PG3401

Programmering i C for Linux

11.05.2023

Skriftlig individuell hjemmeeksamen

Høyskolen Kristiania

Vår 2023

Oppgave 1. Teori

a) Programmeringsspråket C er et generelt formål språk. Det betyr at det kan i utgangspunkt benyttes til alt, men i mer moderne tid så er språket landet på applikasjoner som krever meget høy ytelse som OS, drivere og protokoller. C benyttes på all muligens data arkitekturer fra embedded systemer til supermaskiner(“C (Programming Language)” 2023, para. 1)

Språket ble utviklet hos Bell Labs og skulle være en etterfølger av språket B.
Dennis Ritchie var mannen som startet utviklingen av det rundt 1972 og 1973.
Raskt begynte C å få stor popularitet, men på denne tiden var det mangel på en del standarder.
Språket ble mer samlet når ANSI C først kom i 1989(“C (Programming Language)” 2023, para. 2).
I senere tid har det kommet flere større samlinger som; C99, C11 og C17. Disse korrigerer eller setter nye standarder på hvordan man skal kunne skrive C.

Når C ble laget var hovedsaklig det for å kunne skrive ny kode inn i Unix kjernen på en mer effektiv måte uten å skrive assembly. Hvordan C gjør dette er å gå igjennom en kompiler som endrer C fra det lettleste koden mennesker kan lese og igjennom flere prosesser og ender som maskinkode. I en av disse prosessene så får man faktisk assembly kode tilgjengelig.

Siden C er kompilert ned til maskinkode gjør det at man får kode rett på metallet, som mange kaller det. Dette gir en egenskap at C programmer er ekstremt effektivt i forhold til mange moderne språk som ofte kjører igjennom en eller annen form for VM til å utføre koden. Det gjør at C sitt bruksområde har gått i retning til høy krevende og effektiv programmer som eks. OS og drivere. Det gjør også C som en stor kandidat til plattformer som embedded med få resurser eller supermaskiner som skal behandle enorme mengder data.

Egenskapene som gjør dette mulig er at C tillater utvikleren til å gjøre og styre alt. C gjør dette med eksempelvis å la utvikleren å direkte tilgang til minne adresser igjennom pointere, definere egne minne samlinger med struct eller la flere data typer benytte samme minne lokasjon med union.

I moderne tid så ses C som et språk som tas lang tid å utvikle programvare med, siden alt må håndteres selv av utvikleren, men utbytte er ekstremt lettvekt og raskt kjørt programmer.

b) Dennis MacAlistair Ritchie (September 9, 1941 – October 12, 2011)(“Dennis Ritchie” 2023, para. 1) var en Amrikansk informatiker som er mest kjent for å lage programmeringsspråket C og sammen med sin kollega Unix OS. Igjennom årene har han vunnet samtlige priser som; Turing Award i 1983, Hamming Medal i 1990 og National Medal of Technology i 2007(“Dennis Ritchie” 2023, para. 1).

Ritchie har en lang historie i andre halvdel av 1900-tallet for å bygge grunnsteinene for mye av dataverden benytter den dag i dag. Som datahistoriker Paul E. Ceruzzi sa:
«Ritchie was under the radar. His name was not a household name at all, but... if you had a microscope and could look in a computer, you'd see his work everywhere inside.» (“Dennis Ritchie” 2023, pt. Death)

Med å utvikle programmeringsspråket C og bygge Unix OS som videre har fått derivater som Linux, BSD og MacOS har store betydninger for mange den dag gi dag. C har en lang historie og har derivat språk som C++, Objectiv-C brukt av Apple, C# brukt av Microsoft og Java brukt stort av av bedrifter har stor inspirasjon fra C som brukes globalt over hele verden. Alt dette benyttes ofte i mange kritiske systemer den dag gi dag kan du begrunne at mye av jobben til Ritchie har vært samfunnskritisk og viktig for teknologisk utvikling.

c) Sudo er en terminal kommando for de fleste Unix-like systemer som eleverer en brukers rettigheter, som oftest super bruker. Sudo er originalt forkortet til «superuser do», men i dag i dokumentasjonen står det for «substitute user, do»(“Sudo” 2023, para. 1). Grunnen for endring av navn er at sudo kan kjøre kommandoer som andre brukere.

Sudo benyttes før en annen kommando som eks: «sudo apt install <app-name>». Etter denne kommandoen må du skrive inn brukeren sin passord for å bekrefte at en skal kjøre apt-install kommandoen som eksempelvis super bruker.

