C-kurssi
kevät 2006

Liisa Marttinen & Tiina Niklander

17.1.2005

Luennon sisältö

- Kurssin rakenne
- C-kielen yleisperiaate
- (tauko)
- Tasokoe (15 min)
- Ohjelmointiprosessi

Kurssin rakenne

- Luennot: ti 16-18
- Verkkokurssimahdollisuus
- Laskuharjoitukset: ke 12-14
- **Harjoitustyö**
- **Kurssikoe**
- Kurssikirja:
  Müldner: C for java programmers

Luennot

- Luento 1 – tämä kerta
- Luento 2 – tyypit, rakenteet, makrot
- Luento 3 – tekstitiedostot
- Luento 4 – funktiot
- Luento 5 – osoittimet
- Luento 6 – tietueet ja joukot
- Periodi TAUKO
- Luento 7 – merkkijonot
- Luento 8 – taulukot
- Luento 9 – moduulit ja kirjastot

Verkkokurssi mahdollisuus

- Kaupallinen ohjelmisto
- http://helsinki.viope.fi/ TARKISTA!!!
- Vain 50 lisenssiä – ne maksavat oikeasti
- Yksinkertainen oppimateriaali, mutta suhteellisen paljon harjoituksia
- Automaattinen harjoitusten tarkistus perustuu tulostuksen ulkoasuun, joten olkaa tarkkoina
- Kurssin suoritukseen saa lisäpisteitä tehtyjen harjoitusten mukaan

Laskuharjoitukset

- Joka viikko ke 12-14 alkaen 8.2.
- Tehtävät tulevat kurssin www-sivulle viimeistään viikkoa ennen
- Lisäpisteitä jaossa

Kurssin tilanne

- Ilmoittautuneita >80
- Yritetään perustaa toinen laskariryhmä
- Verkkokurssin lisenssit eivät riitä kaikille

Harjoitustyö

- Aiheet tulevat 6. viikolla
- Aikaa on tehdä koko 2. periodi
- Laskuharjoitusten päättymisen **jälkeen** on vielä saatavilla säännöllisesti ohjausta harjoitustöiden tekemiseen
- Demotilaisuus luentoaikana luentojen päättymisen jälkeen (luentokerta 10)

Kurssikoe

- Ti 2.5. 16-19 (TARKISTA!)
- Tehtävätyypit
  - Laskarien kaltaisia
  - Tee ohjelma
  - ”Mitä virheitä oheisessa ohjelmassa”
  - Mahdollisesti vielä jotain muitakin muotoja
- Teemat
  - Osoittimet, tiedostot, taulukot, tietueet, merkkijonot, komentoriviparametrit

Luennon sisältö

- Kurssin rakenne
- **C-kielen yleisperiaate**
- (tauko)
- Tasokoe (15 min)
- Ohjelmointiprosessi

C-kielen yleisperiaate:

Ohjelmoija tietää mitä tekee!

- Kieli ei estä ’hölmöilyjä’ – ohjelmoija voi kirjoittaa varsin kryptistä koodia, jos haluaa
- Huolimattomuusvirheiden etsintään kuluu paljon aikaa
- Ei olioita, jotka piilottavat rakenteita
- Osoittimet tärkeä osa kielen käyttöä
- Sopii koneen läheiseen ohjelmointiin, koska tehokas kääntäminen konekielelle osataan
- Esimerkiksi Linux on ohjelmoitu C:llä

Comparison of C and Java

- *primitive data types*: character, integer, and real
  In C, they are of different sizes,
  there is no Unicode 16-bit character set
- *structured data types*: arrays, structures and unions.
  In C, arrays are static
  there are no classes
- *Control structures* are similar
- *Functions* are similar

Comparison of C and Java

- Java references are called pointers in C.
- Java constructs missing in C:
  - packages
  - threads
  - exception handling
  - garbage collection
  - standard Graphical User Interface (GUI)
  - built-in definition of a string
  - standard support for networking
  - support for program safety.

