Dynamic & Static Libraries in Linux

Wim Cardoen and Brett Milash
CHPC User Services

Overview

- Warm-up exercise: Install GSL
- What is a library?
- Static vs. Dynamic libraries
- Create static & dynamic libraries
- Link with the above libraries to generate an executable.

Exercise 5: GSL install – warm-up

- GSL: Gnu Scientific Library (external package/library)
  -> Numerical Library for C & C++ Programmers
- How to proceed:

  -> Download source code in: ~/Ex5/src/

  wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.4.tar.gz

  -> Extract the *.tar.gz file

  -> Use the gcc compiler with the following flags:

  setenv CC gcc ; setenv CFLAGS " -m64 -O2 **-fPIC** " # Tcsh

  export CC=gcc ; export CFLAGS=" -m64 -O2 **-fPIC**" # Bash

  -> Install GSL in: ~/Ex5/pkg/gsl/2.4 using:

  ./configure --prefix=$HOME/Ex5/pkg/gsl/2.4 --with-pic

  make –j 2 & make install

  -> Have a look in the following directory:

  ~/Ex5/pkg/gsl/2.4/lib

Exercise 5: Solution (Bash)

```
# Setup + check
export EX5_DIR=~/Ex5
export GSL_DIR=$EX5_DIR/pkg/gsl/2.4
echo $EX5_DIR
echo $GSL_DIR

# Download the GSL 2.4 source code
mkdir -p $EX5_DIR/src
cd $EX5_DIR/src
echo `pwd`
wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.4.tar.gz
tar -zxvf gsl-2.4.tar.gz
```

 # Compile + install the GSL package:

```
export CC=gcc
export CFLAGS=“ -m64 -O2 -fPIC “
echo $CC
echo $CFLAGS
```

 # Installation:

```
cd $EX5_DIR/src/gsl-2.4
./configure --prefix=$GSL_DIR --with-pic
make –j 2
make install
ls –la $GSL_DIR/lib
```

Exercise 5: Solution (Csh/Tcsh)

```
# Setup + check
setenv EX5_DIR ~/Ex5
setenv GSL_DIR $EX5_DIR/pkg/gsl/2.4
echo $EX5_DIR
echo $GSL_DIR

# Download the GSL 2.4 source code
mkdir -p $EX5_DIR/src
cd $EX5_DIR/src
echo `pwd`
wget ftp://ftp.gnu.org/gnu/gsl/gsl-2.4.tar.gz
tar -zxvf gsl-2.4.tar.gz
```

 # Compile + install the GSL package:

setenv CC gcc

setenv CFLAGS “ -m64 -O2 -fPIC “

echo $CC

echo $CFLAGS

 # Installation:

cd $EX5_DIR/src/gsl-2.4

./configure --prefix=$GSL_DIR --with-pic

make -j 2

make install

ls -la $GSL_DIR/lib

What is a library?

- Library: collection of objects
- Can contain data sets, functions, classes, etc.
- Primary use: **reuse of the code**

  e.g. zlib, gsl, etc.
- There are 2 types:
  - Static libraries: **.a** suffix
  - Dynamic libraries: **.so** suffix

Static Libraries

- Appeared first in time
- Have the **.a** suffix (**a**rchive file)

  e.g. libgsl**.a**, libz**.a**, etc.
- Specifics:

  -> copies **only** the required objects in the executable at linking time.

  -> larger executables than for the dyn. case

  -> requires more memory to load

  -> more portable & faster

Dynamic libraries

- Have the **.so** suffix (**s**hared **o**bject)

  e.g. libgsl**.so**, libz**.so**
- Specifics:

  -> no copy of object files into exe at linking

  -> require less disk space & less memory

  -> lib. can be updated without recompiling exe

  -> a little slower than static case

Create a library & use it.

- **Goal 1**:

  -> we want to create a 1D num. integ. library

  -> the library (**integ** directory) contains:

  src directory:

  a. *mc.c* (Monte-Carlo integration -> dep. on gsl)

  b. *trap.c* (Trapezoid rule)

  include directory:

  integ.h (header file)

  lib directory:

  we will create **libinteg.a** & ***libinteg.so***

- **Goal 2**:

  Use **newly created libraries** to create executables.

Create the Static Library

- Step 1: Generate the object files

  cd integ/src

  gcc -c -I$GSL_DIR/include -I../include mc.c

  gcc -c -I../include trap.c

- Step 2: Create the static library libinteg.a

  cd integ/lib

  ar -crv libinteg.a ../src/{mc.o,trap.o}

A little more on **ar**(chive)

- *ar -t libinteg.a* # Lists/**T**abulate content archive
- *ar -x libinteg.a mc.o* # E**x**tract mc.o WITHOUT deletion in the archive
- *ar -d libinteg.a mc.o* # **D**elete mc.o from archive
- *ar -q libinteg.a mc.o* # Append mc.o to archive
- *ar -r libinteg.a mc.o* # **R**eplace mc.o in archive
- *man ar*

