Linux C編譯
2021 03 06

https://github.com/MyDearGreatTeacher/2021_1_courses/tree/main/CTF%20%E6%90%B6%E6%97%97%E5%A4%A7%E8%B3%BD%E5%AF%A6%E5%8B%99%E6%8A%80%E8%A1%93

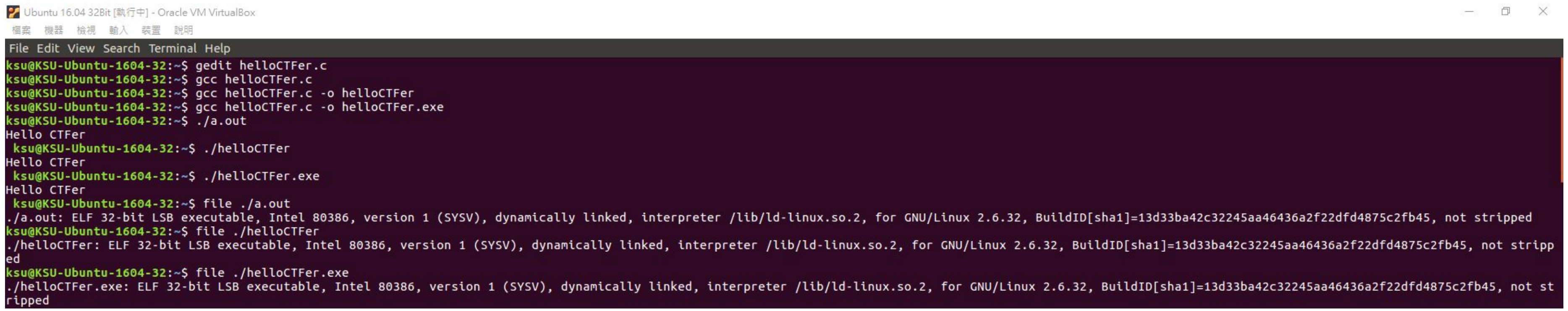
Linux C 程式的編譯與運行:

範例

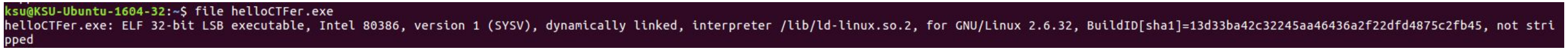
```
// helloCTFer.c
#include <stdio.h>

int main()
{
   printf("Hello CTFer\n ”);
   return 0;
}
```

Unumtu 16.04 LTS(32 bits)

```
(1)編譯
   ==> gcc helloCTFer.c  ==>  產生a.out執行檔
   ==> gcc helloCTFer.c -o helloCTFer
   ==> gcc helloCTFer.c -o helloCTFer.exe

(2)執行
    ==> ./a.out
    ==> ./helloCTFer
    ==> ./helloCTFer.exe

 (3)檢查執行檔檔案格式
    ==> file ./a.out
    ==> file ./helloCTFer
    ==> file ./helloCTFer.exe
```


Ubuntu 16.04 32Bit [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
File Edit View Search Terminal Help

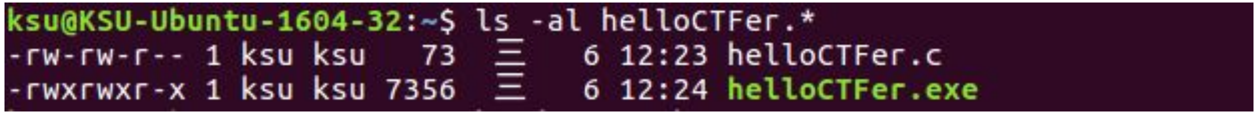
```
ksu@KSU-Ubuntu-1604-32:~$ gedit helloCTFer.c
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c -o helloCTFer
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c -o helloCTFer.exe
ksu@KSU-Ubuntu-1604-32:~$ ./a.out
Hello CTFer
 ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer
Hello CTFer
 ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer.exe
Hello CTFer
 ksu@KSU-Ubuntu-1604-32:~$ file ./a.out
./a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
ksu@KSU-Ubuntu-1604-32:~$ file ./helloCTFer
./helloCTFer: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripp
ed
ksu@KSU-Ubuntu-1604-32:~$ file ./helloCTFer.exe
./helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not st
ripped
```



```
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.exe
helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stri
pped
```

