UiO : Universitetet i Oslo

Hjemmeeksamen: filnavigator for terminalvinduer

Utlevering: onsdag 11. November 2015

Innlevering: onsdag 25. November 2015, kl 23:59

Denne hjemmeeksamen går ut på å programmere to program: tjener og klient. Det skal brukes programmeringsspråket C, og koden som leveres skal fungere med kompilatoren gcc på Ifis Linux-maskiner. Det skal skrives en rapport.

Hjemmeeksamen er karaktertellende for kurset (ca. 40%). All kode *må* være skrevet selv: det er ikke lov å hente kode fra annet hold eller dele egen kode med andre. Det er derimot tillatt – og vi oppfordrer sterkt til – å diskutere problemer og utveksle kunnskap med andre studenter. (Men altså ikke dele kode direkte.)

Det er *viktig* at du leser eget avsnitt om hvordan oppgaven skal leveres inn.

Introduksjon

For å finne fram på sin lokale harddisk, bruker man veldig ofte verktøy som for eksempel en lokal filnavigator. På Ifi bruker vi i tillegg NFS, som gjør at operativsystemet lar det se ut som om filer som ligger på filtjeneren også er lokale, men i mange andre sammenhenger bruker man spesielle filnavigatorer for slike eksterne filer. Det skjer ofte at man bruker web-browseren bevisst eller ubevisst som filnavigator for eksterne filer når man finner en lesbar katalog uten index.html; webtjeneren sender til web-browseren en liste over filer i katalogen inklusive størrelsesinformasjon. Den tilbyr også mulighet til å vise filene hvis rettighetene stemmer.

Oppgaven

I denne oppgaven skal dere lage en tjener (filtjener) og en klient (filnavigator) som gjør det mulig å:

- navigere mellom alle kataloger innenfor den katalogen som tjeneren startes i, men aldri utenfor denne katalogen
- vise innholdet i den aktuelle katalogen
- forandre den aktuelle katalogen (change directory)
- velge et navn fra den aktuelle katalogen og skrive ut filtypen: om navnet tilsvarer en vanlig

fil, en katalog, en link eller en type spesialfil

- velge et navn fra den aktuelle katalogen og skrive innholdet ut på klientens skjerm (erstatt tegn som ikke er «printable» med punktum, da disse ikke kan skrives til skjerm)

Klienten og tjeneren *skal* kommunisere over en TCP-forbindelse. Klienten skal kjøre i et terminalvindu og ikke åpne flere vinduer. (For interesserte er det lov å bruke Ncurses, men det gir ikke ekstra uttelling.) Klienten skal tilby funksjonene sine i en interaktiv dialog, som *for eksempel* kan se slik ut:

```
cmd (? for help)> ?
! You are connected to the server
! Please press a key:
! [1] list content of current directory (ls)
! [2] print name of current directory (pwd)
! [3] change current directory (cd)
! [4] get file information
! [5] display file (cat)
! [?] this menu
! [q] quit
cmd (? for help)> 2
! /ifi/fenris/p13/nd
cmd (? for help)> 3
new dir (? for help)> ?
! ..     the parent directory
! / a new absolute directory
!   a new directory relative to the current position
! [?]    this menu
! [q]    leave this menu
new dir (? for help)> ..
! OK
cmd (? for help)> 1
! README
! loesfors
! loesfors.tgz
cmd (? for help)> 4
! Get file information for:
! [1] README
! [2] loesfors
! [3] loesfors.tgz
! [q] leave this menu
file (? for help)> 3
! loesfors.tgz is a regular file
cmd (? for help)> q
```

All kommunikasjon *skal* skje over en TCP-forbindelse. Klienten skal nå ikke lese fra disk. Klienten skal lytte til både input fra brukeren og data fra serveren. Tjeneren skal være i stand å opprettholde forbindelse med flere klienter samtidig og ha en egen «aktuelle katalog» for

enhver klient. I neste avsnitt beskrives alle spørsmål som skal besvares, inklusive programmeringsdelen beskrevet over.

Oppgaven i detalj

1. Ved bruk av TCP er det viktig å lage en egen *protokoll* (applikasjonslags-protokoll) på toppen av Berkeley socket laget. Hvilke egenskaper har TCP som gjør denne protokollen så viktig?
2. Hvordan ser din applikasjonslags-protokoll ut? Begrunn designet!
3. Ta hensyn til maskinarkitekturen! Hva kan gå galt? Ville programmet ditt virke hvis du kompilerer og kjører tjeneren din på en arkitektur med Big Endian, og klienten på en av Linux-maskinene i Ifis terminalstuer (som kjører Little Endian)?
4. Programmet ditt bruker forbindelsesorientiert kommunikasjon, men nettfilsystemet NFS bruker hovedsaklig forbindelsesløs kommunikasjon. Hvilke problemer unngår du i programmet ditt ved å bruke forbindelsesorientiert kommunikasjon?
5. Hvilke muligheter tilbyr Linux og Berkeley socket APIet for å utvikle tjenere som kan opprettеholde flere forbindelser samtidig, og som kan kommunisere med flere klienter samtidig? Hvilken av mulighetene har du valgt og hvorfor?
6. Implementer filnavigatoren og filtjeneren.
   - Bruk flere filer, f.eks. slik at du skriver applikasjonslags-protokollen din én gang og bruker koden både i klienten og tjeneren.
   - Det må være mulig å kompilere og testkjøre programmene dine på Linux-maskinene i Ifis terminalstuer.
   - Tjeneren skal ta et portnummer som kommandolinje-argument.
   - Klienten skal ta et maskinnavn og et portnummer som kommandolinje-argumenter.

