```
$Id: extern-tutorial.mm,v 1.67 2014-10-10 17:13:03-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps104a-wm/Assignments/extern-tutorial
URL: http://www2.ucsc.edu/courses/cmps104a-wm/:/Assignments/extern-tutorial/
```

This is a short tutorial on the use of the **`extern`** keyword in C. Each brief item comments on a shell command, the output of which is shown after the command. User input is shown in **`Courier-Bold`** and computer output is shown in plain **`Courier`**.

(1) All of these commands are being run on a Unix server. First, let's look at some of the server's properties.

```
-bash-1$ hostname
unix1.lt.ucsc.edu

-bash-2$ uname --kernel-name --kernel-release --kernel-version
Linux 2.6.32-431.29.2.el6.x86_64 #1 SMP Tue Sep 9 21:36:05 UTC 2014

-bash-3$ uname --nodename --operating-system
unix1.lt.ucsc.edu GNU/Linux

-bash-4$ uname --machine --processor --hardware-platform
x86_64 x86_64 x86_64
```

(2) The program was built with the simple script **`mk`**.

```
-bash-5$ cat code/mk
#!/bin/sh
cid + *.h *.c $0
GCCOPT="-g -O0 -Wall -Wextra -std=gnu11"
gcc -c $GCCOPT *.c
gcc *.o

-bash-6$ cd code; mk
```

(3) Using the command **`file`**(1), we examine the types of the files in the **`code/`** subdirectory.

```
-bash-7$ file code/*
code/HEADER.html: HTML document text
code/RCS:         directory
code/a.out:       ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.18, not stripped
code/ext.c:       ASCII C++ program text
code/ext.h:       ASCII C++ program text
code/ext.o:       ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/main.c:      ASCII C++ program text
code/main.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/mk:          POSIX shell script text executable
```

(4) The files in **`code/`** are listed as follows.

```
-bash-8$ ls -goad code/*
-rw-------. 1  527 Oct  3 15:38 code/HEADER.html
drwx------. 2 2048 Oct 10 17:13 code/RCS
```

```
-rwx------. 1 8421 Oct 10 17:13 code/a.out
-rw-------. 1  189 Oct 10 14:23 code/ext.c
-rw-------. 1  152 Oct 10 14:23 code/ext.h
-rw-------. 1 3512 Oct 10 17:13 code/ext.o
-rw-------. 1  223 Oct 10 14:27 code/main.c
-rw-------. 1 3472 Oct 10 17:13 code/main.o
-rwx------. 1   95 Oct 10 14:26 code/mk
```

(5) The file **`code/main.c`** uses an external variable exported from another module.

```
-bash-9$ cat code/main.c
// This is an example of a module accessing an external variable.
#include <stdio.h>
#include <stdlib.h>
#include "ext.h"
int main (void) {
   print_ext_var();
   ext_var = 56;
   print_ext_var();
   return EXIT_SUCCESS;
}
```

(6) The file **`code/ext.c`** exports an external variable.

```
-bash-10$ cat code/ext.c
// This is an example of a module exporting an external variable.
#include <stdio.h>
#include "ext.h"
int ext_var = 44;
void print_ext_var (void) {
   printf ("ext_var = %d\n", ext_var);
}
```

(7) The file **`code/ext.h`** is included in both and links the two. Note the file guards.

```
-bash-11$ cat code/ext.h
// This is an example of a header exported by the ext module.
#ifndef __EXT_H__
#define __EXT_H__
extern int ext_var;
void print_ext_var (void);
#endif
```

(8) When run, the program produces the following output.

```
-bash-12$ code/a.out
ext_var = 44
ext_var = 56
```

(9) Every module that accesses an external variable must declare it using the **`extern`** keyword. In order to ensure consistency of declaration, this should be placed in a header file. The module exporting the variable, and only that module, then redeclares that same variable without the **`extern`** keyword. Every external variable must be declared without the **`extern`** keyword in

exactly one module.

(10) If not declared at all, one gets an undefined external reference error at link time. If declared more than once, then the error is a duplicate declaration error at link time. If not declared as **`extern`** in a header file, then the variables are local to the file and not related.

(11) Now consider the output of running **`nm`**(1) on each of the object files. The **`extern`** keyword in the header file marks the variable as external, that is global to both modules. It is redeclared in the file **`ext.c`** without the **`extern`**keyword,**so `nm code/ext.o`** produces the following output.

```
-bash-13$ nm code/ext.o
0000000000000000 D ext_var
0000000000000000 T print_ext_var
                 U printf
```

(12) On the other hand, running **`nm code/main.o`** shows that **`external_variable`** is undefined in that module.

```
-bash-14$ nm code/main.o
                 U ext_var
0000000000000000 T main
                 U print_ext_var
```

(13) Looking at the executable image **`a.out`** with **`nm code/a.out`** we see that each symbol has a specific address assigned to it. It also has references included from the library. The letter shows whether the object is Undefined, or belongs to the Text, Data, or BSS segment, or if it is Absolute. See **`nm`**(1) for a complete explanation.

```
-bash-15$ nm code/a.out
0000000000600700 d _DYNAMIC
00000000006008d8 d _GLOBAL_OFFSET_TABLE_
00000000004005c8 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
                 w _Jv_RegisterClasses
00000000004006e0 r __FRAME_END__
00000000006006f8 d __JCR_END__
00000000006006f8 d __JCR_LIST__
0000000000600908 D __TMC_END__
0000000000600908 B __bss_start
0000000000600900 D __data_start
0000000000400490 t __do_global_dtors_aux
00000000006006f0 t __do_global_dtors_aux_fini_array_entry
00000000004005d0 R __dso_handle
00000000006006e8 t __frame_dummy_init_array_entry
                 w __gmon_start__
00000000006006f0 t __init_array_end
00000000006006e8 t __init_array_start
0000000000400520 T __libc_csu_fini
```

```
0000000000400530 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000600908 D _edata
0000000000600910 B _end
00000000004005bc T _fini
0000000000400390 T _init
00000000004003d0 T _start
00000000004003fc t call_gmon_start
0000000000600908 b completed.6272
0000000000600900 W data_start
0000000000400420 t deregister_tm_clones
0000000000600904 D ext_var
00000000004004b0 t frame_dummy
0000000000400500 T main
00000000004004e0 T print_ext_var
                 U printf@@GLIBC_2.2.5
0000000000400450 t register_tm_clones
```

(14) The sizes of the segments in the object files and executable binary can be obtained via **`size`**(1).

```
-bash-16$ cd code; size *.o a.out
   text    data     bss     dec     hex filename
     99       4       0     103      67 ext.o
     87       0       0      87      57 main.o
   1236     544       8    1788     6fc a.out
```