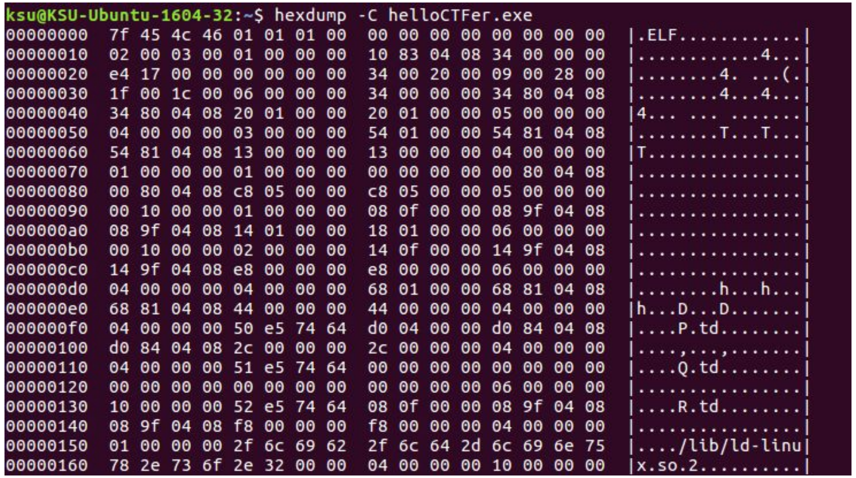
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.exe
helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped

```
ls -al  helloCTFer.*
```



```
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu   73  三   6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu 7356  三   6 12:24 helloCTFer.exe
```

```
hexdump -C helloCTFer.exe

00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  10 83 04 08 34 00 00 00  |............4...|
00000020  e0 17 00 00 00 00 00 00  34 00 20 00 09 00 28 00  |........4. ...(.|
00000030  1f 00 1c 00 06 00 00 00  34 00 00 00 34 80 04 08  |........4...4...|
00000040  34 80 04 08 20 01 00 00  20 01 00 00 05 00 00 00  |4... ... .......|
00000050  04 00 00 00 03 00 00 00  54 01 00 00 54 81 04 08  |........T...T...|
00000060  54 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |T...............|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 c4 05 00 00  c4 05 00 00 05 00 00 00  |................|

....................
```



```
ksu@KSU-Ubuntu-1604-32:~$ hexdump -C helloCTFer.exe
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  10 83 04 08 34 00 00 00  |............4...|
00000020  e4 17 00 00 00 00 00 00  34 00 20 00 09 00 28 00  |........4. ...(.|
00000030  1f 00 1c 00 06 00 00 00  34 00 00 00 34 80 04 08  |........4...4...|
00000040  34 80 04 08 20 01 00 00  20 01 00 00 05 00 00 00  |4... ... .......|
00000050  04 00 00 00 03 00 00 00  54 01 00 00 54 81 04 08  |........T...T...|
00000060  54 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |T...............|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 c8 05 00 00  c8 05 00 00 05 00 00 00  |................|
00000090  00 10 00 00 01 00 00 00  08 0f 00 00 08 9f 04 08  |................|
000000a0  08 9f 04 08 14 01 00 00  18 01 00 00 06 00 00 00  |................|
000000b0  00 10 00 00 02 00 00 00  14 0f 00 00 14 9f 04 08  |................|
000000c0  14 9f 04 08 e8 00 00 00  e8 00 00 00 06 00 00 00  |................|
000000d0  04 00 00 00 04 00 00 00  68 01 00 00 68 81 04 08  |........h...h...|
000000e0  68 81 04 08 44 00 00 00  44 00 00 00 04 00 00 00  |h...D...D.......|
000000f0  04 00 00 00 50 e5 74 64  d0 04 00 00 d0 84 04 08  |....P.td........|
00000100  d0 84 04 08 2c 00 00 00  2c 00 00 00 04 00 00 00  |....,...,.......|
00000110  04 00 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td........|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000130  10 00 00 00 52 e5 74 64  08 0f 00 00 08 9f 04 08  |....R.td........|
00000140  08 9f 04 08 f8 00 00 00  f8 00 00 00 04 00 00 00  |................|
00000150  01 00 00 00 2f 6c 69 62  2f 6c 64 2d 6c 69 6e 75  |..../lib/ld-linu|
00000160  78 2e 73 6f 2e 32 00 00  04 00 00 00 10 00 00 00  |x.so.2..........|
```

ELF File magic

```
ELF ==> 可執行與可鏈接格式 （英語：Executable and Linkable Format，縮寫為ELF），常被稱為ELF格式，
        在電腦科學中，是一种用於執行檔、目的檔、共享库和核心转储(core dump)的标准檔案格式。

ELF defines the binary format of executable files used by Linux.
When you invoke an executable, the OS must know how to load the executable into memory properly,
how to resolve dynamic library dependencies and then where to jump into
the loaded executable to start executing it.
The ELF format provides this information. ELF magic is used to identify ELF files
and is merely the very first few bytes of a file:

ELF Magic ==> "Magic numbers" is the name given to constant sequences of bytes (usually)
               at the beginning of files, used to mark those files as
               being of a particular file format.
               They serve a similar purpose to file extensions.

What is ELF Magic?
https://unix.stackexchange.com/questions/153352/what-is-elf-magic

ELF File magic ==> 7f 45 4c 46 01 01 01 00
```

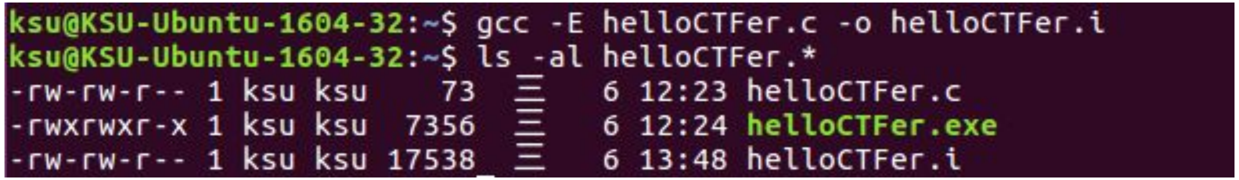
Linux C 程式的編譯與運行:

```
編譯的各階段
 原始程式碼 ===>  helloCTFer.c
 預處理 ===> gcc -E helloCTFer.c -o helloCTFer.i    ===> 查看.i的架構
 編譯 ===> gcc -S helloCTFer.i  -o helloCTFer.s   ===> 查看.s的架構
 彙編 ===> gcc -c helloCTFer.s -o helloCTFer.o ===> 查看.o的架構
 連結 ===> gcc  helloCTFer.o -o helloCTFer  ===> 查看helloCTFer的架構
```

