

Unixin ohjelmointityökaluja

Riku Saikkonen

12. helmikuuta 2008

Sisältö

Vapaita (open source) työkaluja:

man, info	Dokumentaatiota
make	Automaattisoi kääntämistä
cpp	Esikäöntäjä
diff, patch	Tiedostojen vertailuun
valgrind	Muistinhallintadebuggeri
Emacs	Tekstieditori
lex, yacc	Jäsenningeneraattorit
ja muutama muu...	

Dokumentaatiota: man

- Unixin C-kirjasto on dokumentoitu manuaalisivuilla ("man pages")
 - komento `man printf`
 - jos näyttää jotain muuta, `man 3 printf` tai `man 2 printf`
 - 2 = systeemikutsuja, 3 = niiden päälle rakennettuja kirjastofunktioita
 - esim. `man 3 printf` näyttää manuaalisivun `printf(3)`
- haku: `apropos printf`
- myös lisäkirjastot on usein dokumentoitu manuaalisivuina
- manuaalisivu kertoo, mihin kirjastoon funktio kuuluu (esim. "ANSI C" tai "ISO 9899" on standardi-C:tä, "POSIX" Unixin kirjastofunktio)

Dokumentaatiota: info

- Monet GNU-työkalut on dokumentoitu ns. info-sivuilla
- `info gdb` tai Emacsista `C-h i m gdb RET`
- Infon paras piirre: `s`-komento hakee koko dokumentista eikä vain tietyltä sivulta
- Infoa voi myös muuttaa HTML:ksi, PS:ksi tai PDF:ksi
- GNU/Linuxesta löytyy myös GNU:n C-kirjaston info-dokumentaatio, `info libc`
 - manuaalisivuja laveampi, myös esimerkkejä
 - jotkut asiat GNU/Linux-spesifisiä
 - myös Webissä:
<http://www.gnu.org/software/libc/manual/>

make

- työkalu, joka kääntää C-ohjelman automaattisesti
- lukee Makefile-tiedostosta käännösohjeet, katsoo tiedostojen päivämääriä, ja kääntää uudelleen kaiken, mikä riippuu jostain muuttuneesta tiedostosta
- Makefilessa voi olla useita käännösvaihtoehtoja, esim. `make all` tai `make foo.o`
- usein myös `make clean`, joka poistaa käännöksen tulokset (ajaa joukon `rm`-komentoja)
- Make toimii myös muilla kielillä ja esim. \LaTeX -dokumenteilla
- isojen ohjelmien Makefilet tehdään usein automaattisesti, esim. ohjelmilla Autoconf ja Automake
- `info make` (ohje GNU Makelle, jossa on enemmän ominaisuuksia kuin perinteisessä Makessa)

make: Esimerkki Makefilestä

```
CC = gcc
CFLAGS = -Wall -g
# tai esim. CFLAGS = -Wall -O2 -g
LDFLAGS =
LIBS =
# LIBS = -lm   jos tarvitaan

TARGET = prog
OBJS = prog.o mysort.o

all: $(TARGET)

clean:
    rm -f $(OBJS) $(TARGET)

$(TARGET): $(OBJS)
    $(CC) $(LDFLAGS) -o $@ $^ $(LIBS)

%.o: %.c
    $(CC) -c $(CFLAGS) $<

.PHONY: all clean
```

make: Ohjeita

- säännön rakenne (sisennetyin rivin alussa Tab!):
kohdetiedosto: tarvittavat tiedostot ...
 komennot
 ...
- edellistä esimerkkiä voi käyttää runkona yksinkertaisissa C-ohjelmissa
- `make -n` kertoo, mitä komentoja `make` ajaisi, samoin `make -n clean`
- `make clean all`
- *varo* clean-säännön `rm`-komentoja (kokeile ensin `make -n`:llä tai älä käytä ollenkaan, äläkä käytä tähtiä tms. tiedostonimissä)
- Lue Maken ohje, jos teet jotain isompaa!
(Ja harkitse Autoconfin ja Automaken käyttöä.)

