universität wien

STL, homework 2

due 15.10.2018, 08:00

- **Prequisites**
  - Basics
    - your programs must have been written and tested on Harris
    - use subdirectories $HOME/exercises/hw2_1, $HOME/exercises/hw2_2, $HOME/exercises/hw2_3
    - write your codes in C, use the GNU C compiler gcc
    - no global variables allowed
    - entries for arrays can be “hard coded”, output can be simple
    - your solutions should be as portable as possible
  - Building
    - include the following three arguments to compile:
      - -Wall -std=iso9899:2011 -pedantic
    - no warnings should be issued by the compiler
    - use Makefiles
      - one for each exercise
      - no arguments (simply “make” should be sufficient for building)
      - compilation and linking must be separate
  - Submitting
    - your archive should include the following files: Makefiles, source code files, gnuplot data files (where applicable), figure in pdf (where applicable), output in pdf (copy&paste)
    - do not include object files, executables, libraries
    - create a single gzipped tar file of the following form:
      - <last name>_<matriculation number>.tar.gz
    - upload only this single archive file to Moodle
  - Presentation
    - you must be ready to present and discuss your submission
    - the files on Harris must match with your submission

- Links
  - gnuplot: http://www.gnuplot.info/
  - Valgrind: http://valgrind.org/
  - gdb: https://www.gnu.org/software/gdb/

- **Exercises**
  - 1) Use gnuplot to generate a figure which looks simular to the one on slide 13 of lecture 1, i.e., (Valle et al., 2014); create the data files yourself (one data file should correspond to one line in the figure); the output file must be in pdf, a Makefile must generate the pdf.
    - include the following files in your submission: Makefile to generate a figure in pdf, gnuplot script file, gnuplot data files, figure in pdf; **no C program is required here**.
  - 2) Use Valgrind to demonstrate two different defects (e.g., usage of uninitialized variables, memory leaks, access outside of arrays) of a program which you debug; note that you must switch off optimization and build with debug symbols enabled.
    - include the following files in your submission: C source, Makefile, output in pdf.
  - 3) Use the GNU debuggger gdb to demonstrate simple debugging; write an arbitrary C program which uses at least one variable and a function; demonstrate in gdb how to i) enter a breakpoint and execute to the breakpoint, ii) print the value of a variable, iii) step into a function and execute the whole function.
    - include the following files in your submission: C source, Makefile, output in pdf.