Operating Systems

(Lab - Unit 2)

Witold Rakoczy, [rakoczy@agh.edu.pl](mailto:rakoczy@agh.edu.pl) D17, room 3.10

* Literature
  + <https://gcc.gnu.org/onlinedocs/gcc/index.html>

The source of all information (but too much for start ;-) )

* + <https://linux.die.net/man/> This is browser version of man pages:
    - User commands (1)
    - **System calls (2)** [may be needed in some exercises…]
    - **Library functions (3)** [important - description of libraries for C lang, including Standard C

Library]

* + - **Administration and privileged commands (8)**
      * may be useful - you are priviledged users!
    - **Math library functions (L)**
    - Other [special files, file formats, games, conventions, etc.]
    - Also there is an alphabetical index of all commands and functions
* <https://gcc.gnu.org/onlinedocs/gcc/Invoking-GCC.html#Invoking-GCC>
* W. Richard Stevens, Stephen A. Rago, "Advanced programming in the Unix Environment", Addison Wesley Publishing Company,
  + <https://images.lab.ii.agh.edu.pl/rakoczy/>
  + There are two versions: *.zip* and *.7z*
* We will use gcc compiler (consult ***man 1 gcc*** for list of options)
* Compiler options allow to control the way
  + the source program is interpreted,
  + how warning and error messages are produced,
  + object code generated and optimized,
  + where and how resulting binary code is written
* We will always use some options to avoid some non-standard extensions to be accepted
* Some other options we will use selectively, when it is useful
  + at first compilations when we want to catch all mistakes quickly, or
  + when we want to debug our program
* General format of the command invoking gcc
  + ***gcc [options] inputfiles -o outputfile***
    - ***gcc -Wall -Werror -ansi -pedantic hello.c -o hello.exe***
* Most inportant options
  + ***-Wall*** enable all compilation warnings
  + ***-ansi*** enable ANSI C compatibility
  + ***-pedantic*** (opt) disable all non-standard C extensions
  + ***-Werror*** convert warnings into errors
  + ***-g*** prepare code suitable for debugging purposes
  + ***-o execfilename*** name executable output file *execfilename*
  + ***-lLibraryName***
  + ***-c*** don't attempt to create executable file
* Optimisation flags allow to set up needed code optimization level
  + *-O*, *-O1* Optimize.
    - Optimizing compilation takes somewhat more time, and a lot more memory for a large function.
  + *-O2* Optimize even more (all -O1 optimization included)
    - GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.
  + *-O3* Optimize yet more.
    - It turns on all optimizations specified by -O2 and also turns on additional optimization
  + *-Og* Optimize for debugging
    - Turns on optimizations that don't interfere with debugging - for debugging otherwise slow programs.
* Make is usefull tool used to automatically perform routine activities in Linux environment
  + Mainly used in simple developer's tasks
* Make uses configuration file located in project directory named *Makefile*
  + *Makefile* contains at least one task definition but may contain many named tasks (*targets*), which can be executed selectively when make is invoked with a parameter
* Syntax used in Makefile

*all:*

*command options parameters*

*targetname:*

*command1 ….*

* To execute Makefile primary target invoke make without parameters
* To execute Makefile named *targetname* invoke make with the target name as parameter
* Example (assuming the program source file is named hello.c)

*all:*

*gcc -Wall -Werror -ansi -pedantic hello.c -o hello*

*clean:*

*rm -rf \*.o hello*

* Create working directory and enter it
* Download files *ex1open.c, t.c* and *t.h*
* Compile the program in one step using both .c files
* Execute the program using any file as input
  + Specify the input file name as program parameter
* Create *ex2open.c* program from the previous one
* Modify algorithm to read whole content of input file counting characters
* Compile the program and execute it using any file as input
  + Specify the input file name as program parameter
* Try to invoke the *ex2open* program without any parameters then with a name of nonexisting file as parameter
  + What's the reason of such behawior?
* Modify the pro *ex2open.c* program to avoid breaking in case of incorrect parameters and with printing for each case an appropriate error message
* Repeat the above tests
* Create Makefile for the program
  + Try exactly the same command as used previously
* Add to Makefile additional target named *clean* allowing a cleanup operation (only original source files should left after cleanup)
* Test *clean* operation and primary operation
  + Check contents for the directory after cleanup and then after subsequent making the program
* Create modified procedure for separate compilation of modules of the program from the previous exercise
* Create Makefile for this procedure and test it