Kompilering av koden

For å kompilere koden skal dere lage en `Makefile` (see `info make`) slik at man bare kan skrive `make` i katalogen hvor filene er lagret for å kompilere programmet. Makefilen skal også leveres inn og gir poeng.

Noen tips

Noen funksjoner og annen informasjon som dere kan tenkes å ha bruk for:

- `lstat` (man 2 lstat)
- `isprint` (man 3 isprint)

- `snprintf` (man 3 snprintf)
- `strncmp` (man 3 strncmp)
- `strlen` (man 3 strlen)
- `fgets` (man 3 fgets)
- `perror` (man 3 perror)
- `system` (man 3 system)
- Skjerm I/O for terminaler skjer i Unix gjennom en filabstraksjon. Programmet leser det som brukeren skriver fra `stdin` eller fildeskriptor 0, og programmet skriver til skjerm enten ved å skrive til `stdout` (fildeskriptor 1) eller til `stderr` (fildeskriptor 2). Det betyr at dere kan bruke de vanlige funksjonene `read()`, `write()`, `send()`, `recv()`, `select()`, osv. også for å lese fra tastatur og skrive til skjerm. Det kan være nyttig å vite.
- Interaktive klienter vil kunne forandre tilstanden til en TCP socket som de bruker i kommunikasjon med serveren ved å bruke:

```
int activate=1;
setsockopt(sock, IPPROTO_TCP, TCP_NODELAY, &activate, sizeof(int));
```

- Servere kan unngå noen småproblemer ved å bruke:

```
int activate=1;
setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &activate, sizeof(int));
```

- For de som skjønner dokumentasjonen og har lyst å eksperimentere, åpner `dup2` (se `man 2 dup2`) noen spennende snarveier.
- `Readv` (man 2 readv) og `writev` (man 2 writev) finnes på POSIX systemer men f.eks ikke på Windows. De som liker eksperimenter, kan prøve å unngå en del kopiering ved å bruke writev istedenfor write og send.

Innlevering - VIKTIG!

Besvarelsen skal bestå av den *godt kommenterte* kildekoden til programmet, makefilen samt en rapport hvor dere svarer på spørsmålene (og eventuelt beskriver det dere har gjort og begrunner eventuelle valg dere har tatt). Under finner dere instruksjoner på hvordan dere skal levere inn, og ved å levere inn oppgaven samtykker dere på at dere overholder reglene for innleveringen.

Rapport

I forbindelse med de teoretiske spørsmålene (og eventuelt forklaringer i forbindelse med implementasjonen), skal det leveres en rapport. Denne rapporten skal leveres elektronisk som **PDF**. Bruk punkt 11, enkel linjeavstand og normale marger.

Elektronisk innlevering

Alt skal leveres elektronisk hvor alle filer (Makefile, *.c, *.h, rapport.pdf, etc.) er samlet i én katalog med *kandidatnummeret* som navn. Av denne katalogen lager dere en komprimert tar-ball – bruk kommandoen `tar zcvf knr.tgz knr` der *knr* er kandidatnummeret ditt. Den elektroniske innleveringen skal leveres via Devilry.

Når skal man levere?

Man må ha sendt elektronisk versjon – komprimert tarball – *FØR kl. 23:59, den 25.11.2015*. Merk at denne tidsfristen er *HARD*, det vil si at oppgaver levert etter fristen betegnes som «manglende innlevering» og gir automatisk karakteren F (stryk) på denne oppgaven!

Krav til innleverte oppgaver ved Institutt for informatikk

Emnet INF1060 har en eksamensordning som går ut på at enkelte oppgaver i kurset bedømmes med karakter som utgjør en del av den endelige karakteren som gis i kurset (denne oppgaven teller ca. 40%). Derfor skal ALL kode i forbindelse med de karaktergivende oppgavene skrives av studenten selv.

Regler for innlevering og bruk av kode

Ved alle pålagte innleveringer av oppgaver ved Ifi enten det dreier seg om obligatoriske oppgaver, hjemmeeksamen eller annet forventes det at arbeidet er et resultat av studentens egen innsats. Å utgi andres arbeid for sitt eget er uetisk og kan medføre sterke reaksjoner fra Ifis side. Derfor gjelder følgende:

1. Deling (både i elektronisk- og papirform) eller kopiering av hele eller deler av løsningen utviklet i forbindelse med de karaktergivende oppgavene i kurset er ikke tillatt.
2. Deling/distribuering av kode og oppgavetekst med personer som ikke er eksamensmeldt i INF1060 dette semesteret, med unntak av kursledelsen og gruppelærere, er ikke tillatt.
3. Hente kode fra annet hold, f.eks. fra andre ``open source'' prosjekter eller kode funnet på nettet er ikke tillatt.
4. Det er greit å få generelle hint om hvorledes en oppgave kan løses, men dette skal eventuelt brukes som grunnlag for egen løsning og ikke kopieres uendret inn.
5. Kursledelsen kan innkalle studenter til samtale om deres innlevering.

Samarbeid

Reglene om kopiering betyr ikke at Ifi fraråder samarbeid. Tvert imot, Ifi oppfordrer studentene til å utveksle faglige erfaringer om det meste. Vi oppfordrer til at studentene skal kunne lære av hverandre, men, som sagt, man skal ikke dele/distribuere/kopiere noen form for kode i forbindelse med hjemmeeksamen. Det som kreves er som nevnt at man kan stå inne for det som leveres. Hvis du er i tvil om hva som er lovlig samarbeid, kan du kontakte gruppelærer eller faglærer.

Lykke til!
– Tor, Michael og Pål

Publisert 11. nov. 2015 14:23 - Sist endret 11. nov. 2015 14:50