2 ukers HJEMMEEKSAMEN
PG3401 C Programmering
Tillatte hjelpemidler: Alle
Varighet: 14 dager
Karakterskala/vurderingsform: Nasjonal karakterskala A - F
Dato: 8.-22. desember 2022

Oppgavesettet har 6 sider. Det er totalt 7 oppgaver i oppgavesettet.

Det er 2 ukers frist på denne hjemmeeksamen, men forventet arbeidsmengde er 5-7 dager med «normale» arbeidsdager. Perioden kan overlappe med andre eksamener dere har, det er derfor viktig at dere bruker tiden effektivt i starten av eksamensperioden så dere ikke rett før innlevering blir sittende og skulle levere flere eksamener samtidig. Vær obs på at eksamen MÅ leveres innen fristen som er satt i Wiseflow, og oppgaven kan kun leveres via WISEFLOW. Det vil ikke være mulig å få levert oppgaven etter fristen – det betyr at du bør levere i god tid slik at du kan ta kontakt med eksamenskontoret eller brukerstøtte hvis du har tekniske problemer.

Det presiseres at studenten skal besvare eksamen selvstendig og individuelt, samarbeid mellom studenter og plagiat er ikke tillatt. Eksamen skal løses på Linux.

Merk at oppgavene er laget med stigende vanskelighetsgrad, og spesielt de tre siste oppgavene er vanskeligere enn de første oppgavene. Det oppfordres derfor til å gjøre de første oppgavene (helt) ferdige slik at studenten ikke bruker opp all tid på å gjøre de siste oppgavene først.

Format på innlevering

Dette er en praktisk programmeringseksamen (bortsett fra oppgave 1), fokus bør derfor være på å forklare hvordan du har gått frem, begrunne valg og legge frem eventuelle antagelser du har gjort i din løsning.

Hvis du ikke klarer å løse en oppgave er det bedre om du forklarer hvordan du har gått frem og hva du ikke fikk til – enn å ikke besvare oppgaven i det hele tatt. Det forventes at alt virker hvis ikke annet er beskrevet i tekstbesvarelsen, hvis du vet at programmet krasjer, ikke kompilerer eller ikke virket slik den var tenkt er det viktig å forklare dette sammen med hvilke skritt du har tatt for å forsøke å løse problemet.

Besvarelsen skal være i 1 ZIP fil, navnet på filen skal være PG3401_H22_[kandidatnummer].zip. Denne filen skal ha følgende struktur:
\ oppgave_2 \ makefile
\ oppgave_2 \ […]
[…]

I Wiseflow skal du laste opp en tekstbesvarelse med navn «PG3401_H22_[kandidatnummer].pdf» (PDF og Microsoft Word format godkjennes), og ZIP filen skal lastes opp som vedlegg til tekstbesvarelsen.

Vær sikker på at alle filer er med i ZIP filen. Hvis du velger å ha flere programmer som en del av løsningen så legger du det ene programmet i (for eksempel) \oppgave7_A og den andre i \oppgave7_B. Hver mappe skal ha en makefile fil, og det skal ikke være nødvendig med noen endringer, tredjeparts komponenter eller parametere – sensor vil i shell på Debian Linux 10 gå inn i mappen og skrive «make» og dette skal bygge programmet med GCC.

Tekstbesvarelsen skal inneholde besvarelse på oppgave 1 (skriv det kort og konsist, trenger ikke noe stor avhandling). Etter den rene tekstbesvarelsen skal det være 1 sides begrunnelse/dokumentasjon for hver oppgave, hver av disse begrunnelsene skal være på en ny side for å gjøre tekstbesvarelsen oversiktlig for sensor. Besvarelsen skal være i PDF format eller i Microsoft Word format (DOCX) og ha korrekt file-extension til å kunne åpnes på både en Linux og en Windows maskin (.pdf eller .docx). Besvarelser i andre formater vil ikke bli lest.

Oppgave 1. Generelt (5 %)

a) Forklar hva C programmeringsspråket kan brukes til.

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

c) Forklar hva verktøyet valgrind brukes til når man programmerer på Linux.

Oppgave 2. Filhåndtering (15 %)

Opprett en fil med følgende innhold:

```
Dette er en test på oppgave 2, eksamen 2022. Teksten skal
konverteres til hexadesimalt enkodede ascii tegn.
```

Teksten skal lagres i en tekstfil bestående av 7 bit ASCII, og teksten skal være på 1 linje (se bort fra linjeskift du ser i denne oppgaveteksten).

Skriv et program som leser denne filen, programmet skal så opprette en ny fil hvor hver bokstav/tegn skal skrives som ASCII representasjon av den hexadesimale verdien til bokstaven/tegnet i ASCII tabellen.

Eksempel: Første bokstav i teksten er en stor D, i ASCII tabellen er dette hexadesimalt 0x44. I output filen skal du derfor skrive de to karakterene 4 og 4 for dette første «tegnet». (Med andre ord vil output filen være dobbelt så stor som input filen). De fem første bokstavene i input teksten vil altså resultere i følgende ti tegn i output filen: 4465747465.

