Eksamen PG3401 – H21

Oppgave 1: Generelt

a) Forklar hva C programmeringsspråket kan brukes til.

C kan brukes til å skrive programmer som kan kjøre både på windows, mac og linux.
C brukes til å skrive lavnivå kode, enklere enn å skrive assembly og kan aksisere minnet direkte. C er kanskje det eneste språket som har en kompilator skrevet med samme språk som den skal kompilere. C kan brukes til det meste, embedded systems, enterprice applikasjoner, spill, grafikk, utregninger og kalkulering av store mengder data osv.

b) Hvem er Linus Torvalds og hva er han kjent for innen Informasjonsteknologi?

Linus Torvalds er en finsk data ingeniør og open source forkjemper, mest kjent for å bli sett på mannen bak Linux kernelen. Han utviklet også versjonkontrollsystemet Git.

c) Hvis du laster ned en binær eksekverbar fil fra internett kan det være at filen ikke kan kjøres fordi den ikke har «execute» rettigheter satt. Forklar hvordan du kan løse dette på kommandolinje i Linux.

Dersom du, mens du befinner deg i samme mappe som filen, kan du skrive kommandoen "ls -l", dette vil liste opp alle filer (som ikke starer med ".") i mappen. Lengst til venstre kan man hvem som har hvilke rettigheter til filen. "x" symboliserer retten til å eksekvere filen. For å få rett til å eksekvere filen kan man bruke kommandoen "chmod +x <filnavn>". Eventuelt kan man skrive "chmod 777 <filnavn>", dette vil gi alle (root og brukere) alle rettigheter (read, write, execute), noe man egnetlig vil ungå med mindre alle har behov for alle rettigheter.

Kommentar til de neste oppgaveløsningene:

Debugger: Jeg har valgt å bruke debugger libraryet vi lagde tidligere i semesteret i en av oppgavene. Jeg var usikker på hvordan jeg skulle gjøre det med mappestrukturen. Jeg vurderte å lage en egen lib-mappe med .a fila og headerfilen utenfor oppgavemappene slik at jeg enkelt kunne bruke debuggeren i alle programmene uten å måtte kopiere med filene til hver oppgave. Jeg valgte i stedet å legge de ved i oppgave 3, siden det er den med mest kode, og kun bruke debuggeren der. Jeg kunne referert til fila i oppgave 3 mappen i de andre oppgavene også, men synes det ble litt rotete. Jeg har derfor valgt å ikke bruke debuggeren i de andre oppgavene.

Oppgave 2: Filhåndtering Deloppgave A

Programmet kjører som det skal.

Testet med valgrind, no errors, all heap blocks freed, no leaks.

($ valgrind –tool=memcheck –leak-check=yes ./main)

Bygg:

$ make

($ make clean && rm decoded.txt)

Kjør:

$ ./main

Se resultat:

$ cat decoded.txt

Oppgave 2: Filhåndtering Deloppgave B

Jeg startet denne oppgaven med en kopi av deloppgave A.
Jeg la til en ny fil, letterCounter.c, her her jeg en funksjon som går gjennom alle tegn, teller dem, og hvis det er en bokstav, skriver det ut i terminalen.
Jeg husker at vi hadde en funksjon som svarte på denne oppgaven på tavlen i en lab, som utførte operasjonen på en mye kulere måte med å bitshifte, men i stedet for å bruke tid på å huske hvordan den fungerte skrev jeg en funksjon som gjør det på den første måten jeg kunne tenke på for å få jobben gjort. Hadde det ikke vært for andre eksamner hadde jeg heller brukt tid på å feks først isAlpha() og så or'et (|=) med 0x20 for å gjøre alt til lowercase istedet for å bruke tolower(). Og kanskje istedet for isAlpha(), ha et array med chars å sammenligne med, eller jeg kun tatt utgangspunkt i 0x41 og telt meg gjennom ascii-tabellen (hadde tatt hensyn til at det er noen tegn mellom store og små bokstaver). "Veier til roma og sånn.."

Programmet kjører som det skal. Det inneholder fremdeles funksjonen for å lage decoded.txt filen. Denne er kommentert ut i main, og jeg endret slik at man skriver inn filnavn som parameter i stedetfor å ha det hardkodet.

