SMP_C cursus
studiejaar 2020-2021 blok 4

docent : E.Koenen

In les 1 van deze cursus zijn verschillende typen programmeertalen besproken. Ook is het proces van compileren en linken besproken.
We gaan deze onderwerpen nader bestuderen met behulp van enkel standaard programma's.

Het gereedschap van deze les is :

- **gcc**
  de gnu c-compiler
- **file**
  een standaard utility op het linux platform waarmee file informatie kan worden opgevraagd. (wat zegt een extensie nu eigenlijk …)
- **strip**
  een utility om de *symbol table* van een executable aan te passen.
- **strings**
  een (zeer eenvoudig) programma dat alle strings in een bestand toont.
- **een hexeditor**
  er zijn vele verschillende hexeditors.

Gcc

De gnu c-compiler is een zeer uitgebreid en complex programma. Het is een command-line programma en kent een (zeer) groot aantal parameters. Gelukkig zijn slechts een beperkt aantal hiervan voor ons (nu) van belang.

We gebruiken het volgende c-source bestand voor de tests.

```
1.      #include <stdio.h>
2.      int main(){
3.        printf("hello world\n");
4.        return 0;
5.      }
```

helloworld.c

Met het volgende commando kunnen we deze source code compileren :

```
gcc -Wall -o helloworld helloworld.c
```

toelichting :

-Wall	geeft aan dat alle warnings moeten worden getoond,
-o helloworld	geeft aan dat het output bestand de naam **helloworld** moet hebben als je dit weglaat wordt de output naar **a.out** geschreven
helloworld.c	dit bestand is het bronbestand.

Het resultaat is een executable bestand met de naam '**helloworld**'.
We kunnen dit programma op de standaard manier runnen, met de verwachte string als output :

```
$ ./helloworld
hello world
```

Er zijn zeer veel command-line opties voor gcc. Zie 'man gcc' voor een complete lijst.
Er zijn opties om het 'tussenresultaat' per compile stap te zien :

optie -E :

stop na de preproces stap (output naar stdout , je kunt dit natuurlijk 'redirecten')

je ziet hier alle functie-definities en globale variabelen uit alle include files, met aan het einde de source code.

```
gcc -E helloworld.c
```

optie -c :

compileer maar niet linken. Resultaat is een '*.o' bestand (object file)

```
gcc -c helloworld
```

optie -S :

stopt na een 'volledige' compilatie, maar voor de assembly stap.

Het resultaat is een assembler code file : ***.s** .

De assembly code is voorzien van labels en 'speciale' instructies voor de assembly stap.

Het verschilt daarom van de code die we krijgen als we een executable 'disassemblen' !

Notatie is in AT&T stijl. Met de extra optie **-masm=intel** krijg je de output in intel notatie.

```
gcc -S helloworld.c
```

File

We kunnen dit commando gebruiken om informatie van een file op te vragen.
Het kijkt naar de inhoud van de file en , indien aanwezig , naar het '**magic number**'.

Zie ook '**man file**'.

```
$ file helloworld*
helloworld:   ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
       dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
       GNU/Linux 3.2.0, BuildID[sha1]=b92bd55cb6fccd2f36260e27beccbf0d992a3c33,
       not stripped
helloworld.c: C source, ASCII text
helloworld.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
       not stripped
helloworld.s: assembler source, ASCII text
```

we zien hier dan de executable en het object file beide een **ELF 64-bit** structuur hebben.
De helloworld.c en helloworld.s file zijn beide **ASCII** files.

Strip

Met dit commando kunnen we de informatie in de *string table* aanpassen.

Zie ook '**man strip**'.

```
$ strip -s helloworld
$ file helloworld
helloworld: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux
3.2.0, BuildID[sha1]=b837de2e6245dbebe0d3f02115a1fec553654dfa, stripped
```

Het commando **strip** geeft geen output op de commandline.
strip geeft de mogelijkheid om bepaalde strings te bewaren. Gebruik dan de **-K 'naam'** optie.

Je kunt de bestaande symbols ook vinden m.b.v. het commando **strings**.
(Zie ook utility : readelf)

Strings

Een simpel programma dat een bestand onderzoekt op reeksen 'printable character's.
Elke reeks van voldoende lengte (default = 4) wordt getoond.

