Koodi tükeldamine

MITME KOODIFAILI KOMPILEERIMINE

Koodi tükeldamine

Suuremate projektidega töötades tuleb kood tükeldada mitmete koodi ja päisefailide vahel

- Sageli räägime sadadest … tuhandetest failidest keskmise suurusega projektides

Iga fail peaks sisaldama sarnase funktsionaalsusega koodi

Mõned eelised

- Parem koodi organiseerimine, hallatavus ja loetavus
- Kiirem kompileerimine (Makefile, CMake kasutusel kompileeritakse vaid uuendatud koodifailid)
- Lihtsam koodi korduvkasutus (teekide loomine)
- Konfliktide vähendamine meeskonnatöös (nt Giti kasutades)

NB! Korraliku sorteerimise aluseks on ka Makefile, mida me täna veel ei käsitle!

Failidest

Iga koodifail (.c) saab endaga samanimelise päise (.h)
- On erandeid, kus päisefaili pole vaja (sageli pannakse ikkagi)

Ühes koodifailis võidakse kasutada mitut enda loodud päisefaili
- Osad päised kirjeldavad ühist funktsionaalsust – nt struktuuride kirjeldused, loendid, konstandid

Sageli jaotatakse failid eraldi kaustadesse
- src – koodifailid
- include – päisefailid
- lib – kolmanda osapoole teegifailid
- build – ehitamisel loodavad ajutised failid (enamjaolt .o failid)
- bin – kompileeritud rakendus(ed)

Suuremates projektides on täiendav organiseerimine kaustadesse moodulite kaupa

Kindlaid standardiseeritud „reegleid“ struktureerimise kohta ei eksisteeri

Kompileerimine

Kasutades Makefile-i (soovitatav!)

- Alternatiiviks on CMake (vajab eraldi paigaldamist)

Kasutades projekte

- Kompileerimise lahendab sinu eest IDE (võib vajada täiendavat seadistamist)
- Vajab keerukamaid IDEsid. Nt Code::Blocks, Visual Studio, Eclipse, CLion jne

Käsitsi kompileerimine (väiksematele projektidele)

- Tavapäraselt kompileeritakse kõik .c failid korraga ühel real
- `gcc file1.c file2.c`
- `gcc *.c`

Töötab vaid juhul kui kaustas puuduvad kõrvalised või konfliktsed koodifailid

Otse koodifaile lisades (**halb praktika, ei kasuta!**)

- `#include "mycode.c"`

Proovi ise järgi

1. Lae alla näidisprojekt, paki see lahti

2. Tutvu projekti struktuuriga
   - Vaata kuidas on .c ja .h failid organiseeritud
   - Vaata mis paikneb nende failide sees
   - Märka ka kommenteerimise stiili erinevust! Päises olevad kommentaarid on funktsiooni kasutajale, koodifailis olevad funktsioonid on arendajale!

3. Kompileeri käsurealt (asenda õigete faili nimedega!)

```
gcc -std=c99 -g -Wall -Wextra -Wconversion -o program_name source1.c source2.c source3.c
```

4. Käivita programm

Võid vaadata ja proovida ka pakutud lihtsat Makefile'i
   - Lihtsustatud Makefile'i koostamisest räägime järgmine nädal
   - Makefile'iga kompileerimiseks kirjuta käsk `make`

Täiendavad teegid

C/C++ jaoks eksisteerib ulatuslik teekide ja tööriistade valik

- Sobiliku teegi võid leida peaaegu ükskõik mille tegemiseks
- Nt https://github.com/oz123/awesome-c

Laialtlevinud teegid on enamasti lihtsalt kättesaadavad paketihalduritest

- Nt Linuxil `apt install libcurl4-openssl-dev`
- Nt Windowsil `choco install curl`

Loe kindlasti juhendmaterjali kuidas kompileerida!

- Tavaliselt vaja kompilaatorile anda 1 – 2 lisaparameetrit (`-I` ja/või `-L`)
- Enamasti pannakse vajalikud lisaparameetrid Makefile'i

Teekide kasutus on oluliselt lihtsam UNIX maailmas (Linux, MacOS). Üldiselt töötab kõik ka Windowsil, kuid seadistus on ebamugavam

cURL and libcurl

cURL-i kasutatakse andmete saatmiseks / vastuvõtuks kasutades erinevaid protokolle

- Failide allalaadimine
- Teenuste testimine
- Internetis olevate APIde kasutamine
- „Sobilike“ ja „Huvitavate“ pakettide saatmine veebilehtedele
- …

libcurl

- cURL kasutamiseks vajaminev teek C programmeerimiskeeles
- https://curl.se/libcurl/
- Laboriarvutitesse juba paigaldatud
- Enda arvutisse paigaldamiseks Debiani-põhisel distributsioonil:
  `sudo apt install libcurl4-openssl-dev`
- Kompileerimiseks vajalik lisalipp UNIXis `-lcurl`