En annen kommando som er meget lik «sudo», er «su». Som gjør at man endrer brukeren til ei annen midlertidig. Eksempelvis kan man skrive «su root» etterfølgt av root passord. Med å gjøre dette kan man gjør terminalkommandoer som root til man enten logger av eller terminalen lukkes.

Oppgave 2. Filhåndtering og funksjoner

Tolkning av oppgaven:
Her skal det benyttes samtlige .c filer i sammenheng med å analysere tall som er vedlagt eksamens oppgaven. Resultatet skal lagres som en bool i en struct som holder all dataen samlet på et sted. All metadataen skal så lagres i en fil. Her ble jeg usikker i oppgave teksten om det er kun tallet som skal analyseres som printes binært i filen (med 1 og 0) og resten som ascii tekst eller alt skal lagres som en binær fil. Løsningen ble at programmet gjør begge type filer som et resultat når den kjøres.

Framgangsmåte:
Lastet ned alle .c filene og laget .h filer for alle .c filerne. Begrunnelsen for dette er at jeg ønsker å behandle hver .c fil som et eget bibliotek samt i C språket skal alle .c filer snakke med hverandre igjennom .h filer.

Lagde en egen .c fil som gjør alt av analyse og lagring til fil, hvor main.c kun kaller på en start funksjon til denne filen. I funksjonen begynte jeg sakte å skrive en hoved logikk som basere seg på en loop som kjører så lenge det er flere char's i filen vi leser fra.

Har benyttet mindre kommentarer for å forklare mindre åpenbare detaljer og prøvd å benytte lange forklarende navn for å gjøre koden mer selvforklarende.
Eksempel på dette er funksjonnavn: writeResultToFileAsciiText() og writeResultToFilePureBinary().
I .h filen skriver jeg lengre dokumenterende kommentarer til funksjonene. Disse går fort igjennom hva dem gjør, parametere og eventuelt retur verdier.

En viktig ting er å huske å lukke fil-leserne side er teknisk sett begrenset system resurser må gis tilbake manuelt i programmet. Samt standard rutine som å frigjøre resurser som hentes igjennom malloc og calloc.

Siste detaljen er at hoved funksjonen returnere errno verdien og begrunnelsen for det er at hvis noe skal gå galt med fil operasjonene så lagres feilen i en global variabel navngitt «errno».
Errno har en standard verdi på 0 som tilsvarer programmet kjørte som det skal. Hvis programmet feilet med å gjøre noe så vil den ha en retur verdi som har en beskrivende feilmelding("Errno(3) - Linux Manual Page" n.d.).

Oppgave 3. Liste håndtering

Tolking av oppgaven:
I denne oppgaven skal man benytte en dobbel linklist som er lagrer flyvninger og i hver flyvning har en egen enkel linklist som lagrer passasjer. Med samtlige funksjonaliteter for å lage, slette, endre og vise data.

Fremgangsmåte:
Startet med å lage structs som inneholder ønsket data oppgaven oppgir. I denne prosessen lages det egne funksjoner som skal lage og frigjøre structene på en trygg måte. Når basisen for hele programmet er gjort ble det laget funksjoner som kan lagre data i linklistene på forskjellige nivåer. Med å kunne lagre data startet man med å kunne fjerne data. Når disse funksjonene var på plass ble det gjort manual testing for å sjekke at ting blir lagret og slettet på ønsket hvis.

Når bakgrunn funksjonene var på plass ble menyen og meny valg laget. Her desinfisering inputen inputen og gir deg i noen tilfeller å prøve på nytt eller tar deg tilbake til hovedmenyen.
Siste delen av oppgaven ble parallell implementering av endring og visning av data og menyfunksjoner.

Gjennom hele prosessen av dette ble valgrind aktivt benyttet for å finne minne lekkasjer. Alt fra spesifiserte free funksjoner og meny valg funksjonene som desinfisering input og benytter calloc.

Oppgave 4. Tråder

Tolking av oppgaven:
Det skal lages en applikasjon som har en hoved tråd (alle applikasjoner har en) som starter to arbeids tråder. Første arbeids tråd skal lese en en og en karakter fra dette tekstdokument til et delt minne på 4096 byte. Når dette er fult opp skal den andre arbeids tråden varsles for å få lov til å jobbe og selv gå i pause. Den andre tråden skal nå lese fra dette delte minne og starte en analyse av hvor mange ASCII utskrivbare tegn og antall ord av spesifisert av oppgaven. Når tråden er ferdig med å tømme det delte minne varsler den tråd en om at den skal få jobbe igjen og går selv inn i vente modus.