Ohjelmointityyli

- Pyri kirjoittamaan selkeää koodia ja käytä Java-kursseilla opittua tyyliä
- Tiiveys ja kryptisyys ei ole itseisarvo ja sillä ei saa lisäpisteitä

```
do {
  if (scanf("%d", &i) !=1 ||
      i == SENTINEL)
    break;
  if (i>maxi)
    maxi = i;
} while (1);
```

```
void show (char *p) {
  char *q;
  printf("[ ");
  for (q=p; *q != '\0'; q++)
    printf("%c ", *q);
  printf("]\n");
}
```

Tasokoe

- Jaetaan paperilla, vastataan siihen
- 15 minuuttia aikaa (n. 5 min per tehtävä)
- Tavoite:
  - Kartoittaa yleisiä ohjelmointitaitoja
  - Kartoittaa c-kielen hallintaa
  - Saattaa vaikuttaa verkko-osion käyttöön

Luennon sisältö

- Kurssin rakenne
- C-kielen yleisperiaate
- (tauko)
- Tasokoe (15 min)
- **Ohjelmointiprosessi**

Ohjelmointiprosessi

- Ohjelman kirjoittaminen
  - sopiva tekstinkäsittelyohjelma tai editori
- Kääntäminen
  - valitaan oikea kääntäjä
- Linkitys
  - käännetty ohjelmamoduuli yhdistetään muihin
- Suorittaminen
  - valmiin ohjelman suorittaminen

Ohjelman kirjoittaminen

- Käytettävän ohjelman on tuotettava *tavallinen tekstitiedosto*.

```
int main (void)
{
  printf("Hello world \n");
  return 0;
}
```

- Mahdollisia ohjelmia
  - ue: microemacs – toimii komentotulkin sisällä
  - xemacs: aukeaa omaan ikkunaansa
    - Muista käynnistää komentotulkista komennolla xemacs & niin ei komentotulkki jää suotta varatuksi
  - Kate, KEdit, KWrite, Nedit: ainakin nämä tarjolla laitoksen KDE-ympäristössä
- Näiden ohjelmien käyttöä ei kurssilla opeteta

Kääntäminen

- Laitoksen Linux ympäristössä on käytössä gcc (myös komento cc toimii)

```
niklande@wrl-66:~$ which gcc
/usr/bin/gcc
niklande@wrl-66:~$ ls -l /usr/bin/gcc
-rwxr-xr-x  2 root root 93136 Oct 26 11:48 /usr/bin/gcc*
niklande@wrl-66:~$ gcc -dumpversion
3.4.2
```

gcc --help

```
Usage: gcc [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a phase
  - -help                  Display this information
  - -target-help           Display target specific command line options
  (Use '-v --help' to display command line options of sub-processes)
  -dumpspecs               Display all of the built in spec strings
  -dumpversion             Display the version of the compiler
  -dumpmachine             Display the compiler's target processor
  -print-search-dirs       Display the directories in the compiler's search path
  -print-libgcc-file-name  Display the name of the compiler's companion library
  -print-file-name=<lib>   Display the full path to library <lib>
  -print-prog-name=<prog>  Display the full path to compiler component <prog>
  -print-multi-directory   Display the root directory for versions of libgcc
  -print-multi-lib         Display the mapping between command line options and
                           multiple library search directories
  -print-multi-os-directory Display the relative path to OS libraries
  -Wa,<options>            Pass comma-separated <options> on to the assembler
  -Wp,<options>            Pass comma-separated <options> on to the preprocessor
  -Wl,<options>            Pass comma-separated <options> on to the linker
  -Xassembler <arg>        Pass <arg> on to the assembler
  -Xpreprocessor <arg>     Pass <arg> on to the preprocessor
  -Xlinker <arg>           Pass <arg> on to the linker
```

gcc –help (jatkuu)

```
  -save-temps              Do not delete intermediate files
  -pipe                    Use pipes rather than intermediate files
  -time                    Time the execution of each subprocess
  -specs=<file>            Override built-in specs with the contents of <file>
  -std=<standard>          Assume that the input sources are for <standard>
  -B <directory>           Add <directory> to the compiler's search paths
  -b <machine>             Run gcc for target <machine>, if installed
  -V <version>             Run gcc version number <version>, if installed
  -v                       Display the programs invoked by the compiler
  -###                     Like -v but options quoted and commands not executed
  -E                       Preprocess only; do not compile, assemble or link
  -S                       Compile only; do not assemble or link
  -c                       Compile and assemble, but do not link
  -o <file>                Place the output into <file>
  -x <language>            Specify the language of the following input files
                           Permissible languages include: c c++ assembler none
                           'none' means revert to the default behavior of
                           guessing the language based on the file's extension

Options starting with -g, -f, -m, -O, -W, or --param are automatically
 passed on to the various sub-processes invoked by gcc.  In order to pass
 other options on to these processes the -W<letter> options must be used.
```

Käännetään

- Käännetään
  gcc helloworld.c
  tai
  *gcc –o helloworld *
  helloworld.c
- Tässä tehdään
  - esiprosessointi
  - varsinainen käännös ja
  - linkitys

```
int main (void)
{
  printf("Hello world \n");
  return 0;
}
```