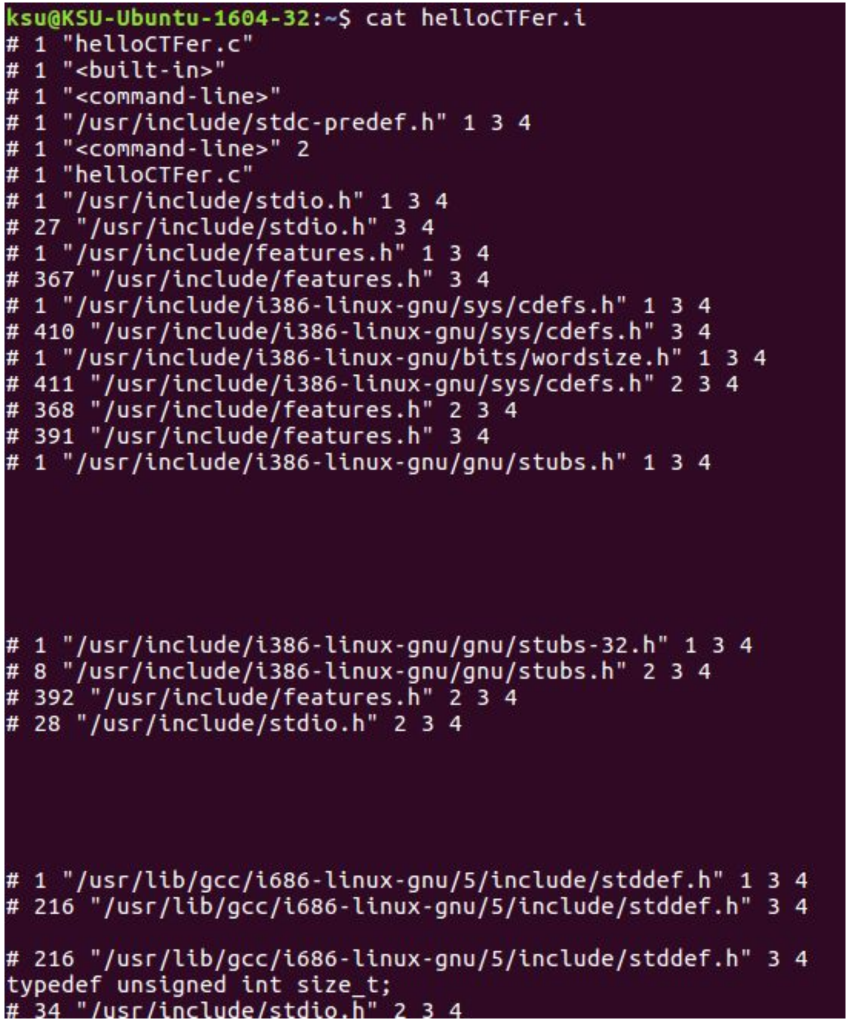
預處理

```
gcc -E helloCTFer.c -o helloCTFer.i
```

```
 ls -al helloCTFer.*
```



```
ksu@KSU-Ubuntu-1604-32:~$ gcc -E helloCTFer.c -o helloCTFer.i
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu    73  三   6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu  7356  三   6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538  三   6 13:48 helloCTFer.i
```

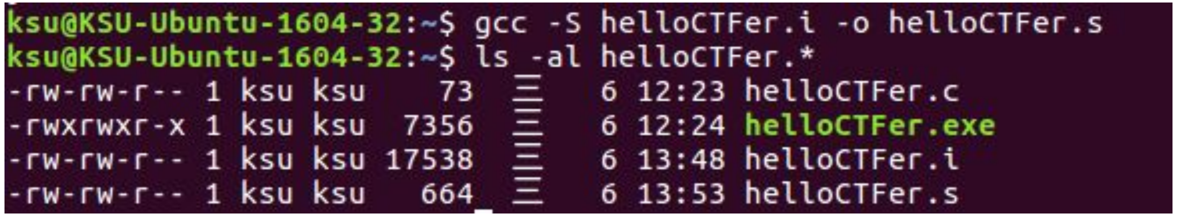

```
ksu@KSU-Ubuntu-1604-32:~$ cat helloCTFer.i
# 1 "helloCTFer.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "helloCTFer.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 367 "/usr/include/features.h" 3 4
# 1 "/usr/include/i386-linux-gnu/sys/cdefs.h" 1 3 4
# 410 "/usr/include/i386-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/i386-linux-gnu/bits/wordsize.h" 1 3 4
# 411 "/usr/include/i386-linux-gnu/sys/cdefs.h" 2 3 4
# 368 "/usr/include/features.h" 2 3 4
# 391 "/usr/include/features.h" 3 4
# 1 "/usr/include/i386-linux-gnu/gnu/stubs.h" 1 3 4







# 1 "/usr/include/i386-linux-gnu/gnu/stubs-32.h" 1 3 4
# 8 "/usr/include/i386-linux-gnu/gnu/stubs.h" 2 3 4
# 392 "/usr/include/features.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4







# 1 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 1 3 4
# 216 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 3 4

# 216 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 3 4
typedef unsigned int size_t;
# 34 "/usr/include/stdio.h" 2 3 4
```

```
cat helloCTFer.i
```