Definitie : **printable character**

> *elk byte met een waarde tussen de 0x20 (=spatie) en 0x7E (=tilde) inclusief.*
>
> De 1e 32 tekens in de ASCII tabel (control characters : 0x00 – 0x1F) evenals DEL en alle 'hogere characters' (0x7F – 0xFF) zijn dus **niet** printable characters !

Met behulp van het strings commando kunnen we alle strings in een executable bestand vinden.
Dit zijn teksten die aan de gebruiker getoond worden maar ook de namen van functies.
Handig bij een analyse !

```
$ strings helloworld | grep -v ^[._]
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
GLIBC_2.2.5
AWAVI
AUATL
[]A\A]A^A_
hello world
;*3$"
GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
crtstuff.c
deregister_tm_clones
completed.7698
frame_dummy
helloworld.c
puts@@GLIBC_2.2.5
main
```

We zien hier onder andere de tekst 'hello world' en de naam van de main routine 'main'.

Opmerking :

> de utility **grep** is handig om de output van dit (en elk ander commando) te filteren.
>
> Dit commando `grep -v ^[._]` filtert alle regel die **niet** (`-v`) op de **1e positie** (`^`) een van de tekens **punt** of **underscore** (`[._]`) hebben.
>
> Zie : **man grep**.

Hex-editor

Een hex-editor is handig om een binair bestand te inspecteren.
Meestal wordt elke byte in het bestand op 2 manieren getoond :

1. in hexadecimale notatie : dus 2 hexcodes per byte
2. als het een 'printable character' betreft wordt ook de 'ascii' weergave van de byte getoond.

Elk linux systeem kent een command-line hex-editor.

hexedit

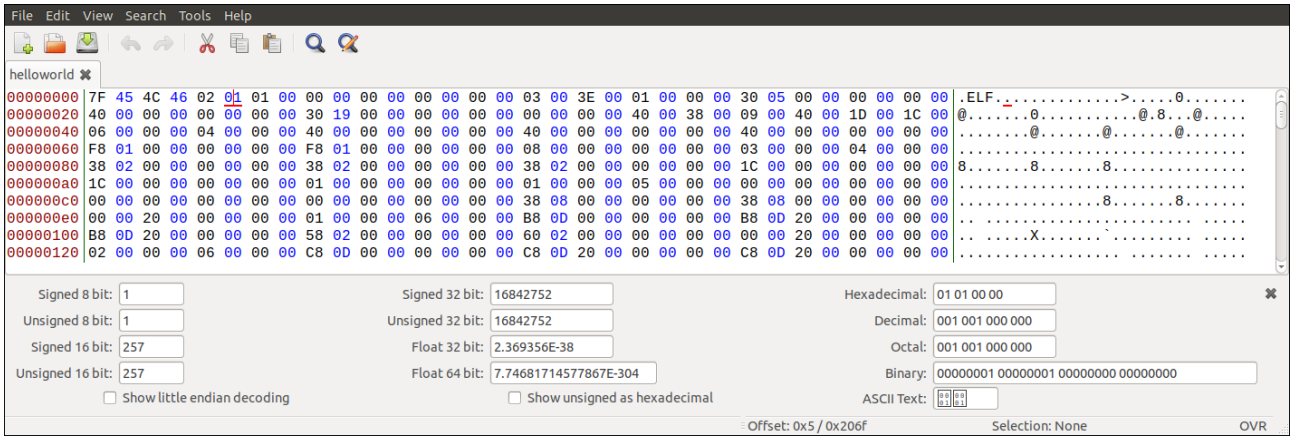
```
$ hexedit helloworld
```

De interface is 'lastig' ! Gebruik dit om snel een overzicht te krijgen van de inhoud van een bestand.
Zie : man hexedit.

Er zijn meerdere hex editors met een grafische interface beschikbaar. (Bless, Ghex, Okteta, en meer …) In deze cursus gebruiken we de Bless hex editor in alle voorbeelden.

Bless

```
$ Bless helloworld
```



Bless – helloworld

we zien hier de beide weergaven voor elke byte.
In de '**statusbar**' (onderste regel) zien we o.a. de offset van het 'huidige' byte (= 0x5) en de schrijf-modus, overwrite of insert (= OVR)
Als we bytes willen aanpassen, dan moet dit ingesteld zijn op OVR !