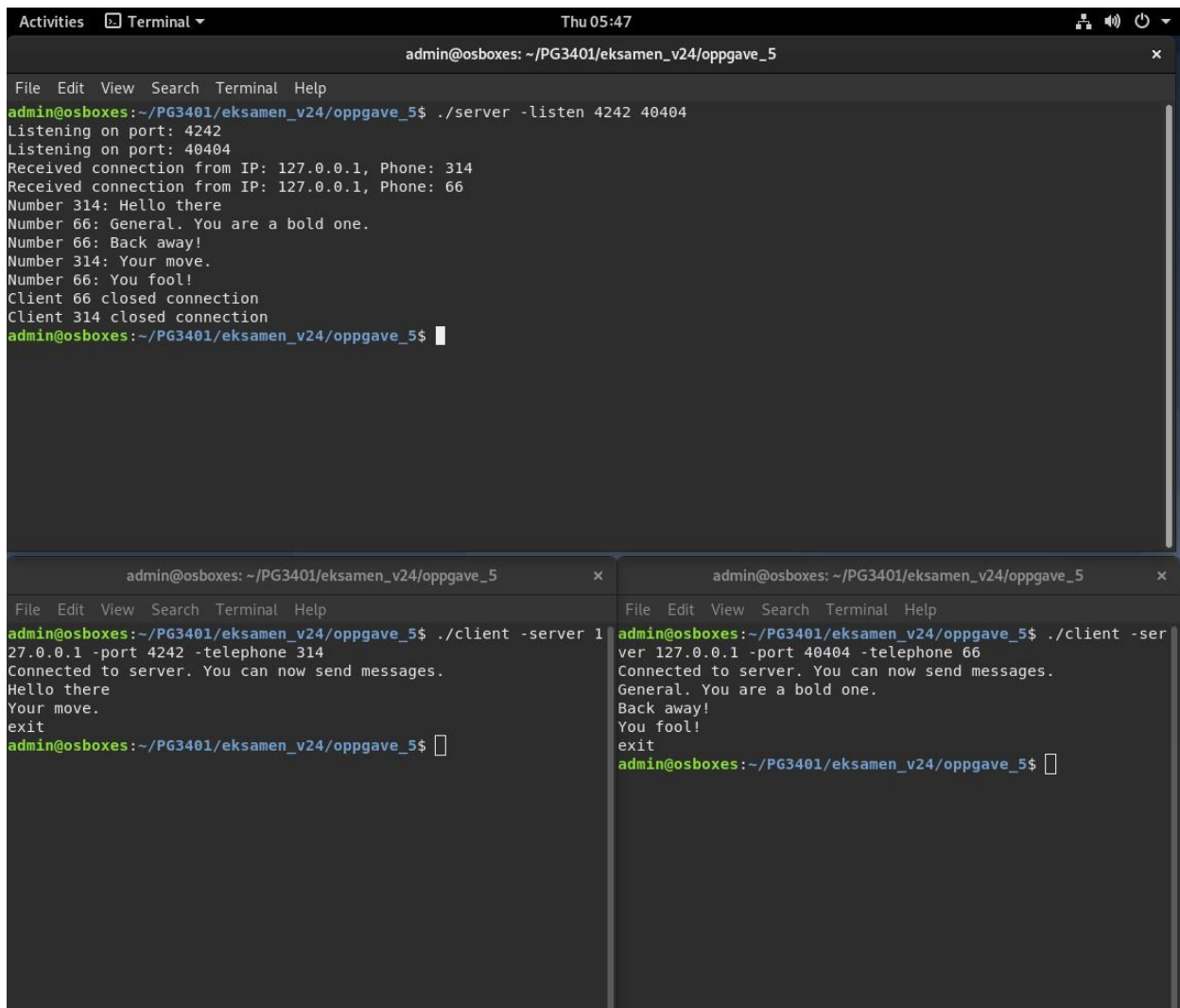
File Edit View Search Terminal Help
admin@osboxes:~/PG3401/eksamen_v24/oppgave_4$ ./ex4 eksamen_v24_oppgave4_pg2265.txt
SHA-1 Hash for text fil:
3e1c6f3787c7a0eb762cb974ee20b810e3cc3bc2
Threads completed: 2
All threads finished.
admin@osboxes:~/PG3401/eksamen_v24/oppgave_4$ sha1sum eksamen_v24_oppgave4_pg2265.txt
3e1c6f3787c7a0eb762cb974ee20b810e3cc3bc2  eksamen_v24_oppgave4_pg2265.txt
admin@osboxes:~/PG3401/eksamen_v24/oppgave_4$
```

Oppgave 5 dokumentasjon.

Jeg løste oppgaven med en separat server.c og en client.c kildefil. Disse kombileres begge med samme makefil.

For å kjøre dem, er syntaksen som vist nedenfor (dersom kjørt uten riktig antall parameter vil det komme feilmelding). For å avslutte programmet bruker en kommandoen 'exit'.

Programmet oppretter to tråder for å håndtere tilkobling på to ulike porter. Nå to klienter er koblet til, vil disse meldingene bli koblet sendt og mottatt på serveren. Dersom en klient prøver å koble til med feil Magic Number, vil de bli nektet tilgang.



The screenshot displays three terminal windows from a Linux environment. The top window shows the server program running, listening on ports 4242 and 40404, and handling two connections from IP 127.0.0.1. The bottom-left window shows a client connected to the server on port 4242, sending messages and then exiting. The bottom-right window shows another client connected to the server on port 40404, sending messages and then exiting.

