PG3401

Programmering i C for linux

Hjemmeeksamen 14-dager

Høyskolen Kristiania
HØST 2021
k.nr. 2040

«Denne oppgaven er gjennomført som en del av utdannelsen ved Høyskolen Kristiania. Høyskolen er ikke ansvarlig for oppgavens metoder, resultater, konklusjoner eller anbefalinger.»

Oppgave 1. Generelt (5 %)

a)

C er et programmeringsspråk som er prosedyremessig programmeringsspråk, som først ble laget med tanke på koding av oppprativsustem. Syntaksen i c er ryddig, og systematisk. I C har du god tilgang til hvordan de forskjellige minneressursene er brukt og kan addressere minne gjennom pekere. Dette gir mye kontroll. Men også en mulighet for misbruk. Derfor er det viktig å bruke ting som valgrind for å kontrollere om jeg har klart å deallokere det minnet programmet har brukt.

C gir også veldig god kontroll på hvor mye minne som brukes. Så ved programmering av embedded systems og andre systemer med begrenset minne, vil C være et veldig bra valg av programmeringsspråk. Det er også endel libraries til C som gjør at mulighetene er veldig gode for at noen har bygget funksjonene du ønsker på et tidligere tidspunkt. C er også et veldig raskt programmeringssråk hvor pekere er en stor og nyttig del. Siden du istedenfor å flytte noe i minne heller kan endre pekeren til variabelen til addressen der den nye variabelen du ønsker ligger lagret. Dette klarer C uten at du må skrive inn minneaddressene men kan aksessere de gjennom enkel kode. Ved å skrive a = &b kan du endre a til å peke til b og programmet slipper å flytte dataen i minnet. C er også portabelt, dvs at det kan kompileres på forskjellige systemer. Dermed kan det lages programmer for flere system uten å tenke noe spesielt over det i utviklingen.

b)

Linus Torvalds er en finsk utvikler som er mest kjent for å være linux sin far. Han er hovedutvikleren av linux kernelen og har også vært med på å skrive mange andre nyttige verktøy som f.eks. Git. Han er nå fortsatt den som har øverste autoritet på endringer i linux kernelen. Han er også en stor forkjemper for open source programmvare. Dette var grunnen til at han startet med git, at versjonskontrollsoftwaren som ble brukt til linux før var BitKeeper. Linus ble kritisert for bruken. Så da var løsningen å utvikle sin egen. Ettersom linux har blitt viderutviklet av mange forskjellige utviklere gjennom tidene har han ikke lenger skrevet «mesteparten» av kernelen. Men nå har har en mindre del. Han er derimot en av de som har skrevet mest og har stor påvirkningskraft i miljøet.

c)

Jeg ville navigert til mappen som innholder filen, via terminal med kommandoene «ls» for å vise hvilke mapper og filer som er i den mappen jeg var i.
Jeg ville brukt cd for å navigere til riktig sted.
cd ./mappenavn og cd .. for å gå til mappen over .
når jeg kom fram til riktig mappe som inneholdt filen ville jeg skrevet
ls -la
for å se hvike rettigheter filen hadde. Det viser da hvilke rettigheten brukeren har for filen. R betyr read, w beyr write. Og x som betyr execute.
Filen vil ha rettigheter som f.eks denne:
--rw-rw-r-- 1 user1 user1 22520 Nov 10 10:09 SOURCE
jeg ville så kjørt kommandoen
chmod u+x ./SOURCE
for å endre rettighetene til user. U = user, a = all, g = group, o = other. Hvor du kan sette rettighentene til spesifikke brukere. Ved å skrive +x så legger vi til execute rettigheter til filen. Vi kan også fjerne rettigheter med – etterfulgt av den rettigheten vi ønsker å endre. f.eks fjerne execute rettigheten til user igjen for filen over ved å skrive
chmod u-x ./SOURCE evt kan du bruke chmod +777 for å gi alle alle rettigheter. Evt +700 så får brukeren som skriver kommandoen alle rettigheter til filen.

Oppgave 2.