Testet med valgrind, no errors, all heap blocks freed, no leaks.
($ valgrind –tool=memcheck –leak-check=yes ./main)

Bygg:
$ make
Kjør:
$ ./main

Oppgave 3: Listehåndering

Jeg startet med å misforstod oppgaven. Jeg lagde et program der man som en kunde velger produkter fra en meny som blir lagt til en liste, som man kan liste opp og summere. Grunnen til at jeg gjorde det på den måten var siden jeg tenkte at siden man hadde med antall i structen måtte man kunne legge til flere av samme produkt flere ganger og det måtte ha samme pris hver gang. Evt måtte man kontrollere at prisen var den samme som sist gang man la til produktet.
I stedet for å starte på nytt, tok jeg det som en utfordring å heller implementere at man OGSÅ kunne skrive inn "navn", "pris" og "antall" manuelt og legge det til den samme listen.
Dette fungerer. Etter å ha gjort oppgaven ferdig kom jeg på debuggern vi lagde tidligere i semesteret. Jeg har lagt den til i ettertid og gjort at den logger feilmeldinger der jeg før kun brukte printf(). Hadde jeg husket dette tidligere ville jeg brukt debugmeldingen veldig mye. Jeg har gjennom kodingen hatt div printf() som har skrevet ut verdier eller hvor jeg er i koden osv for å hjelpe for feilsøking, og det hadde blitt mer oversiktlig for ikke å snakke om sett mye proffere ut om jeg hadde brukt debuggern hele veien. Da kunne jeg latt alle printfene stå og satt btranceEnable til 0 når jeg ikke lenger ville debugge, så kunne man også fått et inntrykk av alle steder jeg har stått fast i løpet av kodingen i programmet.

Programmet kjører som det skal.
Testet med valgrind, no errors, all heap blocks freed, no leaks.
($ valgrind –tool=memcheck –leak-check=yes ./main)

Bygg:
$ make
Kjør:
$ ./main

Oppgave 4: Finn 3 feil

Feil: "char szServer[16]" skapte en buffer overflow dersom servernavnet var på 16+ karakterer.
Fix: Endret 16 til 32. (Tenker at szContentType kanskje også kan støte på lignende problemer..)

Feil: "if (pHttp->iHttpCode = 200)" setter iHttpCode til 200 i stedet for å sjekke om den er 200.
Fix: Endret "=" til "==".

Feil: "pHttp->iContentLength = atoi(pszHttp);" bruker feil peker som argument.
Fix: Endret "pszHttp" til "pszPtr", slik at den leser fra den pekern den akkurat har satt der hvor content-length står og ikke fra pekern til strengen den fikk inn som parameter.

Programmet kjører som det skal etter disse feilene er rettet.
Jeg har lagt til "Last-Modified" til å vise ukedag, dato, årstall og klokkeslett, siden det var det jeg fant i http documentasjonen at last-modified skulle inneholde.
Når programmet kjøres printer det først ut stringen som blir brukt som parameter i funksjonen, deretter printer den ut alle verdiene i structen.

Testet med valgrind, no errors, all heap blocks freed, no leaks.
($ valgrind –tool=memcheck –leak-check=yes ./finn3feil)

Bygg:
$ make
Kjør:
$ ./finn3feil

Oppgave 5: Tråder

Jeg gjorde først oppgaven uten å bruke tråder. Det gikk veldig enkelt. Lite visste jeg om hvor mye hodebry det skulle bli å få det til å fungere med en ekstra tråd. Til slutt skjønte jeg hintene fra oppgaveteksten, etter å ha funnet ut hvordan man kunne passe en strukt med i funksjonen full av argumenter.
Jeg har valgt å gjøre bufferet en char lenger for å legge på \0 i slutten slik at det ble lettere å lese med en while loop. Jeg la inn et "flag" for å signalisere hvilken tråd som jobber og når den er ferdig med å lese en streng og en pointer jeg bruker for å lese meg gjennom bufferet. Den er mulig litt overflødig, men det var den enkleste måten jeg kunne komme på å skrive på, og siden det funket så ble den der.

Programmet kjører som det skal.
Testet med valgrind, no errors, all heap blocks freed, no leaks.
($ valgrind –tool=memcheck –leak-check=yes ./main)

Bygg:
$ make
Kjør:
$ ./main

Oppgave 6: Nettverk

Oppgaven ber om at applikasjonen skal skrive ut websiden den mottar i terminalen.
Jeg har valgt at applikasjonen også skriver ut adressen den prøver å koble til (i hex) og at den har koblet til og meldingen den sender. Dette bare for oversiktens skyld.
Deretter skriver den ut bodyen fra responesen den mottar. Jeg flytter pointeren til \r\n\r\n, som er signalet på at det ikke er flere headere også (hopper den 4 frem for å ikke ta med \r\n\r\n og) skriver jeg ut det den leser derfra som er bodyen, html kode. Jeg har ikke brukt til på å oversette html for å lage kul formatering av teksten selv om det hadde vært moro å prøve på, men det hadde tatt for mye tid.

Programmet kjører som det skal.
Testet med valgrind, no errors, all heap blocks freed, no leaks.
($ valgrind –tool=memcheck –leak-check=yes ./main)

Bygg:
$ make
Kjør:
$ ./client