```
admin@osboxes: ~/PG3401/eksamen_v24/oppgave_5
File Edit View Search Terminal Help
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$ ./server -listen 4242 40404
Listening on port: 4242
Listening on port: 40404
Received connection from IP: 127.0.0.1, Phone: 314
Received connection from IP: 127.0.0.1, Phone: 66
Number 314: Hello there
Number 66: General. You are a bold one.
Number 66: Back away!
Number 314: Your move.
Number 66: You fool!
Client 66 closed connection
Client 314 closed connection
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$

admin@osboxes: ~/PG3401/eksamen_v24/oppgave_5
File Edit View Search Terminal Help
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$ ./client -server 127.0.0.1 -port 4242 -telephone 314
Connected to server. You can now send messages.
Hello there
Your move.
exit
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$

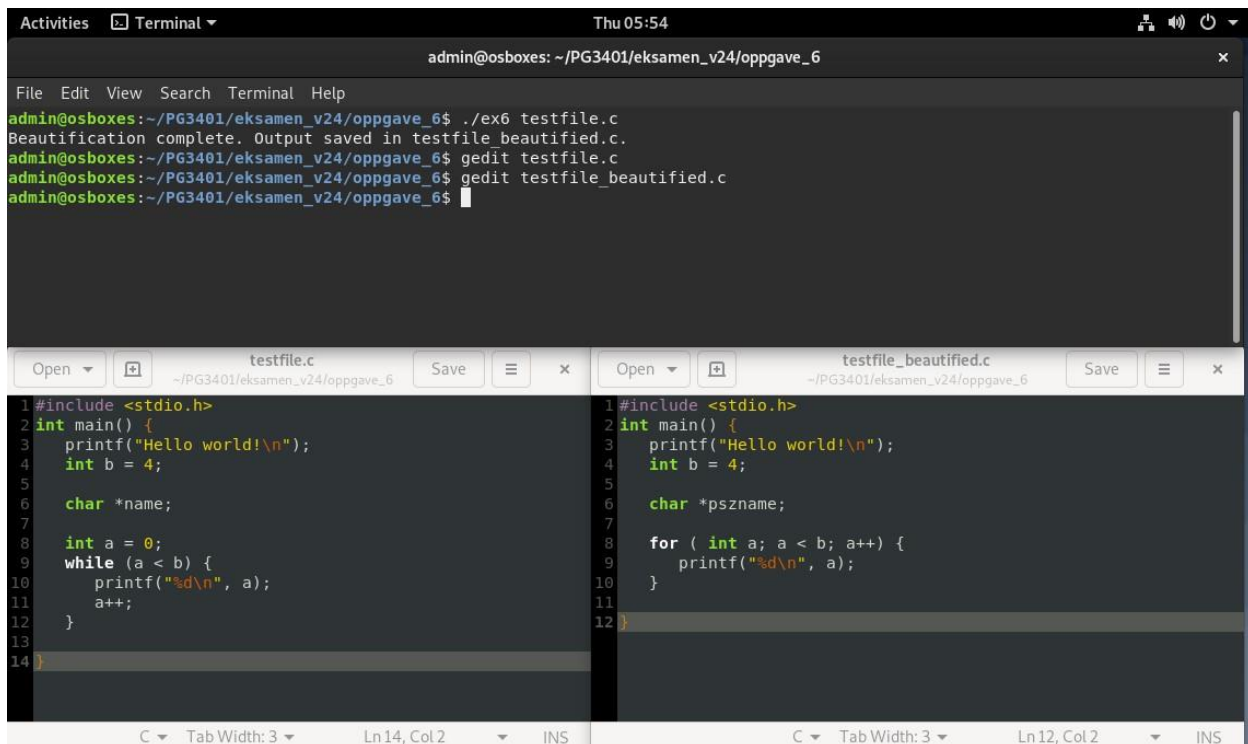
admin@osboxes: ~/PG3401/eksamen_v24/oppgave_5
File Edit View Search Terminal Help
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$ ./client -server 127.0.0.1 -port 40404 -telephone 66
Connected to server. You can now send messages.
General. You are a bold one.
Back away!
You fool!
exit
admin@osboxes:~/PG3401/eksamen_v24/oppgave_5$
```


Oppgave 6 dokumentasjon.

Filen `ex6.c` tar en C kildefil som parameter, og behandler input før den lagrer dette i en ny fil, med navn '`kildefil_beautified.c`'. Endringene programmet gjør, er:

- Finner indenteringer gjort med 3 konsekutive mellomrom, og endrer dette til en tabulator verdi.