Fungerer som forventet, har to funksjoner, main og letteroccurencecounter.
Mainfunksjonen tar inn enfil som innholder den hexadecimale teksten som oppgaven ønsker at vi skulle konvertere. Den laster også inn en fil hvor vi kan skrive ut hva vi har kommet fram til etter vi har konvertert hexadecimalene til ascii.
For å konvertere fra hex til ascii må vi først lese inn filen. Vi bruker så strncat for å legge sammen to karakterer til et nummer da fgetc bare leser inn en og en character. Vi tar så de to neste characterene og bruker strtol for å få konvertert tallet fra Hex (base16) til base 10. Vi bruker så fputc med det konverterte int til karakter med fputc. Da karakterer er lagret som int i c så fungerer dette bra. Vi printer så ut det samme i terminalen.

Før vi lukker fptrOut- Så sender vi den til letterOccurenceCounter som tar en filpointer i argument. For å løse tellingen leser vi inn en og en karakter, før vi kjører en «toupper» funksjon på den som omgjør alle bokstavene til store bokstaver. Dette gjør vi sånn at vi får alt i samme range- mellom 0-26. Vi sjekker så om de er innenfor og hvis de er ikrementerer vi en posisjon i en array som korresponderer med 0-26 og A-Z. For å skrive det ut i terminal kjører vi bare en printf hvor vi bruker en for loop med inkrementasjoner på to. Der den printer bokstaven og det korresponderende arrayposisjonen.

Når den er ferdig med tellingen og å skrive det ut til terminal lukker vi filene og returner med 0; Siden programmet og trenger ikke frigi noe da vi ikke har manuelt tildelt noe minne.

Oppgave 3

Her har jeg valgt å lage 2 structer, hvor den ene lagrer en DLL -Doubly linked list. Denne har bare en head og en tail som data lagret. Dette er pointere til Noder som er på starten og slutten av listen.

Vi har funksjoner som gjør det meste, en menu som kjører menyen.
En egen funksjon både for å opprette og teste om mallocen fungerer for både listen og hver node. Dette sparer oss for noe kode etterhvert som vi oppretter flere noder. Og gjør det mulig å viderutvikle med funksjoner hvor programmet legger til mer enn en liste. Disse funksjonene returnerer pointere til structene med allokeert memory.

Programmet har også en del andre funksjoner slik som oppgaven ber oss om å opprette. Jeg har valgt å lage mange funksjoner da det er lettere å endre moduler ved et senere tidspunkt.

Jeg sleit en stund med memory leaks og forskjellige non conditional jumps ved sletting av noder. Dermed valgte jeg å opprette en egen funksjon som tok seg av slettingen. Dette gjorde at jeg bare måtte feilsøke den ene funksjonen når noe ikke fungerte med slettingen. Og gjorde koden mer oversiktlig.

Jeg valgte også å lage en egen funksjon for inputvaldation. Men kan endre den hvis jeg ønsket å implementere en funksjon for å f.eks legge til i starten av listen ved et senere tidspunkt.

Oppgave 4.

Jeg startet denne oppgaven ved å prøve å kompilere koden som assembly uten noe hell. Jeg prøvde så med -g flagget i GCC. Ved å legge til det så vil kompilatoren legge ved flagg til hvilken linje det er feil på når koden krasjer. Jeg kjørte så valgruind med flaggene -d, -v, --leak-check=full og --track-origins=yes På denne måten fant jeg ut at det var en feil på linjen

```
strchr(pszPtr, '\n')[0] = 0;
```

her prøver programmet å nullterminere strengen direkte ved bruk av pointer. I c fører det til av programmet prøver å accessere og endre noe programmet ikke har lov til å endre. Dett er da strengen som blir lastet inn i funkjsonen regnes som constant. Jeg valgte å skrive om dette til en if/else funksjon hvor jeg først brukte en while som telte antall karakterer framover til Newline karakteren. Jeg bruker så dette tallet for å kopiere og nullterminere strengen ved hjelp av strncat som tar et argument for hvor mye som skal kopieres. Strchr har også ingen beskyttelse om ikke den finner karakteren. Og vil bare fortsette å lese i minne forbi det som er til programmet og fører itl krasj. Programmet nullterminerer så med hjelp av tellevariabelen.

Etter at jeg hadde funnet denne feilen og endret på den gikk jeg videre på å kjøre koden. Og klarte nå å kjøre den jeg fant da ut at `if (pHttp->iHttpCode = 200) {`
hadde et enkelt erliktegn og ikke dobbelt. Dette gjør at den setter variablen til 200 og alltid er true. Dette er enkelt å fikse ved å bare legge inn et dobbelt erliktegn.