編譯

```
gcc -S helloCTFer.i  -o helloCTFer.s
```

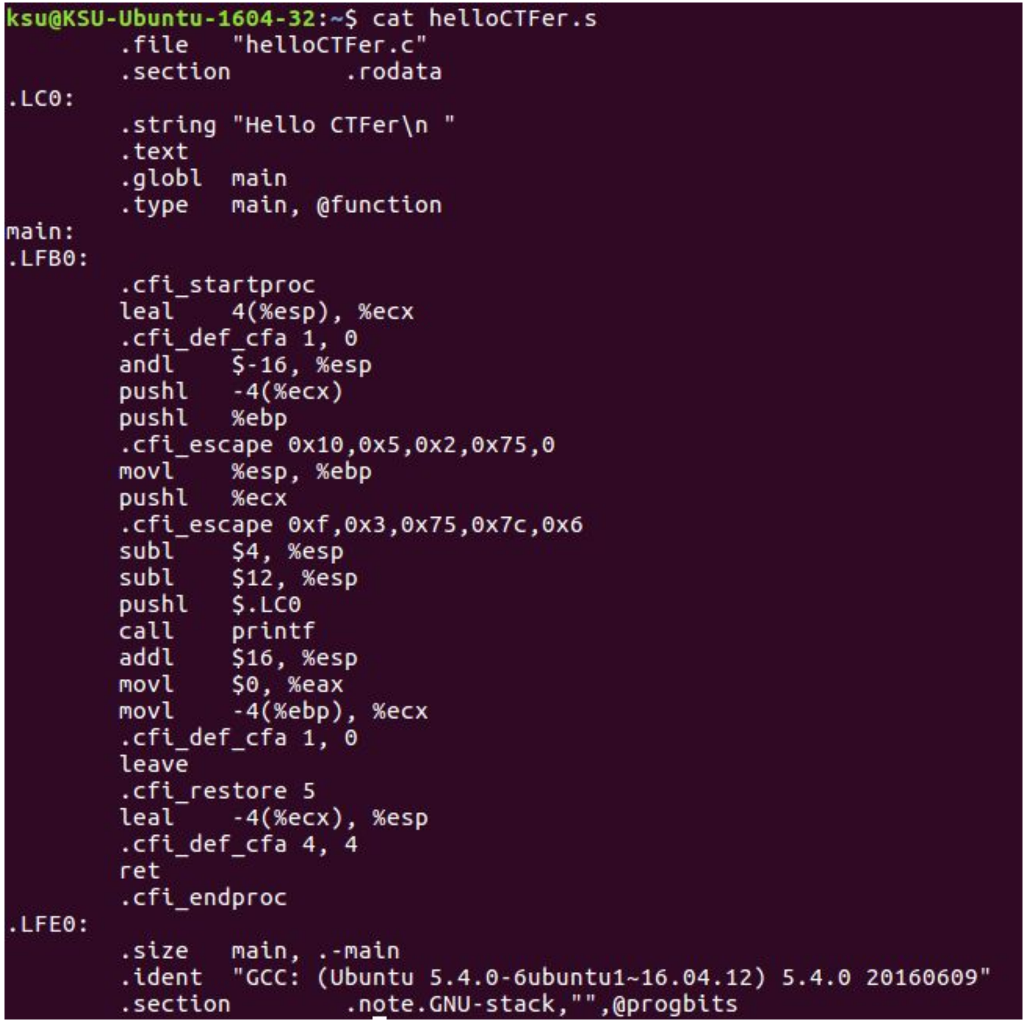
```
ls -al helloCTFer.*
```



```
ksu@KSU-Ubuntu-1604-32:~$ gcc -S helloCTFer.i -o helloCTFer.s
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu    73  三    6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu  7356  三    6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538  三    6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu   664  三    6 13:53 helloCTFer.s
```

AT&T

```
cat helloCTFer.s  ==>
```

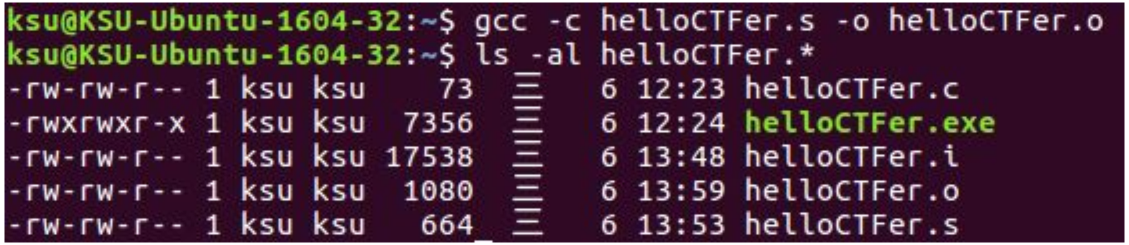


```
ksu@KSU-Ubuntu-1604-32:~$ cat helloCTFer.s
        .file   "helloCTFer.c"
        .section        .rodata
.LC0:
        .string "Hello CTFer\n "
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        leal    4(%esp), %ecx
        .cfi_def_cfa 1, 0
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0
        movl    %esp, %ebp
        pushl   %ecx
        .cfi_escape 0xf,0x3,0x75,0x7c,0x6
        subl    $4, %esp
        subl    $12, %esp
        pushl   $.LC0
        call    printf
        addl    $16, %esp
        movl    $0, %eax
        movl    -4(%ebp), %ecx
        .cfi_def_cfa 1, 0
        leave
        .cfi_restore 5
        leal    -4(%ecx), %esp
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

彙編