- ja näin syntyi suoritettava tiedosto
  a.out
  tai
  helloworld

gcc -v helloworld.c

```
Reading specs from /usr/lib/gcc/i386-redhat-linux/3.4.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --enable-
      shared --enable-threads=posix--disable-checking --with-system-zlib --enable-__cxa_atexit --
      disable-libunwind-exceptions --enable-java-awt=gtk --host=i386-redhat-linux
Thread model: posix
gcc version 3.4.2 20041017 (Red Hat 3.4.2-6.fc3)
 /usr/libexec/gcc/i386-redhat-linux/3.4.2/cc1 -quiet -v helloworld.c -quiet -dumpbase helloworld.c -
      auxbase helloworld -version -o /tmp/niklande/cc1k6oOu.s
ignoring nonexistent directory "/usr/lib/gcc/i386-redhat-linux/3.4.2/../../../../i386-redhat-linux/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /usr/lib/gcc/i386-redhat-linux/3.4.2/include
 /usr/include
End of search list.
GNU C version 3.4.2 20041017 (Red Hat 3.4.2-6.fc3) (i386-redhat-linux)
        compiled by GNU C version 3.4.2 20041017 (Red Hat 3.4.2-6.fc3).
GGC heuristics: --param ggc-min-expand=98 --param ggc-min-heapsize=129136
 as -V -Qy -o /tmp/niklande/ccQshiJR.o /tmp/niklande/cc1k6oOu.s
GNU assembler version 2.15.90.0.3 (i386-redhat-linux) using BFD version 2.15.90.0.3 20040415
 /usr/libexec/gcc/i386-redhat-linux/3.4.2/collect2 --eh-frame-hdr -m elf_i386 -dynamic-linker /lib/ld-
      linux.so.2 /usr/lib/gcc/i386-redhat-linux/3.4.2/../../../crt1.o /usr/lib/gcc/i386-redhat-
      linux/3.4.2/../../../crti.o /usr/lib/gcc/i386-redhat-linux/3.4.2/crtbegin.o -L/usr/lib/gcc/i386-redhat-
      linux/3.4.2 -L/usr/lib/gcc/i386-redhat-linux/3.4.2 -L/usr/lib/gcc/i386-redhat-linux/3.4.2/../../..
      /tmp/niklande/ccQshiJR.o -lgcc --as-needed -lgcc_s --no-as-needed -lc -lgcc --as-needed -lgcc_s -
      -no-as-needed /usr/lib/gcc/i386-redhat-linux/3.4.2/crtend.o /usr/lib/gcc/i386-redhat-
      linux/3.4.2/../../../crtn.o
```

gcc -ansi -pedantic -Wall

- Optioilla –Wall ja –pedantic saa kääntäjän antamaan kaikki mahdolliset varoitukset
- Optio –ansi varmistaa että kääntäjä tekee tulkinnat standardin mukaan

```
gcc -ansi -pedantic -Wall -o helloworld helloworld.c
/home/fs/niklande/C-luennot/esimerkit/helloworld.c: In function `main':
/home/fs/niklande/C-luennot/esimerkit/helloworld.c:3: warning:
implicit declaration of function `printf'
```

```
int main (void)
{
  printf("Hello world \n");
  return 0;
}
```

```
#include <stdio.h>
int main (void)
{
  printf("Hello world \n");
  return 0;
}
```

Ohjelmassa useita moduuleja

- Kukin moduuli, käännösyksikkö, kirjasto omassa tiedostossaan
- Käännetään erikseen
  gcc –c main.c
- Linkitetään yhteen
  gcc –o main.o eka.o toka.o

Ohjelmassa useita moduuleja

```
/* main.c */
#include <stdio.h>
#include ”eka.h”
#include ”toka.h”
int main (void)
{
  eka(); toka ();
  return 0;
}
```

```
/* eka.c */
#include <stdio.h>
#include ”eka.h”
void eka (void)
{
  puts(” eka ”);
}
```

```
/* toka.c */
#include <stdio.h>
#include ”toka.h”
void toka (void)
{
  puts(” toka ”);
}
```

```
/* eka.h */

void eka (void);
```

```
/* toka.h */

void toka (void);
```

```
gcc –c main.c
gcc –c eka.c
gcc –c toka.c
gcc –o ohjelma main.o eka.o toka.o
```

Moduulien kääntäminen – make

- Käsin pitkien käskyjonojen syöttäminen ei ole järkevää
- Käytä siis tiedostoa Makefile
- Suoritettavat komennot ja ohjeet kirjataan säännöiksi tiedostoon

```
kohde: tarvittavat tiedostot
    komento1
    komento2
    ..
    komentoy
```

