Open Source compliance: Technical must-knows for legal experts

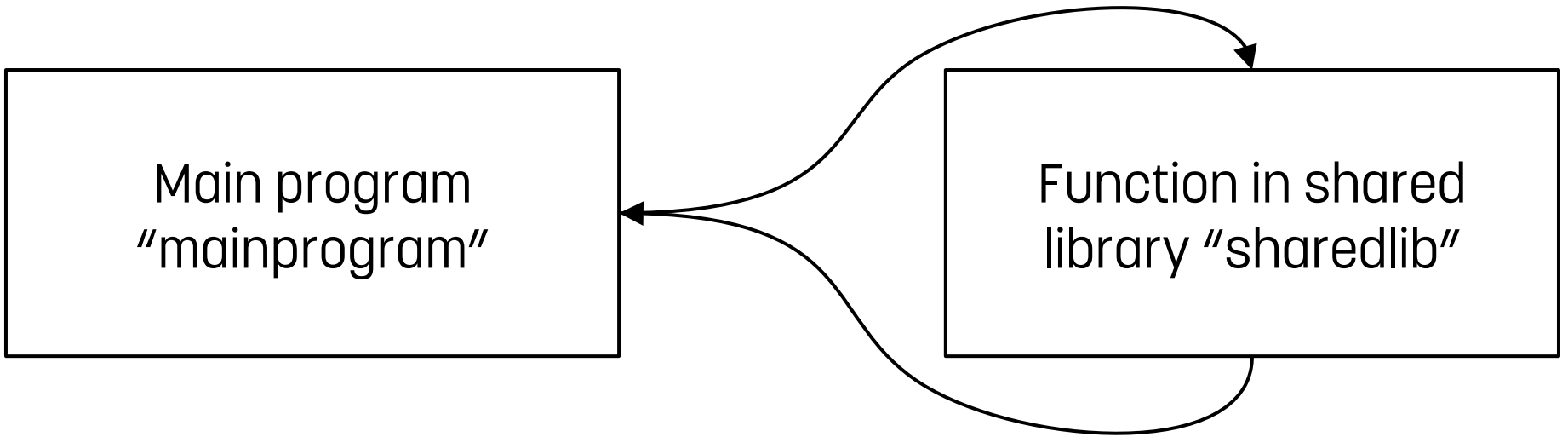
Carsten Emde
Open Source Automation Development Lab (OSADL) eG