Create a Dynamic Library

- Step 1: Generate the object files (use **-fPIC** compil. flag -> to avoid linking error)

  cd integ/src

  gcc -c **-fPIC** -I$GSL_DIR/include -I../include mc.c

  gcc -c **-fPIC** -I../include trap.c

- Step 2: Create the dynamic library libinteg.so

  cd integ/lib

  gcc –shared **–fPIC** -o libinteg.so ../src/{mc.o,trap.o}

A few useful commands/tools

- ldd [options] file
  find a *program*'s/library's shared libraries
  (ldd: list dynamic dependencies)

```
[u0253283@dirac:lib]$ ldd libinteg.a
ldd: warning: you do not have execution permission for
`./libinteg.a'
not a dynamic executable
```

NEVER use ldd against untrusted code (-> will be executed!)
-> use objdump instead

```
[u0253283@dirac:lib]$ ldd libinteg.so
 linux-vdso.so.1 =>  (0x00007fffc999b000)
 libc.so.6 => /lib64/libc.so.6 (0x00002ae6d803d000)
 /lib64/ld-linux-x86-64.so.2 (0x000055c2bddc4000)
```

```
u0253283@dirac:lib]$ ldd -v libinteg.so              # Verbose output => GLIBC
    linux-vdso.so.1 =>  (0x00007ffd66df1000)
    libc.so.6 => /lib64/libc.so.6 (0x00002b16e5789000)
    /lib64/ld-linux-x86-64.so.2 (0x000055be9ac18000)

    Version information:
    ./libinteg.so:
        libc.so.6 (GLIBC_2.2.5) => /lib64/libc.so.6
    /lib64/libc.so.6:
        ld-linux-x86-64.so.2 (GLIBC_2.3) => /lib64/ld-linux-x86-64.so.2
        ld-linux-x86-64.so.2 (GLIBC_PRIVATE) => /lib64/ld-linux-x86-64.so.2


[u0253283@dirac:lib]$ ldd -d libinteg.so    # Reports missing objects
undefined symbol: gsl_rng_default    (./libinteg.so)
    linux-vdso.so.1 =>  (0x00007ffde5318000)
    libc.so.6 => /lib64/libc.so.6 (0x00002ba359c5b000)
    /lib64/ld-linux-x86-64.so.2 (0x0000555f96ad9000)
```

- nm [options] file

  prints the **n**ame **l**ist (i.e. symbol table) of an object file

  Default output:

  1. Virtual address of the symbol
  2. Character/Symbol type:

     lower case: local upper case: external

     A/a: Global/local abs. type (Not changed when linking)

     B/b: Global/local uninitialized data

     D/d: Global/local initialized data

     f: Source file name symbol

     …

     L/l: Global/static thread-local symbol

     T/t: Global/local text symbol

     U: Undefined symbol
  3. Name of the symbol

- Note on the nm flags/options:

  -> nm --help : list an overview (you can also use man nm)

  -> nm -u file : list only the undefined symbols

  undefined: can either be **unresolved or can be resolved at runtime** through shared libraries

- objdump [options] file

  provides thorough information on object files

```
# Contents of the file header
[u0253283@dirac:mytest]$ objdump -f main_d1

main_d1:     file format elf64-x86-64
architecture: i386:x86-64, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x00000000004007a0


# Dumps assembler of the executable content
[u0253283@dirac:mytest]$ objdump -d main_d1 | less
main_d1:     file format elf64-x86-64


Disassembly of section .init:

00000000004006e0 <_init>:
4006e0:       48 83 ec 08             sub    $0x8,%rsp
4006e4:       48 8b 05 0d 19 20 00    mov    0x20190d(%rip),%rax        # 601ff8
<_DYNAMIC+0x210>
4006eb:       48 85 c0                test   %rax,%rax
4006ee:       74 05                   je     4006f5 <_init+0x15>
4006f0:       e8 4b 00 00 00          callq  400740 <__gmon_start__@plt>
4006f5:       48 83 c4 08             add    $0x8,%rsp
4006f9:       c3                      retq
```

To find dependencies

```
[u0253283@dirac:mytest]$ objdump -p main_d1 | grep NEEDED
  NEEDED               libgslcblas.so.0
  NEEDED               libgsl.so.23
  NEEDED               libm.so.6
  NEEDED               libinteg.so
  NEEDED               libc.so.6
```

Exercise 6

- Copy the file *LinuxLibs.tar.gz* in your $HOME
  cp /uufs/chpc.utah.edu/common/home/u0253283/Talks/LinuxLibs.tar.gz ~
- Extract (using the **tar** command) *LinuxLibs.tar.gz*
- Browse through the files within:

  $HOME/LinuxLibs/integ/{src,include}
- Generate the **static** library ***libinteg.a*** in the directory

  $HOME/LinuxLibs/integ/lib
- Use the **ar** command to see the content of *libinteg.a*

- Check the following commands:

  nm mc.o

  objdump –f –s –d mc.o

- Generate the **dynamic** library ***libinteg.so*** in the directory

  $HOME/LinuxLibs/integ/lib

  NOTE:

  To generate the dynamic library you MUST compile the source files with the –fPIC flag!

- Check the libraries using the ***ldd*** command:

  ldd ./libinteg.so

  ldd ./libinteg.a

The linking process => exe

- How to do linking?

