SMP_C cursus
studiejaar 2020-2021 blok 4

docent : E.Koenen

In les 2 van deze cursus zijn verschillende stappen in het compileer proces besproken.
Ook is de assembly code hier besproken.
We gaan deze onderwerpen nader bestuderen met behulp van enkel standaard programma's.

Het gereedschap van deze les is :

- **hd, head, tail**
  enkele handige command-line utility's.
- **readelf**
  een utility om een ELF executable mee te analyseren.
- **gdb**
  de gnu debugger.
  Eigenlijk een command-line utility voor het debuggen van code.
  Maar het kan ook gebruikt worden voor het analyseren/'disassemble'n van executable-code.

Handige Utilities

Dit zijn allemaal console utility's.

HD

HexDump toont een file in hex formaat op het console.
De wijze waarop de informatie wordt getoond kan m.b.v. opties worden aangepast.
Standaard is dit 16 bytes per regel met hex en ASCII weergave.

Zie ook : **man hd**

HEAD

Toont het begin van een file. Standaard de 1e 10 regels. Maar het aantal regels (of het aantal bytes) kan worden beïnvloed mbv opties.
Het is ook mogelijk de laatste n regels (of bytes) te tonen.

Combinatie met het HD commando geeft de info in hex notatie :

```
hd opg | head -n 1
```

Zie ook : **man head**

TAIL

Lijkt veel op **head**. Maar nu werden de laatste regels getoond. Standaard de laatste 10.

Met een belangrijke extra optie : **`-f`** voor follow.
Als data aan de file wordt toegevoegd dan wordt dit automatisch getoond. (Handig bij log-files !)

Zie ook : **man tail**

Readelf

Met behulp van dit programma kunnen we alle details van een ELF file opvragen.
Zie : **man readelf** voor alle opties
Zie ook : **man elf** voor de 'code-structuur' van de ELF onderdelen!

De header informatie van een ELF executable :

```
$ readelf -h helloworld
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x530
  Start of program headers:          64 (bytes into file)
  Start of section headers:          6448 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

belangrijkste opties :

```
-h    header informatie
-l    program headers info
-S    section header info [ hoofdletter S ! ]
-s    symbol info
```

GDB

De gnu debugger. Een commandline tool voor het 'stap voor stap' controleren van een executable.

> Voor een volledig overzicht van gdb zie : https://www.gnu.org/software/gdb/documentation/

Zie : **man gdb** voor de opties.

Een handig gebruik is de batch modus waarbij we een opgegeven file / methode laten disassemblen.
Vb : het executable bestand van het onderstaande c-programma.

```
#include <stdio.h>

int main(){
  for (int i = 0; i < 10; i++) {
        printf("loop var : i = %d\n",i);
      }
  return 0;
}
```

na compileren willen we de assembly code van de main methode zien.
Hiervoor gebruiken we **gdb** met de **-batch** optie gevolgd door 2x een **-ex** ['execute'] optie.

1. voor het inlezen van het bestand.
2. voor het disassemble van de main methode.

```
$ gdb -batch -ex "file opg" -ex "disassemble/rs main"
Dump of assembler code for function main:
   0x000000000000064a <+0>:   55                    push   rbp
   0x000000000000064b <+1>:   48 89 e5              mov    rbp,rsp
   0x000000000000064e <+4>:   48 83 ec 10           sub    rsp,0x10
   0x0000000000000652 <+8>:   c7 45 fc 00 00 00 00  mov    DWORD PTR [rbp-0x4],0x0
   0x0000000000000659 <+15>:  eb 1a                 jmp    0x675 <main+43>
   0x000000000000065b <+17>:  8b 45 fc              mov    eax,DWORD PTR [rbp-0x4]
   0x000000000000065e <+20>:  89 c6                 mov    esi,eax
   0x0000000000000660 <+22>:  48 8d 3d ad 00 00 00  lea    rdi,[rip+0xad]        # 0x714
   0x0000000000000667 <+29>:  b8 00 00 00 00        mov    eax,0x0
   0x000000000000066c <+34>:  e8 af fe ff ff        call   0x520 <printf@plt>
   0x0000000000000671 <+39>:  83 45 fc 01           add    DWORD PTR [rbp-0x4],0x1
   0x0000000000000675 <+43>:  83 7d fc 09           cmp    DWORD PTR [rbp-0x4],0x9
   0x0000000000000679 <+47>:  7e e0                 jle    0x65b <main+17>
   0x000000000000067b <+49>:  b8 00 00 00 00        mov    eax,0x0
   0x0000000000000680 <+54>:  c9                    leave
   0x0000000000000681 <+55>:  c3                    ret
End of assembler dump.
```

(hier is de layout iets aangepast.)

1. de offset vanaf het begin van het bestand
2. de offset vanaf begin methode
3. de binaire instructies
4. de assembly opcode
5. de parameters