```
gcc -c helloCTFer.s -o helloCTFer.o
```

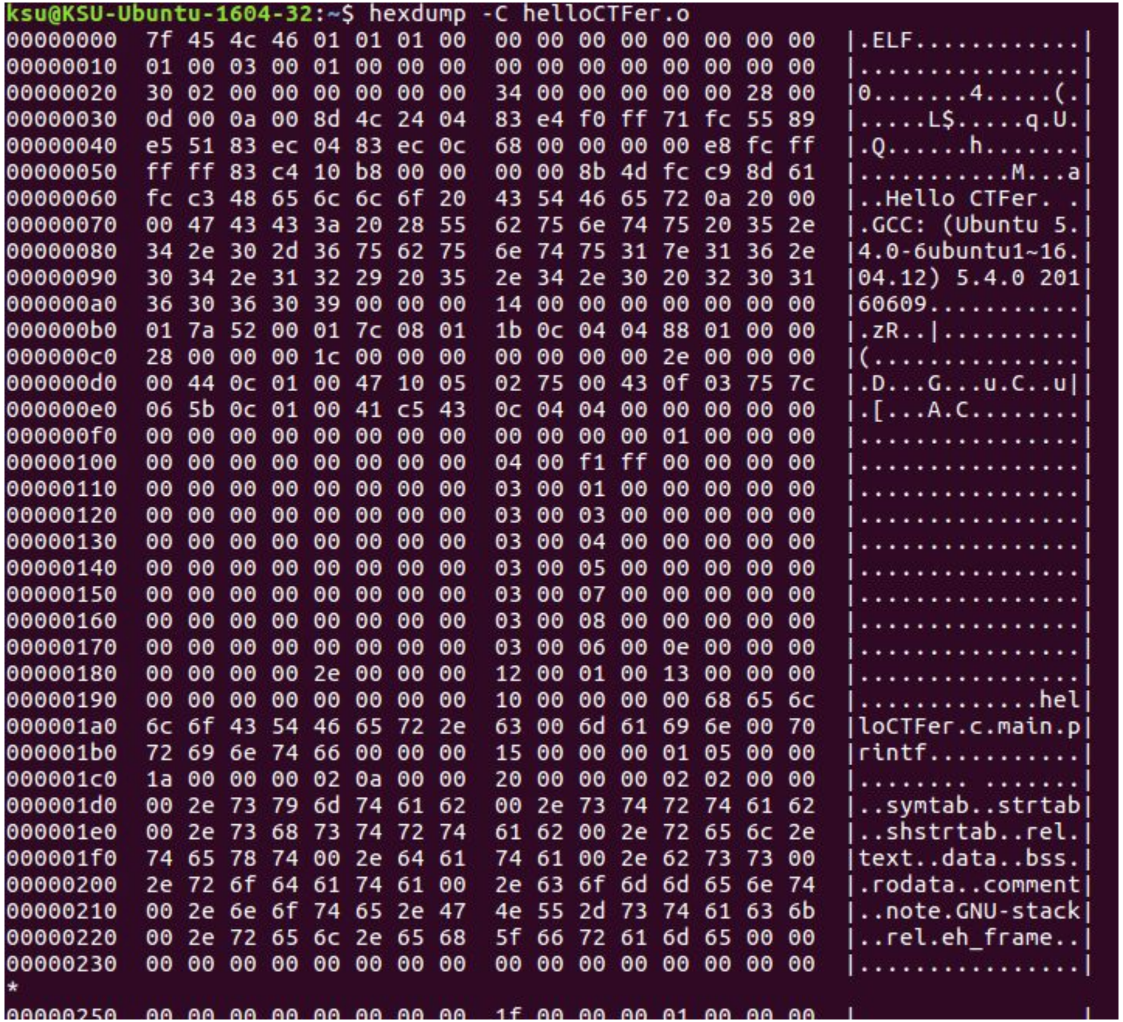
```
ls -al helloCTFer.*
```



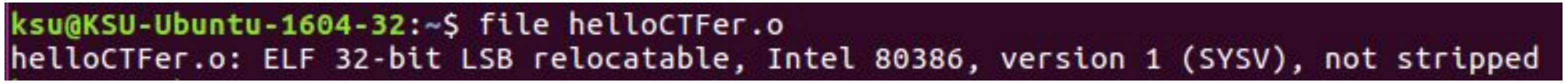
```
ksu@KSU-Ubuntu-1604-32:~$ gcc -c helloCTFer.s -o helloCTFer.o
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu    73 三   6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu  7356 三   6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538 三   6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu  1080 三   6 13:59 helloCTFer.o
-rw-rw-r-- 1 ksu ksu   664 三   6 13:53 helloCTFer.s
```

查看helloCTFer.o

```
hexdump -C helloCTFer.o
```



```
ksu@KSU-Ubuntu-1604-32:~$ hexdump -C helloCTFer.o
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  01 00 03 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  30 02 00 00 00 00 00 00  34 00 00 00 00 00 28 00  |0.......4.....(.|
00000030  0d 00 0a 00 8d 4c 24 04  83 e4 f0 ff 71 fc 55 89  |.....L$.....q.U.|
00000040  e5 51 83 ec 04 83 ec 0c  68 00 00 00 00 e8 fc ff  |.Q......h.......|
00000050  ff ff 83 c4 10 b8 00 00  00 00 8b 4d fc c9 8d 61  |...........M...a|
00000060  fc c3 48 65 6c 6c 6f 20  43 54 46 65 72 0a 20 00  |..Hello CTFer. .|
00000070  00 47 43 43 3a 20 28 55  62 75 6e 74 75 20 35 2e  |.GCC: (Ubuntu 5.|
00000080  34 2e 30 2d 36 75 62 75  6e 74 75 31 7e 31 36 2e  |4.0-6ubuntu1~16.|
00000090  30 34 2e 31 32 29 20 35  2e 34 2e 30 20 32 30 31  |04.12) 5.4.0 201|
000000a0  36 30 36 30 39 00 00 00  14 00 00 00 00 00 00 00  |60609...........|
000000b0  01 7a 52 00 01 7c 08 01  1b 0c 04 04 88 01 00 00  |.zR..|..........|
000000c0  28 00 00 00 1c 00 00 00  00 00 00 00 2e 00 00 00  |(...............|
000000d0  00 44 0c 01 00 47 10 05  02 75 00 43 0f 03 75 7c  |.D...G...u.C..u||
000000e0  06 5b 0c 01 00 41 c5 43  0c 04 04 00 00 00 00 00  |.[...A.C........|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00  |................|
00000100  00 00 00 00 00 00 00 00  04 00 f1 ff 00 00 00 00  |................|
00000110  00 00 00 00 00 00 00 00  03 00 01 00 00 00 00 00  |................|
00000120  00 00 00 00 00 00 00 00  03 00 03 00 00 00 00 00  |................|
00000130  00 00 00 00 00 00 00 00  03 00 04 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  03 00 05 00 00 00 00 00  |................|
00000150  00 00 00 00 00 00 00 00  03 00 07 00 00 00 00 00  |................|
00000160  00 00 00 00 00 00 00 00  03 00 08 00 00 00 00 00  |................|
00000170  00 00 00 00 00 00 00 00  03 00 06 00 0e 00 00 00  |................|
00000180  00 00 00 00 2e 00 00 00  12 00 01 00 13 00 00 00  |................|
00000190  00 00 00 00 00 00 00 00  10 00 00 00 00 68 65 6c  |.............hel|
000001a0  6c 6f 43 54 46 65 72 2e  63 00 6d 61 69 6e 00 70  |loCTFer.c.main.p|
000001b0  72 69 6e 74 66 00 00 00  15 00 00 00 01 05 00 00  |rintf...........|
000001c0  1a 00 00 00 02 0a 00 00  20 00 00 00 02 02 00 00  |........ .......|
000001d0  00 2e 73 79 6d 74 61 62  00 2e 73 74 72 74 61 62  |..symtab..strtab|
000001e0  00 2e 73 68 73 74 72 74  61 62 00 2e 72 65 6c 2e  |..shstrtab..rel.|
000001f0  74 65 78 74 00 2e 64 61  74 61 00 2e 62 73 73 00  |text..data..bss.|
00000200  2e 72 6f 64 61 74 61 00  2e 63 6f 6d 6d 65 6e 74  |.rodata..comment|
00000210  00 2e 6e 6f 74 65 2e 47  4e 55 2d 73 74 61 63 6b  |..note.GNU-stack|
00000220  00 2e 72 65 6c 2e 65 68  5f 66 72 61 6d 65 00 00  |..rel.eh_frame..|
00000230  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000250  00 00 00 00 00 00 00 00  1f 00 00 00 01 00 00 00  |................|
file helloCTFer.o
```

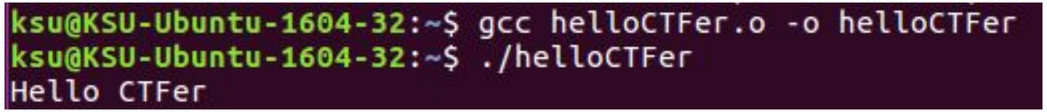


