2 ukers HJEMMEEKSAMEN
PG3401 C Programmering
Tillatte hjelpemidler: Alle
Varighet: 14 dager
Karakterskala/vurderingsform: Nasjonal karakterskala A - F
Dato: 1.-15. november 2021

Oppgavesettet har 6 sider. Det er totalt 6 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 to 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.

Spørsmål fra studenter i løpet av eksamensperioden skal stilles under Diskusjoner på emnesidene på Canvas; svar vil kun gis der slik at alle studenter får den samme informasjonen. Det betyr at alle studenter bør følge med på Canvas under Diskusjoner for å få med seg eventuelle presiseringer og oppklaringer som måtte komme som en følge av spørsmål fra medstudenter. Spørsmål kan også stilles i plenum på siste forelesning i emnet.

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_H21_[kandidatnummer]. Denne filen skal ha følgende struktur:
\ oppgave_2 \ makefile
\ oppgave_2 \ […]
[…]

I Wiseflow skal du laste opp en tekstbesvarelse med navn «PG3401_H21_[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 \oppgave_xA og den andre i \oppgave_xB. 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) 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.

Oppgave 2. Filhåndtering (15 %)

Opprett en fil med følgende innhold:

```
4465747465206572206e2066696c206465726520736b616c207465737465
206d65642c2064656e20696e6e65686f6c6465722074656b737420736f6d20
657220686578206564656e6b6f6465742e2044656e6e65206f7070676176656e20
696c6c7573747265726572657220656e6b656c206c6573696e672061762066696c
6572206f6720617420646572652065206b616e20746120656e20656e6b656c206b
6f6e766572746572696e67206176206461746120612073616d7420656e6b656c
```

```
206c6f6f70206b6f64652e204465747465206b6c61726572206475206e6f6b
2066696e742e
```

Innholdet er en tekst som er enkodet som hexadesimale ASCII tegn. Både den enkodede og dekodede teksten er i 7 bit ASCII, og den enkodede teksten er på 1 linje (se bort fra linjeskift du ser i denne oppgaveteksten).

Deloppgave A

Skriv et program som leser denne filen, dekoder dataene, og skriver resultatet til en annen fil.

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

Eksempel: Første «tegn» i datasettet er 44, dette er med andre ord ASCII koden 0x44, som er en stor bokstav D. I output filen skal du derfor skrive D for dette første «tegnet».

Deloppgave B

Du skal så lage en funksjon som tar de dekodede dataene som parameter og teller opp antall forekomster av hver bokstav (A til Z), funksjonen skal telle både store og små bokstaver som samme bokstav, og skal overse andre tegn. Programmet ditt skal skrive ut i konsoll vinduet antall forekomster av hver bokstav.

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 VARENAVN, en integer som inneholder ANTALL, og en float som inneholder PRIS PER VARE. Totalt sett utgjør listen en salgskvittering i en matvarebutikk.

Du skal lage funksjoner som utfører følgende operasjoner på listen:

- Legge til et element i slutten av listen
- Slette siste element i listen (en «angreknapp»)
- Sletter alle elementer i listen som har et gitt VARENAVN (sendes som parameter til funksjonen)
- Summerer sammen total pris for varene i listen (husk å ta hensyn til antall)
- Printer ut en kvittering (i konsoll vinduet)

Du skal lage en main funksjon som mottar instrukser fra bruker basert på input fra tastaturet, main må altså kunne kalle alle de fem 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 (15 %)

Du har fått beskjed om å utføre feilretting av denne 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 hvilket årstall websiden sist ble endret (Last-Modified), 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;

MYHTTP* ProcessHttpHeader(char *pszHttp) {
   char* pszPtr;
   MYHTTP* pHttp = (MYHTTP*)malloc(sizeof(MYHTTP));
   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;
      strcpy(pHttp->szServer, pszPtr);
      pszPtr[strlen(pHttp->szServer)] = '\n';
   }

   pszPtr = strstr(pszHttp, "Content-Type");
   if (pszPtr) {
      pszPtr += 12; while (!isalpha(pszPtr[0]))pszPtr++;
      strchr(pszPtr, '\n')[0] = 0;
      strncpy(pHttp->szContentType, pszPtr, 15);
      pszPtr[strlen(pHttp->szContentType)] = '\n';
   }

   pszPtr = strstr(pszHttp, "Content-Length");
```

```
   if (pszPtr) {
      pszPtr += 14; while (!isdigit(pszPtr[0])) pszPtr++;
      pHttp->iContentLength = atoi(pszHttp);
   }

   return pHttp;
}
```

Oppgave 5. Tråder (25 %)

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 2 tråder, hovedtråden (som startet main) og en arbeidstråd som skal skrive data. Når hovedtråden starter skal den først starte arbeidstråden med mekanismer for tråd-kommunikasjon og synkronisering. Hovedtråden og arbeidstråden skal ha et minneområde med plass til 10 tegn. (Det kan være en struct som også inneholder andre kontrolldata som for eksempel antall tegn i bufferet og annet studenten finner nyttig – men ikke mer enn 10 tegn som en tekstbuffer. Det er lov for en slik struct å inneholde char a[11] hvis det ellevte tegnet kun brukes til en 0-terminering.)

Hovedtråden skal be brukeren gjennom terminal vinduet om å skrive inn en tekststreng og avslutte med [enter]. Når brukeren har skrevet inn en streng skal hovedtråden sende strengen over til arbeidstråden (gjennom flere sykluser ved hjelp av minneområdet beskrevet over, og signaler eller andre synkroniseringsmekanismer hvis mer enn 10 tegn skrives inn av brukeren). Arbeidstråden skal lagre denne tekststrengen til en fil (linjeskift '\n' skal skrives når hele tekststrengen er skrevet til filen). Når operasjonen er fullført skal hovedtråden be brukeren om en ny tekststreng, eller hvis brukeren oppgir «quit» skal applikasjonen avslutte. Applikasjonen skal gå slik i en loop og sende tekst over til arbeidstråden helt frem til brukeren avslutter, hver tekststreng skal appendes (legges til på slutten) filen.

Oppgave 6. Nettverk (20 %)

Du skal i denne oppgaven lage en enkelt nettleser (browser). Applikasjonen skal hente data fra en reel webserver på internett, hvilket betyr at maskinen (og VMen hvis du kjører Linux i en VmWare maskin slik vi har brukt i undervisningen) må ha internett tilgang når du løser oppgaven.

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.

Applikasjonen (den enkle browseren) skal laste ned følgende URL:

«http://www.eastwillsecurity.com/pg3401/test.html»

Applikasjonen kan startes uten parameter, og main metoden kan hardkode protokoll TCP, port nummer 80 og server manuelt (du kan også hardkode IP adressen til www.eastwillsecurity.com hvis du ønsker). Klient applikasjonen skal CONNECTE til oppgitt port på webserveren, og sende en HTTP GET pakke til URLen som oppgitt over.

Klient applikasjonen skal så lese svaret den får tilbake fra webserveren, og skrive ut websiden som kommer tilbake i terminal vinduet, før den avslutter.

+

Slutt på oppgavesettet.