Den siste feilen som også sto beskrevet i oppgaven var at content-length ikke fungerte som den skulle. Dette var fordi argumentet som ble passet inn ikke var pointeren til hvor i headeren «content lenghth» sto men hele headeren. Jeg endret det fra `pHttp->iContentLength = atoi(pszHttp);`
til å ta pointeren inn og gjøre det om til int med atoi. Da fungerte content length også.

Oppgaven spurte også etter å legge til en funksjon for å parse infoen til «Last-Modified». Jeg brukte da samme metode som parsingen av de andre verdiene fra headeren for å søke etter hvor jeg fant den teksen og oppdatere pekeren til å være på startten av verdien etter last modified. programmet tar så og leter etter newlinecharacter og bruker den for å kopiere så mye progrgammet trenger.

Oppgave 5

I oppgave 5 har jeg lagd en main metode som oppretter en ny tråd. Denne tråden kjører en egen funksjon som jeg har valgt å kalle worrkThread(), denne tar inn et argument som er en pointer til en struct som innehodler en buffer på 11 tegn. Denne structen innehodler bufferen og i og med at tråder deler minne kan programmet få tilgang til det fegts tar i input loopen i main med bruke mutexes. Disse trengs slik at trådene ikke accesserer det samme minne samtidig, noe som fort ville fått programmet til å krasje.
For å synkronisere trådene bruker progra,,et to signaler. Dette valgte jeg framfor semaphores da jeg syntes det var lettere å holde oversikten med to variabler over 1. Jeg bruker en wait i main som venter på signal fra workthread om å kjøre og sende mer data. Om jeg har forstått hvordan fgets fungerer i c og terminal riktig. Så blir alt over antall tegn lest av fgets bufferet og sendt senere som en ny fgets. Dermed går loopen automatisk. Den vil også være 0teminert automatisk og kopierer int -1. Så til bufferen når koden sier

```
if (fgets(structBuffer->input, 11, stdin))
```

blir 10 tegn kopiert og ’\0’ på slutten. maintråden venter så på at worktåden skal skrive inn i dokumentet og sende nytt signal om at main kan kjøre igjen.

Etter at det er lagret så sender maintråden et ekstra signal for å være sikker på at tråd no2 escaper whileloopen. Programmet har så en pthread join som venter på at workthread skal bli ferdig fø rden fortsetter med oppryddingen av programmet.

Programmet sørger for at mutexen er opplåst før programmet sletter den med mutex destroy og addressen. Programmet sletter så begge conditionene og går frigjør minnet til structBufferen. Programmet returnerer så 0 og er ferdig med å kjøre programmet.

Om quit er det første ordet avslutter programmet selvom det er en litt lenger streng som blir puttet inn.

Oppgave 6

Denne oppgaven gikk ut på å oprette en connection til en server og sende en get request for å få svar fra en server. Jeg startet først med å opprette en enkel funksjon hvor alt var hardkodet, fra servernavn til get requesten. Jeg sjekker også underveis om det er noen av argumentene som ikke stemmer. Hvis det er noen av funksjonene som ikke gir korrekt ut, så sender jeg en melding til en funksjon som skriver ut srgumentet til pritf. Og også avslutter programmet med kode 1. Programmet gir også ut statusmeldinger til terminalen etter hvert som de forskjellige socketfunksjoenene har kjørt. På den måten blir feilsøking lettere.

Jeg lagde først hele koden med hardkodet ip, før jeg forsøkte meg på en annen løsning med gethostbyname(). Jeg sleit litt i strarten men når dette fungerte så da valgte jeg å levere den versjonen. Jeg hadde noe problemer med get requesten min i starten. Men fant etterhver ut at det var grunnet mangel på host. Jeg slet også med at programmet holdt socketen åpen etter at den hadde mottatt websiden og gjorde «nettleseren» min treg. Jeg fant ut at dette var pga. Standard connection type var keep alive. Ved å endre requesten til å inkludere `Connection: close` ble det problemet løst.

I de første versjonene av programmet allokerte jeg bare minne til bufferen uten å memsette den til 0 det gjorde at jeg fikk noen rare responses. `memset(recvBuffer, 0, MAXLENGTH);` Dette gjorde at det problemet ble løst. I og med at jeg mallocer memory til ip i getipAddr funksjonen min må jeg free() det mot slutten av main. `close(socketConnection);` Denne lukker sockettilkoblingen.