Function calls, derivative work and callgraphs

Scenario #1: In the developer's room

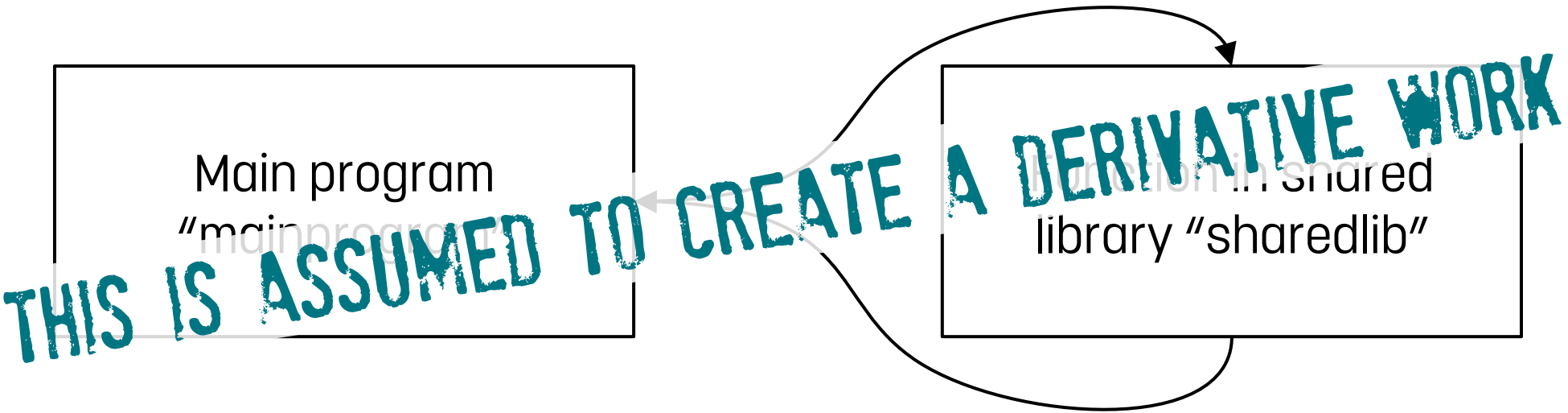
Overview

- We will create a main program and a shared library with a function that is called by the main program:



Overview

- We will create a main program and a shared library with a function that is called by the main program:

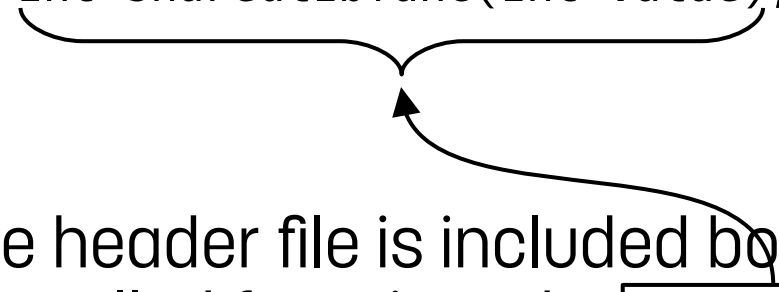


Definition and prototype in header file

```
$ cat sharedlib.h
#define VALUE 3262
int sharedlibfunc(int value);
```

Definition and prototype in header file

```
$ cat sharedlib.h
#define VALUE 3262
int sharedlibfunc(int value);
```



The header file is included both by the calling program and by the called function. The prototype ensures that the same types of arguments and return values are used.

Definition and prototype in header file

```
$ cat sharedlib.h
#define VALUE 3262
int sharedlibfunc(int value);
```

The header file is included both by the calling program and by the called function. The prototype ensures that the same types of arguments and return values are used. For the same reason, the value of VALUE is also defined here in this unique place.

Main program in C language

```
$ cat mainprogram.c
#include "sharedlib.h"

int main()
{
  return sharedlibfunc(VALUE);
}
```

The main program calls the function *sharedlibfunc*, passes an argument (VALUE), receives a return value and returns it to the operating system.

Main program in C language

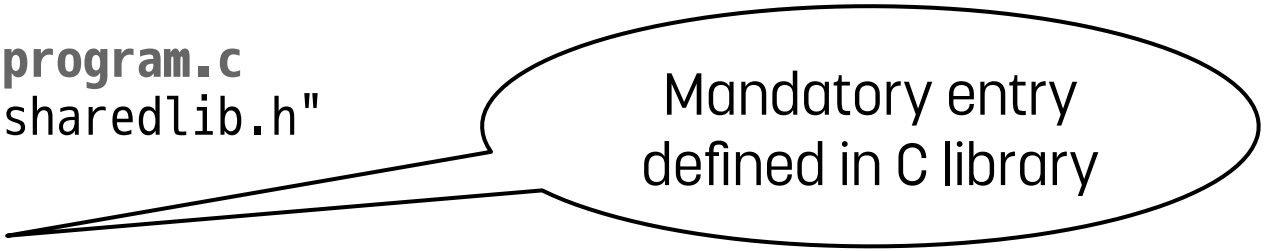
```
$ cat mainprogram.c
#include "sharedlib.h"

int main()
{
  return sharedlibfunc(VALUE);
}
```

```
#define VALUE 3262
int sharedlibfunc(int value);
```

The main program calls the function *sharedlibfunc*, passes an argument (VALUE), receives a return value and returns it to the operating system.

Main program in C language

```
$ cat mainprogram.c
#include "sharedlib.h"

int main()
{
  return sharedlibfunc(VALUE);
}
```



The main program calls the function *sharedlibfunc*, passes an argument (VALUE), receives a return value and returns it to the operating system.