Både input filen og output filen skal være i samme katalog som programmet, og begge filene skal være en del av innlevert besvarelse.

Oppgave 3. Liste håndtering (20 %)

Du skal lage en dobbeltlenket liste, hvert element (struct) i listen skal inneholde pekere til både forrige element og neste element. Elementet skal også inneholde en tekststreng som inneholder NAVN, en tekststreng som inneholder ROMNUMMER, en integer for å representere DATO, en integer som inneholder ANTALL DØGN, og en float som inneholder PRIS PER DØGN. Totalt sett utgjør listen et reservasjonssystem for et hotell.

Du skal lage funksjoner som utfører følgende operasjoner på listen:

- Legge til et element i slutten av listen (reserver et rom)
- Slette siste element i listen (en «angreknapp»)
- Sletter alle elementer i listen som har DATO + ANTALL (dager) eldre enn dagens dato
- Søker gjennom listen etter en reservasjon, gitt et NAVN på en gjest
- Summerer sammen total verdi av bookingene for en gitt dag (sum av PRIS PER DØGN) for en dato som oppgis som parameter til funksjonen
- Printer ut en reserverasjonsliste (i terminal vinduet) med alle reservasjoner for dagens dato, med navn, romnummer og antall døgn gjesten skal bli

Du skal lage en main funksjon som mottar instrukser fra bruker basert på input fra tastaturet, main må altså kunne kalle alle de seks funksjonene over (for eksempel i en form for meny) og be brukeren om data som er nødvendig for å kalle de nevnte funksjoner. Main skal rydde opp alle data før den returnerer (et valg i menyen må være å avslutte).

Oppgave 4. Finn 3 feil (10 %)

Du har fått beskjed om å utføre feilretting av den følgende funksjonen, du får beskjed om at Content-Length ikke blir satt riktig i den returnerte structen. (Funksjonen kalles med en tekstbuffer som inneholder HTTP REPLY fra en webserver, og structen blir frigitt av funksjonen som kaller funksjonen.)

Det viser seg at koden faktisk inneholder 3 feil. Du skal rette alle feilene, i besvarelsen skal du ha den feilrettede funksjonen, og en main metode som tester funksjonen (der hvor det er mulig bør main kalle funksjonen på en slik måte at den fremprovoserer feilene som var i den opprinnelige koden. I tekstbesvarelsen skal du kort forklare hvilke feil du fant, hvorfor dette er feil og forklare hvordan du løste dem.

Det er også ønsket at en ny verdi legges til i structen som sier hvilken http versjon som er benyttet (typisk HTTP/1.0, HTTP/1.1 eller HTTP/2.0), angitt som en float verdi, du skal skrive denne koden og legge den til i funksjonen.

```
typedef struct _MYHTTP { int iHttpCode; int iContentLength;
bool bIsSuccess;  char szServer[16]; char szContentType[16]; } MYHTTP,
*PMYHTTP;

MYHTTP* ProcessHttpHeader(char *pszHttp) {
   char* pszPtr;
   MYHTTP* pHttp = (MYHTTP*)malloc(sizeof(PMYHTTP));
   if (!pHttp) return NULL;
   memset(pHttp, 0, sizeof(MYHTTP));

   pHttp->iHttpCode = atoi(pszHttp + strlen("HTTP/1.x "));
   if (pHttp->iHttpCode == 200) {
      pHttp->bIsSuccess = true;
   }

   pszPtr = strstr(pszHttp, "Server");
   if (pszPtr) {
      pszPtr += 6; while (!isalpha(pszPtr[0]))pszPtr++;
      strchr(pszPtr, '\n')[0] = 0;
      strncpy(pHttp->szServer, pszPtr, 15);
      pszPtr[strlen(pHttp->szServer)] = '\n';
   }

   pszPtr = strstr(pszHttp, "Content-Type");
   if (pszPtr) {
      pszPtr += 12; while (!isalpha(pszPtr[0]))pszPtr++;
      strchr(pszPtr, '\n')[0] = 0;
      strcpy(pHttp->szContentType, pszPtr);
      pszPtr[strlen(pHttp->szContentType)] = '\n';
   }

   pszPtr = strstr(pszHttp, "Content-Length");
   if (pszPtr) {
      pszPtr += 14; while (!isdigit(pszPtr[0])) pszPtr++;
      pHttp->iContentLength = '0' + atoi(pszPtr);
   }

   return pHttp;
}
```

Oppgave 5. Tråder (20 %)

I praktisk programmering er det ofte effektivt å legge tidkrevende operasjoner ut i arbeidstråder, eksempler på dette er filoperasjoner, nettverksoperasjoner og kommunikasjon med eksterne enheter. I denne oppgaven skal du simulere slike operasjoner med et mindre datasett.