- Finner variabler som er deklartert (her kun `Char*`) og sørger for at de er deklartert med Hungarian notation.
- Finner enkle while-løkker (Programmet sjekker om det er en enkel int deklarerings før løkken) og gjør om dette til en for-løkke, mens den beholder innholdet i løkken minus inkrementeringen som er flyttet til headeren av løkken.

Bildet nedenfor viser kjøring av programmet mot en '`testfile.c`', som også er vist i gedit til venstre, og at den blir lagret i den nye filen '`testfile_beautified.c`', som er vist i gedit til høyre.



The screenshot shows a terminal window at the top and two gedit windows below it. The terminal window, titled 'admin@osboxes: ~/PG3401/eksamen_v24/oppgave_6', shows the following commands and output:

```
admin@osboxes:~/PG3401/eksamen_v24/oppgave_6$ ./ex6 testfile.c
Beautification complete. Output saved in testfile beautified.c.
admin@osboxes:~/PG3401/eksamen_v24/oppgave_6$ gedit testfile.c
admin@osboxes:~/PG3401/eksamen_v24/oppgave_6$ gedit testfile_beautified.c
admin@osboxes:~/PG3401/eksamen_v24/oppgave_6$
```

The left gedit window shows the original `testfile.c` with the following code:

```
1#include <stdio.h>
2int main() {
3    printf("Hello world!\n");
4    int b = 4;
5
6    char *name;
7
8    int a = 0;
9    while (a < b) {
10        printf("%d\n", a);
11        a++;
12    }
13
14}
```

The right gedit window shows the beautified `testfile_beautified.c` with the following code:

```
1#include <stdio.h>
2int main() {
3    printf("Hello world!\n");
4    int b = 4;
5
6    char *pszname;
7
8    for (int a; a < b; a++) {
9        printf("%d\n", a);
10    }
11
12}
```

Slutt på besvarelse