  *gcc -o name_exe *.o* [library_info]

- [library_info]

  - If you use a library such as libm.{so,a} (sqrt, exp,…)

    => ‘ -lm ‘ is sufficient

    (no need to specify directory where libm.{so,a} is stored)

    Why? -> /etc/ld.so.conf.d directory

    ldconfig –p | grep libm

  - Otherwise (if library libmylib.{so,a} can’t be found **during linking**)

    => ‘-L$LIBDIR -lmylib ‘

    LIBDIR: directory where libmylib.{so,a} is stored

- Note:

  a. If the dyn. & static version of the same lib. are both present in LIBDIR=> dyn. library will be taken.

  b. If you want the static library to be taken, then use -L$LIBDIR $LIBDIR/libmylib.a

- Example:

  *gcc -o main_d1 main.o functions.o -L$GSL_DIR/lib –lgslblas -lgsl *
  -L$INTEG_DIR/lib -linteg -lm

  1. Dynamic libraries will be taken
  2. Error will be thrown when trying to run ./main_d1 => Why?

```
[u0253283@dirac:mytest]$ ./main_d1
./main_d1: error while loading shared libraries: libgslcblas.so.0: cannot open shared
object file: No such file or directory

[u0253283@dirac:mytest]$ ldd main_d1
    linux-vdso.so.1 =>  (0x00007ffc551b6000)
    libgslcblas.so.0 => not found
    libgsl.so.23 => not found
    libinteg.so => not found
    libm.so.6 => /lib64/libm.so.6 (0x00002b7549f21000)
    libc.so.6 => /lib64/libc.so.6 (0x00002b754a224000)
    /lib64/ld-linux-x86-64.so.2 (0x0000558b57cb2000)
```

Executables

- Two types:

  A. Static executable

  B. Executable relying on dyn. Libraries

- A. Static executable:

  -> You **MUST** use **STATIC** libraries

  -> Use the '-static' flag (GNU) at linking time

  -> Example:

  *gcc -static –o main_s2 main.o functions.o *
  *-L$INTEG_DIR/lib $INTEG_DIR/lib/libinteg.a *
  -L$GSL_DIR/lib $GSL_DIR/lib/libgsl.a $GSL_DIR/lib/libgslcblas.a -lm

- B.Exe based on dyn. Libraries

  The executable **MUST** find the dyn. libraries at **RUNTIME**

  (remember: "*error while loading shared libraries …*")

  Option 1: *ldconfig* command

  Command to create & maintain the cache for dyn. libraries (sys. admin tool => No Option for users!)

```
[u0253283@dirac:mytest]$ ldconfig -p | grep gsl
libgslcblas.so.0 (libc6,x86-64) => /lib64/libgslcblas.so.0
libgsl.so.0 (libc6,x86-64) => /lib64/libgsl.so.0

[u0253283@dirac:mytest]$ ldconfig -p | grep libinteg
[u0253283@dirac:mytest]$
```

Option2:

If the correct version of the library is **NOT** in the ldconfig cache, the user needs to feed the lib. info to the exe.

a. At **Runtime: -> use the LD_LIBRARY_PATH env. var.**

```
[u0253283@dirac:mytest]$ ldd ./main_d1
linux-vdso.so.1 =>  (0x00007ffc3fc96000)
libgslcblas.so.0 => /lib64/libgslcblas.so.0 (0x00002aab37273000)
libgsl.so.23 => not found
libinteg.so => not found
libm.so.6 => /lib64/libm.so.6 (0x00002aab374b1000)
libc.so.6 => /lib64/libc.so.6 (0x00002aab377b3000)
/lib64/ld-linux-x86-64.so.2 (0x000055bc8aeba000)
```

Solution:

export LD_LIBRARY_PATH=$LIBDIR:$LD_LIBRARY_PATH (Bash shell)

setenv LD_LIBRARY_PATH $LIBDIR:$LD_LIBRARY_PATH (Tcsh Shell)

=> the “not found “ message will disappear

b. At **Linking Time**:

Use the following construct when linking the code:

" -Wl,-rpath=$LIBDIR -L$LIBDIR -lmylib "

Example:

```
gcc -o main_d2 main.o functions.o \
    -Wl,-rpath=$GSL_DIR/lib -L$GSL_DIR/lib -lgslcblas -lgsl \
    -Wl,-rpath=$INTEG_DIR/lib -L$INTEG_DIR/lib -linteg -lm
```

Exercise 7

- We will now create executables based on the *gsl* and *integ* libraries.
- Create executables (within mytest) in different ways:

a. main_d1: using **only dynamic libraries** (gsl and integ)

without using the –Wl,-rpath construct

b. main_d2: using **only the dyn. libraries** (gsl & integ)

but **use the –Wl,-rpath construct**

c. main_s1: use the **dyn. gsl libraries** (**using the –Wl,-rpath construct**)

but use the **static library libinteg.a**

d. main_s2: create a **completely STATIC** executable.

(This requires glibc-static.x86_64 to be installed on the machine)

Questions?

Email helpdesk@chpc.utah.edu