Du skal lage en applikasjon som består av 3 tråder, hovedtråden (som startet main) og 2 arbeidstråder. Når hovedtråden starter skal den starte arbeidstrådene med mekanismer for tråd-kommunikasjon og synkronisering (i denne oppgaven har ikke hovedtråden noen annen funksjon, og skal kun starte de to trådene som skal gjøre selve jobben). Trådene skal ha et minneområde med plass til 4096 byte. (Det kan være en struct som også inneholder andre kontrolldata som for eksempel antall byte i bufferet og annet studenten finner nyttig.)

Den ene arbeidstråden (tråd A) skal lese PDF filen som er oppgaveteksten til denne eksamen, arbeidstråd A skal så sende filen over til den andre arbeidstråden (tråd B) (gjennom flere sykluser ved hjelp av minneområdet beskrevet over, og signalere tråd B om at det er data tilgjengelig i bufferet). Tråd B skal så telle opp antall forekomster av bytes med verdi 00, 01, 02, og så videre til; FF i filen den får sendt over. Tråd A og tråd B skal gå i loop for å prosessere PDF filen helt til den er ferdig. Når filen er sendt over i sin helhet skal arbeidstråd A avslutte. Arbeidstråd B skal fullføre sin opptelling av bytes, og så skrive ut til terminal vinduet antall forekomster av hver av de 256 mulige byte-verdiene, før den også avslutter. Hovedtråden (main) skal vente på at begge trådene avslutter, rydde opp korrekt og så avslutte applikasjonen.

Oppgave 6. Nettverk (15 %)

Du skal i denne oppgaven lage (en del av) en enkelt nettleser (browser) og en web server. Browser og server skal støtte HTTP GET for å kunne laste ned en fil. Disse to applikasjonene skal leveres inn i hver sin mappe, med hver sin makefile; oppgave6_srv og oppgave6_klient.

Oppgaven skal ikke løses ved bruk av Curl eller andre tredjepartsbiblioteker, og skal ikke basere seg på wget eller tilsvarende fra operativsystemet – kun bruk av Sockets slik vi har lært på forelesning 11 om Nettverk vil gi poeng på oppgaven.

For ordens skyld nevnes spesifikt at det ikke vil trekkes i poeng for å gjenbruke kode fra løsningsforslag som ble presentert på siste forelesning når dere lager klienten, kommenter i så fall hvilke endringer dere har gjort i koden for å fungere til denne oppgaveteksten. (De fleste poeng i denne oppgaven tilligger server applikasjonen.)

Web serveren skal BINDE (lytte) til en port du velger, det anbefales for denne oppgaven å kun lytte på adresse 127.0.0.1 for å ikke eksponere porten utenfor egen maskin. Web serveren skal akseptere HTTP GET requester over TCP som følger HTTP spesifikasjonen. Serveren skal anse gjeldende mappe som «root» og åpne filer relativ til gjeldende mappe. Filen klienten ber om som «URL» skal sendes tilbake til klienten som en del av serverens HTTP GET reply i henhold til HTTP spesifikasjonen. Serveren skal så lukke forbindelsen til klienten, og vente på en ny klient som kobler seg til.

Browseren (klienten) skal startes med et filnavn som parameter og skal CONNECTE

til 127.0.0.1 på porten du har valgt, og sende en HTTP GET request hvor filen (som ble oppgitt som parameter) skal lastes ned fra server applikasjonen. Klient applikasjonen skal lese HTTP response den får tilbake fra server, lagre denne filen i sin mappe, for så å avslutte.

Applikasjonene kan for eksempel testes ved at klienten laster ned kildekoden til serveren, ved for eksempel «oppgave6_klient include/httpfunc.h».

Da kun opplasting av filer skal støttes kan det tas en del snarveier, deriblant kan HTTP headere stort sett hardkodes med faste verdier. Studenten kan velge å kun støtte filer med tekst (ikke binære filer), men det vel naturligvis telle positivt hvis studenten klarer å også støtte binære filer (for eksempel bilder).

Oppgave 7. Filhåndtering (15 %)

Du skal lage en applikasjon som fungerer som en «kode beautifier», en applikasjon som endrer kildekode til noe som passer forfatterens kodestil (gjør koden «penere»).

Applikasjonen skal ta et filnavn til en C-source fil som parameter når den startes fra terminal. Applikasjonen skal lese denne filen og gjøre to endringer i filen før den lagrer filen med samme navn, men lagt til _beautified før .c i filnavnet.

Først skal applikasjonen ta alle forekomster av for-looper og erstatte disse med while-looper, det betyr at kode som dette:

```
for (a = 0; a < b; a++) {…}
```

skal erstattes med en while loop og vil se noe slik ut:

```
a = 0;

while (a < b) {… a++; }
```

Hvor … angir resten av koden i løkken, og naturligvis må beholdes slik den er.

I tillegg skal alle forekomster av tab (ASCII kode 0x09) erstattes med 3 spacer (mellomrom, ASCII kode 0x20).

+

Slutt på oppgavesettet