Fremgangsmåte:
Benyttet den gitte start .c filen. Gjør variabler lokalt til funksjonene og deler den med trådene igjennom en felles struct. Rette på bugs som gjør at trådene ikke avslutter på riktig måte. Endre hva som analyseres og til slutt implementere ord telling.

Min definisjon på å telle ord er som følger:
Leter etter første bokstav som matcher, så sjekkes index før om det er alt annet en A-Z og a-z. Hvis det stemmer skal den sjekke om index etter ordet er ferdig ender på alt annet enn A-Z og a-z.
Så sjekkes om hvor indexen bokstavene matchet faktisk er orde vi leter etter.
Det ordet som ble søkt skiller mellom store og små bokstaver.

Eksempler på hvor søke ordet telles som ord:
Ord <ord> ord…
Ord-<ord> ord…
Ord. <Ord>-ord…

Eksempler på hvor søke ordet ikke telles som ord:
Or<ord>d
Ord<ord>
<Ord>ord

Oppgave 5. Nettverk

Tolking av oppgaven:
Det skal lages et program som har muligheten til å starte opp i både server eller klient modus. Når programmet kjører i server modus skal det ta imot eksterne meldinger som skal kjøres som linux terminal kommandoer. Når kommandoen er kjørt skal tilbakemeldingen fra terminalen sendes til klienten som først sendte inn kommandoen.

Klienten skal koble seg til serveren når den har startet. Når den har koblet til skal brukeren kunne fritt sende linux terminal kommandoer til serveren og forvente et svar fra«terminalen» igjennom serveren.

På både server og klient skal man kunne initiere SIGINT (ctrl + c) som ikke skal krasje programmet, men på kontrollert måte avslutte programmet og samtidig melde i fra til motpart at dem også skal stenge av på en kontrollert måte.

Fremgangsmåte:
Det første som ble løst var å kunne få klient og server å ha muligheten til å lytte etter en terminering og samtidig sende meldinger på vanlig vis. Løsningen ble å kjøre to tråder, hvor en aktivt lytter hele tiden etter terminering og andre benyttes for vanlig kommunikasjon.

Vanlig kommunikasjon ble satt på vent mens implementeringen av kontrollert nedtegning på først implementert. Med å benytte en eldre funksjon som heter «signal(para1, para2)» så kan man endre oppførselen av et system signal. Her ble SIGINT benyttet og egen funksjon som skal håndtere dette kallet. I sammenheng med dette ble en funksjon med statiske variabler laget for å huske pointeren til en struct som definere om programmet kjører eller skal avslutte.
Grunnen for dette er fordi SIGINT handleren kan ikke ta in parametere, så for å jobbe rundt dette lage en funksjon som blir primet før trådene starter og kan senere bli kalt på av handleren for å slå ned programmet.

Etter terminering funksjonene var implementert ble kommunikasjonen lagt til.
For å få programmet til å kunne aktivt sjekke om den skal kjøre eller slå av så har jeg benyttet flagg for recv() funksjonen at den skal ikke fungere som et stoppe punkt i programmet. Mest sannsynlig gjør dette at klient og server går ut av synk når dem sender meldinger i noen tilfeller som gjør den henger seg opp.

Oppgave 6. Filhåndtering og tekst-parsing

Tolking av oppgaven:
Det skal lages en applikasjon som leser inn .c fil med en spesifikk skrive stil som skal «beautify» koden. Første tingen som skal gjøres penere er while loops til for loops. Andre er å gjøre unsigned int variabler uten Hungarian notation til å ha Hungarian notaiton. Siste er å gjøre tree spaces til å ha være en horisontal tab.

Fremgangsmåte:
Startet med å vurdere om jeg skal gjøre regex eller manual håndtering av kode med å sammenligne andre strenger. Landet på det siste fordi regex returnere bare true eller false hvis noe matcher og trolig lager bare et ekstra lag med kompleksitet.

Med å lese hele filen for å så konvertere det til en lang string som jeg kan endre på og «kopiere» over til en ny med endringer ble prinsippet. Løste C, B og A i denne rekkefølgen.

Kilder:

"C (Programming Language)." 2023. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=C_(programming_language)&oldid=1151920941.

"Dennis Ritchie." 2023. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Dennis_Ritchie&oldid=1152157761.

"Errno(3) - Linux Manual Page." n.d. Accessed May 1, 2023. https://man7.org/linux/man-pages/man3/errno.3.html.

"Sudo." 2023. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Sudo&oldid=1149397607.