Esikäntäjä (cpp)

- cpp esikäntää C-ohjelman (tai muun tiedoston) eli käsittelee `#definet`, `#ifdefit` ym.
- tällä voi esimerkiksi tutkia, mitä makromäärittelyistä tulee
- `cpp foo.c >foo.cpp.c`
- tai: `gcc -E foo.c | less`

diff ja patch

- komentorivityökalu diff etsii eroavaisuudet kahdesta tiedostosta
- käyttö yleensä: `diff -u vanha-koodi.c uusi-koodi.c`
- useita tulostusformaatteja:
 - oletuksena vain muuttuneet rivit
 - `diff -c`: myös ympäröiviä muuttumattomia rivejä, vanha ja uusi versio erillään
 - `diff -u`: kuten `-c`, mutta vanha ja uusi versio samassa
- versionhallintatyökaluissa (kuten CVS) diff on yleensä integroituna valmiiksi (esim. `cv diff -u -r1.5`)
- patch ottaa diffin tulosteen ja vanha-koodi.c:n ja tekee niistä uusi-koodi.c:n
 - helppo tapa levittää muutoksia lähdekoodiin
- `man diff`, `man patch`

valgrind

- "muistinhallintadebuggeri"
- ajaa C-ohjelman (hitaasti) ja kertoo, missä kohdissa se:
 - käytti alustamatonta muistia
 - kirjoitti olemattomaan paikkaan pinossa
 - kirjoitti muualle kuin mallocilla varatulle muistialueelle
 - ym.
- mukana myös muita työkaluja, mm. cachegrind kertoo kuinka ohjelma käyttää välimuistia
- <http://www.valgrind.org/>

Emacs 1/3

- Emacs-tekstieditorissa on paljon ohjelmointia helpottavia ominaisuuksia
- `M-x global-font-lock-mode` värittää avainsanoja (pysyväksi editoimalla `~/.emacs:ia` tai `M-x customize:lla`)
- `M-x man` näyttää manuaalisivuja
- GNU/Linuxissa `C-h TAB` näyttää C-kirjaston info-sivuja
- `M-x compile` ajaa Maken tai vastaavan, virheilmoituksista voi hyppiä suoraan oikealle koodiriville
- `M-x gdb` ajaa gdb:n Emacsin sisällä ja näyttää kooditiedostoista, missä ollaan menossa

Emacs 2/3

- `M-x speedbar` näyttää hakemiston tiedostot ja niissä määritellyt funktiot erillisessä ikkunassa (vain graafisessa Emacsissa)
- pelkkä hakemistolistaus: `C-x C-f` ja hakemiston nimi
- `M-x which-function-mode` näyttää missä funktiossa ollaan
- `M-x imenu`:lla voi liikkua tiedoston sisällä funktiosta toiseen
- TAGS on hienempi tapa funktioiden välillä liikkumiseen, ks. `man etags` ja Emacs-komennot `M-.` ja `M-*`

Emacs 3/3

- pikkukomentoja (kokeile...):
 - TAB sisentää, C-j vaihtaa riviä ja sisentää
 - C-M-q funktion alkavan {:n kohdalla sisentää koko funktion uudelleen
 - C-u 2 C-x \$ piilottaa ≥ 2 merkkiä sisennetyt rivit; C-x \$ näyttää taas kaiken
- paljon muutakin...
- useimmat graafisten kehitysympäristöjen toiminnot löytyvät jossain muodossa Emacsistakin
- Emacsin perusteet: C-h t
- lisäohjeita: C-h i m emacs RET, katso erityisesti kohta "Programs"

Jäsentimet

Jäsentimien tekemiseen käytetään hyvin yleisesti näitä:

- lex tai GNU flex: tekee konfiguraation perusteella selaimen (lekserin) eli tuottaa C-ohjelman, joka lukee merkkijonon ja jakaa sen osiin
- yacc tai GNU bison: tekee annetusta kieliopista jäsentimen (parserin) eli tuottaa C-ohjelman, joka lukee selaimen tulostetta ja jäsentää sen jäsennyspuuksi (tai vastaavaksi)
- useimmille muillekin kielille kuin C:lle on vastaavat työkalut
- lisää kurssilla T-106.4200 Johdatus kääntäjäteknikkaan

Muita työkaluja

- Autoconf auttaa hyvin porttautuvan C-koodin kääntämisessä
- Automake on Autoconfin päälle rakennettu kokonaisvaltaisempi Makefilejen rakentaja
- versionhallintaan: Subversion, CVS, darcs, ...
 - lisää myöhemmällä luennolla
- strace kertoo, mitä systeemikutsuja ohjelma tekee
- gprof eli "profiler" kertoo, missä funktioissa ohjelma vietti eniten aikaa (`info gprof`, `man gprof`)
- lint antaa C-koodista samantyyppisiä varoituksia kuin `gcc -Wall`, mutta vähän enemmän (nykyään melko harvinainen)