```
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.o
helloCTFer.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped
```

連結

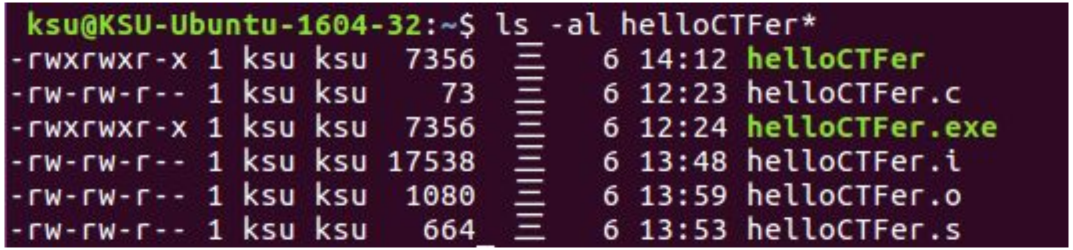
```
gcc  helloCTFer.o -o helloCTFer
```

```
./helloCTFer
```



```
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.o -o helloCTFer
ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer
Hello CTFer
```

```
ls -al helloCTFer*
```

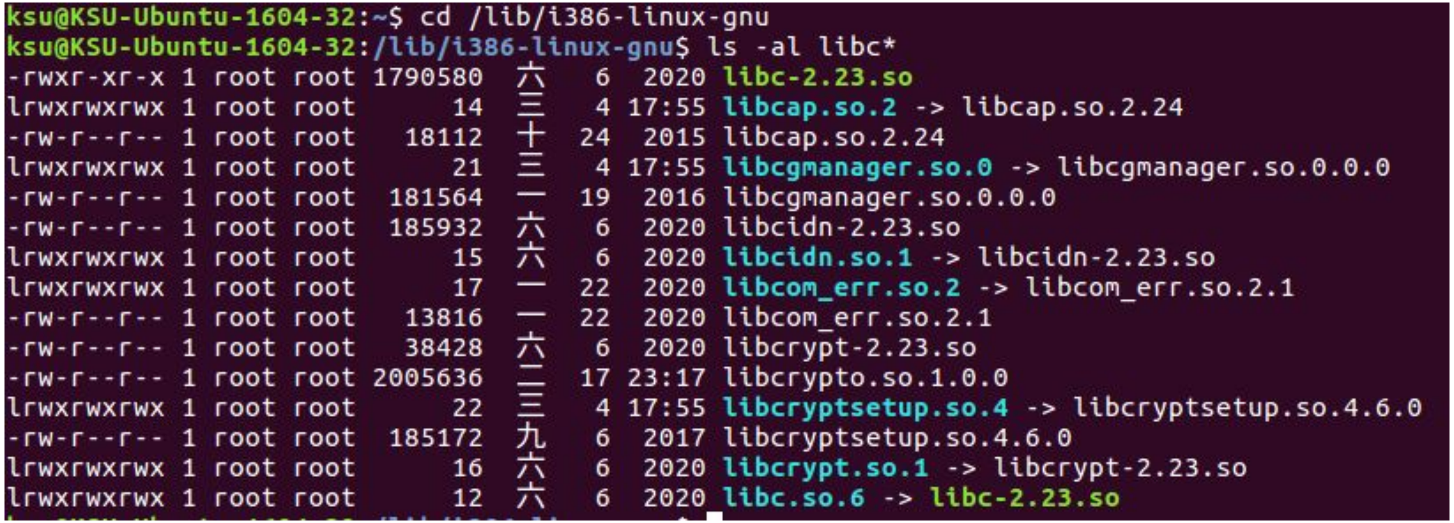


```
 ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer*
-rwxrwxr-x 1 ksu ksu  7356 [illegible]  6 14:12 helloCTFer
-rw-rw-r-- 1 ksu ksu    73 [illegible]  6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu  7356 [illegible]  6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538 [illegible]  6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu  1080 [illegible]  6 13:59 helloCTFer.o
-rw-rw-r-- 1 ksu ksu   664 [illegible]  6 13:53 helloCTFer.s
```

查看你linux所使用的C函式庫版本[Ubuntu 16.04 LTS (32bits)]

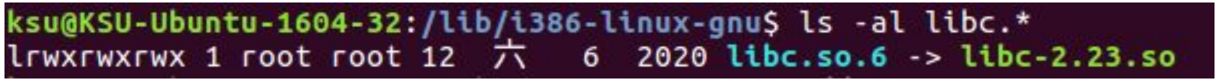
```
Linux: Check the glibc version
Posted on January 28, 2015
https://benohead.com/blog/2015/01/28/linux-check-glibc-version/
```

```
cd /lib/i386-linux-gnu

ls -al libc*
```



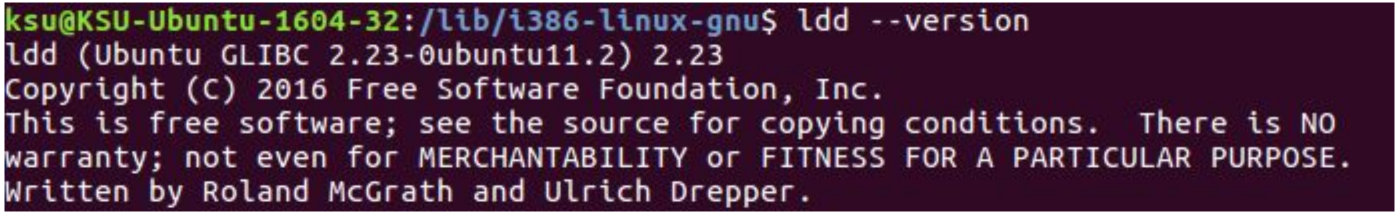
```
ksu@KSU-Ubuntu-1604-32:~$ cd /lib/i386-linux-gnu
