Exercise sheet 2 – Build C programs

Goals:

- Build on command line
- ELF structure and commands
- Makefiles usage
- Autotools usage

Update the `OS_exercises` repository with `git pull`

```
cd ~/Desktop/OS_exercises
git pull
```

Exercise 2.1: Build and run on command line

(a) Go to `OS_exercises/sheet_02_build/1_hello_world`

Proposal for solution: `cd OS_exercises/sheet_02_build/1_hello_world`

(b) Build `main.c`

Proposal for solution: `gcc -o hello_world main.c`

(c) Run the program

Proposal for solution: `./hello_world`

Exercise 2.2: Explore ELF file

Use your compiled program from the above exercise for this.

(a) Find all strings that are contained in the ELF.

Proposal for solution: `strings hello_world`

(b) Which sections has the ELF?

Proposal for solution: `readelf -S hello_world`

```
                Section Headers:
[Nr] Name              Type             Address           Offset
     Size              EntSize          Flags  Link  Info  Align
[ 0]                   NULL             0000000000000000  00000000
     0000000000000000  0000000000000000           0     0     0
[ 1] .interp           PROGBITS         0000000000000238  00000238
     000000000000001c  0000000000000000   A       0     0     1
[ 2] .note.ABI-tag     NOTE             0000000000000254  00000254
     0000000000000020  0000000000000000   A       0     0     4
[ 3] .note.gnu.build-i NOTE             0000000000000274  00000274
     0000000000000024  0000000000000000   A       0     0     4
[ 4] .gnu.hash         GNU_HASH         0000000000000298  00000298
     000000000000001c  0000000000000000   A       5     0     8
[ 5] .dynsym           DYNSYM           00000000000002b8  000002b8
     00000000000000a8  0000000000000018   A       6     1     8
[ 6] .dynstr           STRTAB           0000000000000360  00000360
     0000000000000082  0000000000000000   A       0     0     1
[ 7] .gnu.version      VERSYM           00000000000003e2  000003e2
     000000000000000e  0000000000000002   A       5     0     2
[ 8] .gnu.version_r    VERNEED          00000000000003f0  000003f0
     0000000000000020  0000000000000000   A       6     1     8
[ 9] .rela.dyn         RELA             0000000000000410  00000410
     00000000000000c0  0000000000000018   A       5     0     8
[10] .rela.plt         RELA             00000000000004d0  000004d0
```

```
      0000000000000018  0000000000000018  AI       5     22     8
 [11] .init             PROGBITS         00000000000004e8  000004e8
      0000000000000017  0000000000000000  AX       0     0     4
 [12] .plt              PROGBITS         0000000000000500  00000500
      0000000000000020  0000000000000010  AX       0     0     16
 [13] .plt.got          PROGBITS         0000000000000520  00000520
      0000000000000008  0000000000000008  AX       0     0     8
 [14] .text             PROGBITS         0000000000000530  00000530
      00000000000001a2  0000000000000000  AX       0     0     16
 [15] .fini             PROGBITS         00000000000006d4  000006d4
      0000000000000009  0000000000000000  AX       0     0     4
 [16] .rodata           PROGBITS         00000000000006e0  000006e0
      0000000000000010  0000000000000000   A       0     0     4
 [17] .eh_frame_hdr     PROGBITS         00000000000006f0  000006f0
      000000000000003c  0000000000000000   A       0     0     4
 [18] .eh_frame         PROGBITS         0000000000000730  00000730
      0000000000000108  0000000000000000   A       0     0     8
 [19] .init_array       INIT_ARRAY       0000000000200db8  00000db8
      0000000000000008  0000000000000008  WA       0     0     8
 [20] .fini_array       FINI_ARRAY       0000000000200dc0  00000dc0
      0000000000000008  0000000000000008  WA       0     0     8
 [21] .dynamic          DYNAMIC          0000000000200dc8  00000dc8
      00000000000001f0  0000000000000010  WA       6     0     8
 [22] .got              PROGBITS         0000000000200fb8  00000fb8
      0000000000000048  0000000000000008  WA       0     0     8
 [23] .data             PROGBITS         0000000000201000  00001000
      0000000000000010  0000000000000000  WA       0     0     8
 [24] .bss              NOBITS           0000000000201010  00001010
      0000000000000008  0000000000000000  WA       0     0     1
 [25] .comment          PROGBITS         0000000000000000  00001010
      0000000000000024  0000000000000001  MS       0     0     1
 [26] .symtab           SYMTAB           0000000000000000  00001038
      00000000000005e8  0000000000000018          27    43     8
 [27] .strtab           STRTAB           0000000000000000  00001620
      0000000000000202  0000000000000000           0     0     1
 [28] .shstrtab         STRTAB           0000000000000000  00001822
      00000000000000fe  0000000000000000           0     0     1
```

(c) Which symbols are defined?

