```
$Id: extern-tutorial.mm,v 1.5 2019-04-12 16:18:17-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cse111-wm/Lecture-notes/extern-tutorial
URL: http://www2.ucsc.edu/courses/cse111-wm/:/Lecture-notes/extern-tutorial/
```

This is a short tutorial on the use of the **`extern`** keyword in C and 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`**.

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
unix3.lt.ucsc.edu

-bash-2$ uname --kernel-name --kernel-release --kernel-version
Linux 3.10.0-1062.9.1.el7.x86_64 #1 SMP Fri Dec 6 15:49:49 UTC 2019

-bash-3$ uname --nodename --operating-system
unix3.lt.ucsc.edu GNU/Linux

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

-bash-5$ which g++
/opt/rh/devtoolset-8/root/usr/bin/g++

-bash-6$ g++ --version
g++ (GCC) 8.3.1 20190311 (Red Hat 8.3.1-3)
Copyright (C) 2018 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.
```

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

```
-bash-7$ cat -nv code/mk
     1  #!/bin/sh
     2  cid + *.h *.cpp $0
     3  GPPOPT="-g -O0 -std=gnu++17 -Wall -Wextra -Wold-style-cast"
     4  g++ -c $GPPOPT *.cpp
     5  g++ *.o

-bash-8$ cd code; mk
```

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

```
-bash-9$ file code/*
code/HEADER.html: HTML document, ASCII 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 2.6.32,
BuildID[sha1]=7084ced6da687de337483827c4f0a8d5043292ef, not stripped
code/ext.cpp:     C source, ASCII text
code/ext.h:       C source, ASCII text
code/ext.o:       ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/main.cpp:    C source, ASCII text
code/main.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/mk:          POSIX shell script, ASCII text executable
```

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

```
-bash-10$ ls -goad code/*
-rw------- 1   545 Mar 11 15:25 code/HEADER.html
drwx------ 2  2048 Mar 11 15:26 code/RCS
-rwx------ 1 14360 Mar 11 15:26 code/a.out
-rw------- 1   240 Mar 11 15:26 code/ext.cpp
-rw------- 1   199 Mar 11 15:26 code/ext.h
-rw------- 1  6240 Mar 11 15:26 code/ext.o
-rw------- 1   255 Mar 11 15:26 code/main.cpp
-rw------- 1  7704 Mar 11 15:26 code/main.o
-rwx------ 1   118 Mar 11 15:25 code/mk
```

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

```
-bash-11$ cat -nv code/ext.h
// $Id: ext.h,v 1.1 2017-10-11 14:05:24-07 - - $
// 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();

#endif
```

The file **`code/ext.cpp`** exports an external variable.

```
-bash-12$ cat -nv code/ext.cpp
// $Id: ext.cpp,v 1.1 2017-10-11 14:05:24-07 - - $
// 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() {
   printf ("ext_var = %d0, ext_var);
}

```

The file **`code/main.cpp`** uses an external variable exported from another module.

```
-bash-13$ cat -nv code/main.cpp
// $Id: main.cpp,v 1.1 2017-10-11 14:05:24-07 - - $
// This is an example of a module accessing an external variable.

#include <stdlib.h>

#include "ext.h"

int main() {
   print_ext_var();
   ext_var = 56;
   print_ext_var();
   return EXIT_SUCCESS;
}

```

When run, the program produces the following output.

```
-bash-14$ code/a.out
ext_var = 44
ext_var = 56
```

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.

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.

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.cpp`** without the **`extern`**keyword,**`so nm code/ext.o`** produces the following output.

```
-bash-15$ nm code/ext.o
0000000000000000 T _Z13print_ext_varv
0000000000000000 D ext_var
                 U printf
```

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

```
-bash-16$ nm code/main.o
                 U _Z13print_ext_varv
                 U ext_var
0000000000000000 T main
```

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

```
-bash-17$ cd code; size *.o a.out
   text    data     bss     dec     hex filename
    100       4       0     104      68 ext.o
     87       0       0      87      57 main.o
   1305     592       8    1905     771 a.out
```

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-18$ nm code/a.out
0000000000600df8 d _DYNAMIC
0000000000601000 d _GLOBAL_OFFSET_TABLE_
0000000000400600 R _IO_stdin_used
0000000000400542 T _Z13print_ext_varv
000000000040076c r __FRAME_END__
0000000000400620 r __GNU_EH_FRAME_HDR
0000000000601038 D __TMC_END__
0000000000601038 B __bss_start
0000000000601030 D __data_start
0000000000400510 t __do_global_dtors_aux
0000000000600df0 t __do_global_dtors_aux_fini_array_entry
0000000000400608 R __dso_handle
0000000000600de8 t __frame_dummy_init_array_entry
                 w __gmon_start__
0000000000600df0 t __init_array_end
0000000000600de8 t __init_array_start
00000000004005f0 T __libc_csu_fini
0000000000400580 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000601038 D _edata
0000000000601040 B _end
00000000004005f4 T _fini
0000000000400408 T _init
0000000000400470 T _start
0000000000601038 b completed.7247
0000000000601030 W data_start
00000000004004a0 t deregister_tm_clones
0000000000601034 D ext_var
0000000000400540 t frame_dummy
0000000000400560 T main
                 U printf@@GLIBC_2.2.5
00000000004004d0 t register_tm_clones
```