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ls -al libc*
-rwxr-xr-x 1 root root 1790580  六   6  2020 libc-2.23.so
lrwxrwxrwx 1 root root      14  三   4 17:55 libcap.so.2 -> libcap.so.2.24
-rw-r--r-- 1 root root   18112  十  24  2015 libcap.so.2.24
lrwxrwxrwx 1 root root      21  三   4 17:55 libcgmanager.so.0 -> libcgmanager.so.0.0.0
-rw-r--r-- 1 root root  181564  一  19  2016 libcgmanager.so.0.0.0
-rw-r--r-- 1 root root  185932  六   6  2020 libcidn-2.23.so
lrwxrwxrwx 1 root root      15  六   6  2020 libcidn.so.1 -> libcidn-2.23.so
lrwxrwxrwx 1 root root      17  一  22  2020 libcom_err.so.2 -> libcom_err.so.2.1
-rw-r--r-- 1 root root   13816  一  22  2020 libcom_err.so.2.1
-rw-r--r-- 1 root root   38428  六   6  2020 libcrypt-2.23.so
-rw-r--r-- 1 root root 2005636  二  17 23:17 libcrypto.so.1.0.0
lrwxrwxrwx 1 root root      22  三   4 17:55 libcryptsetup.so.4 -> libcryptsetup.so.4.6.0
-rw-r--r-- 1 root root  185172  九   6  2017 libcryptsetup.so.4.6.0
lrwxrwxrwx 1 root root      16  六   6  2020 libcrypt.so.1 -> libcrypt-2.23.so
lrwxrwxrwx 1 root root      12  六   6  2020 libc.so.6 -> libc-2.23.so
```

```
ls -al libc.*
```



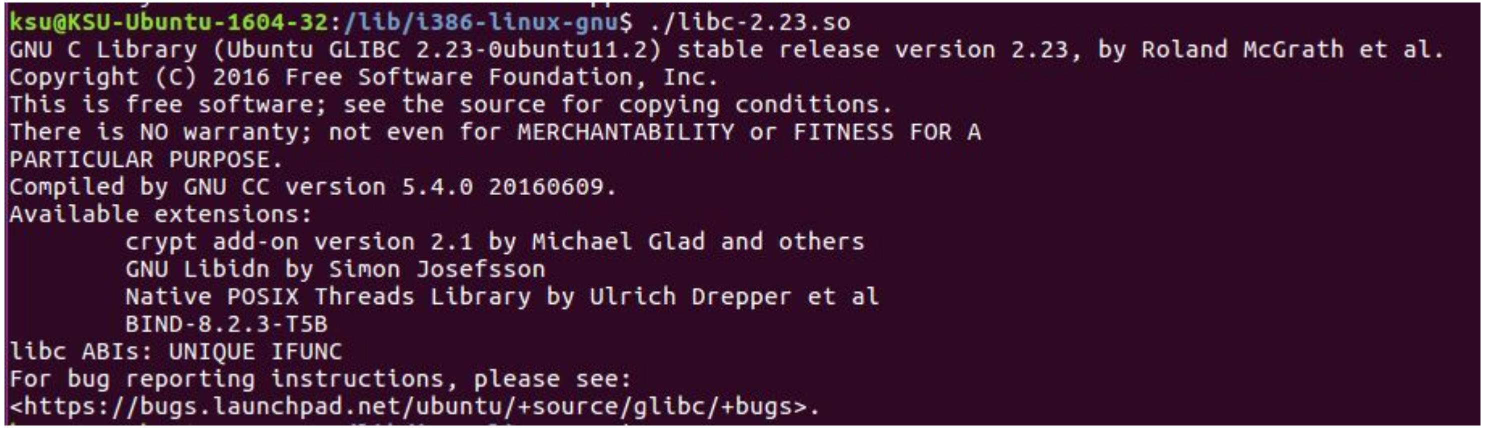
```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ls -al libc.*
lrwxrwxrwx 1 root root 12  六   6  2020 libc.so.6 -> libc-2.23.so
```