- Huomaa, että komennot sisennetään tabulaattorimerkillä – EI välilyönnillä!

makefile

```
gcc –c main.c
gcc –c eka.c
gcc –c toka.c
gcc –o ohjelma main.o eka.o toka.o
```

make

- Kirjoita tuo makefile vain kerran
- Käytät sitä useita kertoja

```
# makefile
CC = gcc –ansi –pedantic –Wall
ohjelma: main.o eka.o toka.o
        $(CC) –o ohjelma main.o eka.o toka.o
eka.o: eka.c eka.h
        $(CC) –c eka.c
toka.o: toka.c toka.h
        $(CC) –c toka.c
main.o: main.c eka.h toka.h
        $(CC) –c main.c
```

make --help

```
Usage: make [options] [target] ...
Options:
  -b, -m                      Ignored for compatibility.
  -C DIRECTORY, --directory=DIRECTORY
                              Change to DIRECTORY before doing anything.
  -d                          Print lots of debugging information.
  --debug[=FLAGS]             Print various types of debugging information.
  -e, --environment-overrides
                              Environment variables override makefiles.
  -f FILE, --file=FILE, --makefile=FILE
                              Read FILE as a makefile.
  -h, --help                  Print this message and exit.
  -i, --ignore-errors         Ignore errors from commands.
  -I DIRECTORY, --include-dir=DIRECTORY
                              Search DIRECTORY for included makefiles.
  -j [N], --jobs[=N]          Allow N jobs at once; infinite jobs with no arg.
  -k, --keep-going            Keep going when some targets can't be made.
  -l [N], --load-average[=N], --max-load[=N]
                              Don't start multiple jobs unless load is below N.
```

make --help (jatkuu)

```
  -n, --just-print, --dry-run, --recon  Don't actually run any commands; just print
     them.
  -o FILE, --old-file=FILE, --assume-old=FILE
                              Consider FILE to be very old and don't remake it.
  -p, --print-data-base       Print make's internal database.
  -q, --question              Run no commands; exit status says if up to date.
  -r, --no-builtin-rules      Disable the built-in implicit rules.
  -R, --no-builtin-variables  Disable the built-in variable settings.
  -s, --silent, --quiet       Don't echo commands.
  -S, --no-keep-going, --stop
                              Turns off -k.
  -t, --touch                 Touch targets instead of remaking them.
  -v, --version               Print the version number of make and exit.
  -w, --print-directory       Print the current directory.
  --no-print-directory        Turn off -w, even if it was turned on implicitly.
  -W FILE, --what-if=FILE, --new-file=FILE, --assume-new=FILE
                              Consider FILE to be infinitely new.
  --warn-undefined-variables  Warn when an undefined variable is referenced.
```

Entä käännöksen jälkeen

- Meillä on suorituskelpoinen ohjelma, mutta toimiiko se?
- Kokeillaan ja testataan
- Etsitään virheitä
  - aputulostukset
  - koodin lukeminen ja miettiminen
  - virheenjäljittimen (debuggeri) käyttö
- Analysoidaan testien kattavuutta (ei tällä kurssilla -> Ohjelmistojen testaus)
  - Tällä kurssilla riittää ns. savutestaus (eli ohjelman toiminta vaikuttaa näiden testien jälkeen stabiililta)

Testaus

- Tavoitteena löytää virheitä
- Mahdollisimman erilaisia syötteitä
- Saa automatisoida (esim. skriptien tai varsinaisten testityökalujen avulla)

  ei kuulu tämän kurssin varsinaiseen asiaan
- Tällä kurssilla riittää
  - syötteiden oikeat ja väärät arvot
  - tyypilliset raja-arvot syötteissä (-1,0,1)

Aputulostus

- printf (”Fnimi: Muuttujan nimi %d \n”, muuttuja);
- Pyritään kartoittamaan ohjelman toimintaa virhetilanteessa.
- Sijoitetaan tulostuslauseet todennäköisimmän virhekohdan ympärille
- Usein varsinaista virheenjäljitintä kätevämpi tapa muuttujien arvojen tarkasteluun, kunhan virheen sijainnista on joku käsitys etukäteen

Virheenjäljitin gdb

(gdb) help

List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

core dump

- Kaatunut ohjelman tuottaa usein tiedoston, jossa on muistin ja rekisterin tila ohjelman kaatumishetkellä (ns. core dump)
- Näitä voi tarkastella esim. virheenjäljittimellä, jolloin saattaa olla mahdollista katsella muuttujien arvoja ja/tai selvittää missä käskyssä ohjelma oli kaatuessaan.
- *Tämän opiskeleminen jää kotitehtäväksi*