Proposal for solution: `nm hello_world` Defined symbols:

```
0000000000201010 B __bss_start
0000000000201010 b completed.7696
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000201000 D __data_start
0000000000201000 W data_start
0000000000000560 t deregister_tm_clones
00000000000005f0 t __do_global_dtors_aux
0000000000200dc0 t __do_global_dtors_aux_fini_array_entry
0000000000201008 D __dso_handle
0000000000200dc8 d _DYNAMIC
0000000000201010 D _edata
0000000000201018 B _end
00000000000006d4 T _fini
0000000000000630 t frame_dummy
0000000000200db8 t __frame_dummy_init_array_entry
0000000000000834 r __FRAME_END__
0000000000200fb8 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
00000000000006f0 r __GNU_EH_FRAME_HDR
00000000000004e8 T _init
0000000000200dc0 t __init_array_end
0000000000200db8 t __init_array_start
00000000000006e0 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
00000000000006d0 T __libc_csu_fini
0000000000000660 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
000000000000063a T main
                 U puts@@GLIBC_2.2.5
00000000000005a0 t register_tm_clones
0000000000000530 T _start
0000000000201010 D __TMC_END__
```

(d) Determine the size of the program.

Proposal for solution: `ls -lh hello_world` Size is about 8.1 KiB.

(e) Strip the symbols of the program.

Proposal for solution: `strip hello_world`

(f) Try to list the symbols again.

Proposal for solution: There are no symbols left that can be listed.

(g) Determine the size of the program again? Has something changed?

Proposal for solution: `ls -lh hello_world` Size is now about 6.0 KiB.

Exercise 2.3: Build with a Makefile

(a) Go to `OS_exercises/sheet_02_build/2_simple_prog`

Proposal for solution: `cd OS_exercises/sheet_02_build/2_simple_prog`

(b) Build the program

Proposal for solution: `make`

(c) Run the program

Proposal for solution: `./simple_prog`

(d) Clean the object files

Proposal for solution: `make clean`

(e) Can you easily install the program into the system?

Proposal for solution: No, you have to move the file into the `/usr/bin` directory (or equivalent)

Exercise 2.4: Build with Autotools

(a) Go to `OS_exercises/sheet_02_build/3_simple_prog_automake`

Proposal for solution: `cd ../3_simple_prog_automake`

(b) Initialise the build system

Proposal for solution: `autoreconf -i`

(c) Configure the build

Proposal for solution: `./configure`

(d) Explore `configure.ac`

Proposal for solution: `less configure.ac`

(e) Explore `Makefile.am`

Proposal for solution: `less Makefile.am`

(f) Build the program

Proposal for solution: `make`

(g) Explore the automatically created `Makefile`.

Proposal for solution: `cat Makefile`

(h) Install the program

Proposal for solution: `sudo make install`

(i) Run the program

Proposal for solution: `simple_prog`

(j) Set the `-DUSE_SPECIAL_ADD` define

Proposal for solution: Add the line `simple_prog_CFLAGS = -DUSE_SPECIAL_ADD` into `Makefile.am`

(k) Create a manpage (you may do a test view directly on the created file)

Proposal for solution: Create the file `simple_prog.1` with the content:

```
.\" Manpage for simple_prog
.\" Contact florian.kuenzner@fh-rosenheim.de to correct errors
.TH man 7 "14 Septenber 2018" "1.0" "simple_prog man page"
.SH NAME
simple_prog \- do something useful
.SH SYNOPSIS
simple_prog
.SH DESCRIPTION
simple_prog is a program that does something useful.
.SH OPTIONS
The simple_prog does not take any options.
.SH BUGS
No known bugs.
.SH AUTHOR
Florian Künzner (florian.kuenzner@fh-rosenheim.de)
```

A test view (without installation) is possible with: `man ./simple_prog.1`

(l) Include the manpage into the build

Proposal for solution: Add the line `man_MANS = simple_prog.1` into `Makefile.am`

(m) Build, install, and run again.

Proposal for solution:

```
make
sudo make install
simple_prog
```

(n) Can you use `man simple_prog`?

Proposal for solution: `man simple_prog` Yes, the manpage is working.