```
ldd --version
```

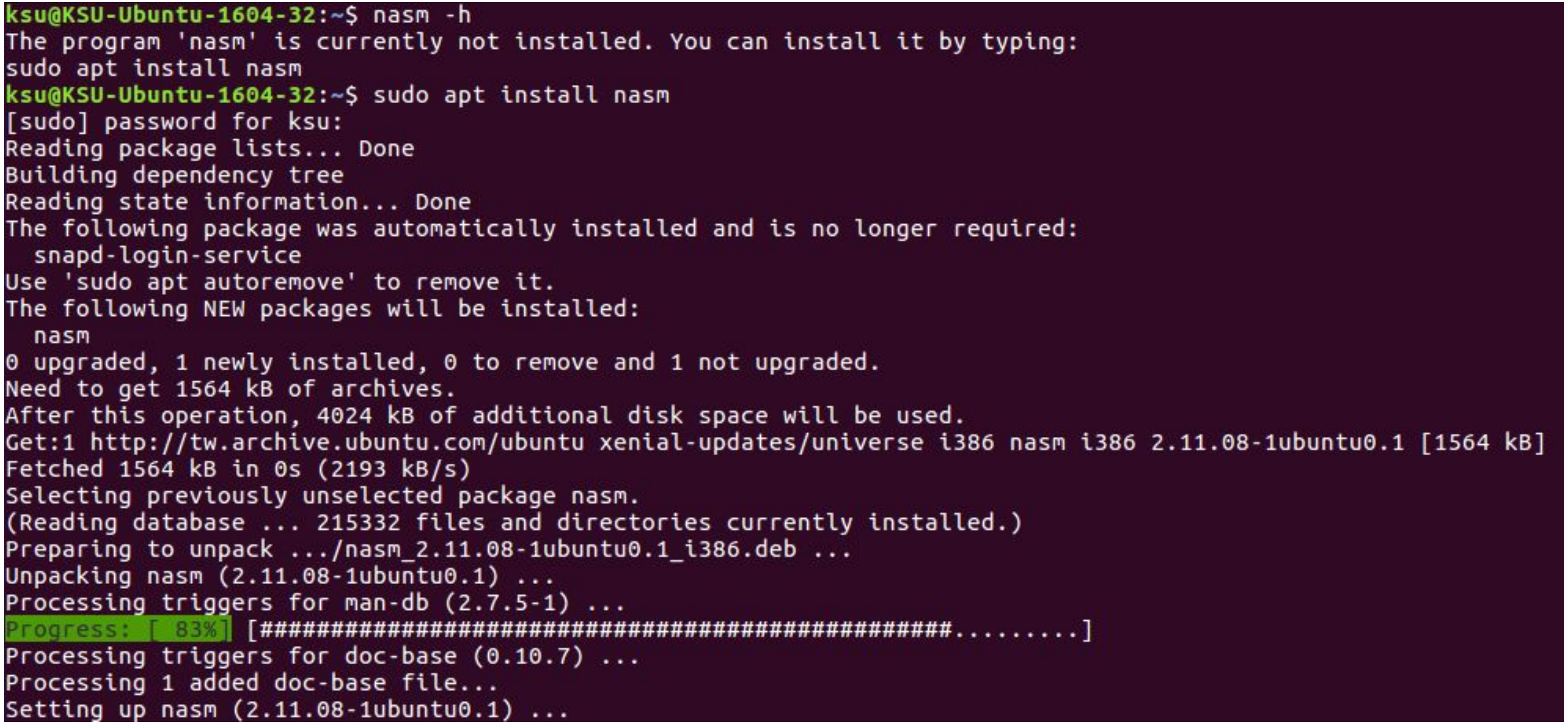


```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ldd --version
ldd (Ubuntu GLIBC 2.23-0ubuntu11.2) 2.23
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```

```
./libc-2.23.so
```



```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ./libc-2.23.so
GNU C Library (Ubuntu GLIBC 2.23-0ubuntu11.2) stable release version 2.23, by Roland McGrath et al.
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 5.4.0 20160609.
Available extensions:
        crypt add-on version 2.1 by Michael Glad and others
        GNU Libidn by Simon Josefsson
        Native POSIX Threads Library by Ulrich Drepper et al
        BIND-8.2.3-T5B
libc ABIs: UNIQUE IFUNC
For bug reporting instructions, please see:
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
```



```
ksu@KSU-Ubuntu-1604-32:~$ nasm -h
The program 'nasm' is currently not installed. You can install it by typing:
sudo apt install nasm
ksu@KSU-Ubuntu-1604-32:~$ sudo apt install nasm
[sudo] password for ksu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  snapd-login-service
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  nasm
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 1564 kB of archives.
After this operation, 4024 kB of additional disk space will be used.
Get:1 http://tw.archive.ubuntu.com/ubuntu xenial-updates/universe i386 nasm i386 2.11.08-1ubuntu0.1 [1564 kB]
Fetched 1564 kB in 0s (2193 kB/s)
Selecting previously unselected package nasm.
(Reading database ... 215332 files and directories currently installed.)
Preparing to unpack .../nasm_2.11.08-1ubuntu0.1_i386.deb ...
Unpacking nasm (2.11.08-1ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Progress: [ 83%] [#################################################.........]
Processing triggers for doc-base (0.10.7) ...
Processing 1 added doc-base file...
Setting up nasm (2.11.08-1ubuntu0.1) ...
```