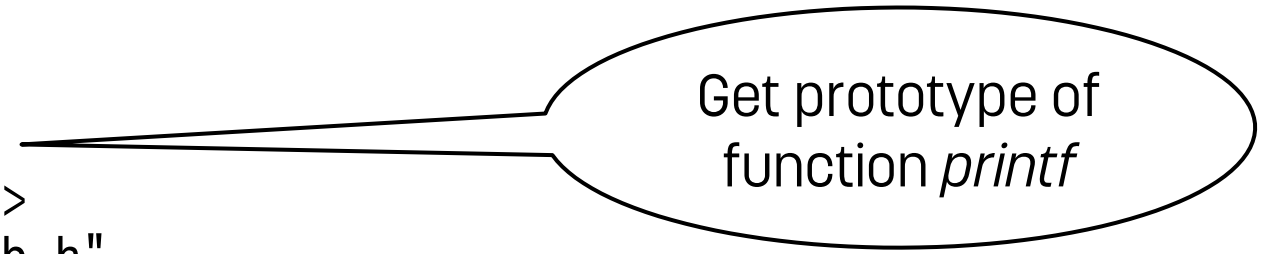
Library function in C language

```
$ cat sharedlib.c
#include <stdio.h>
#include <stdlib.h>
#include "sharedlib.h"

int sharedlibfunc(int value)
{
  printf("%d\n", value);
  return value == VALUE ? EXIT_SUCCESS : EXIT_FAILURE;
}
```

The function *sharedlibfunc* prints the argument and returns the result of comparing it with the expected value VALUE.

Library function in C language

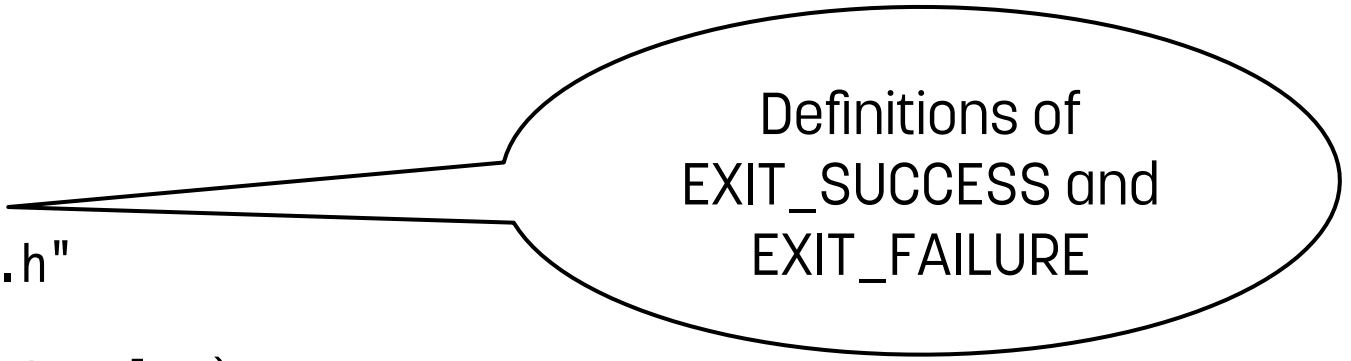
```
$ cat sharedlib.c
#include <stdio.h>
#include <stdlib.h>
#include "sharedlib.h"

int sharedlibfunc(int value)
{
  printf("%d\n", value);
  return value == VALUE ? EXIT_SUCCESS : EXIT_FAILURE;
}
```



The function *sharedlibfunc* prints the argument and returns the result of comparing it with the expected value VALUE.

Library function in C language

```
$ cat sharedlib.c
#include <stdio.h>
#include <stdlib.h>
#include "sharedlib.h"

int sharedlibfunc(int value)
{
  printf("%d\n", value);
  return value == VALUE ? EXIT_SUCCESS : EXIT_FAILURE;
}
```



The function *sharedlibfunc* prints the argument and returns the result of comparing it with the expected value VALUE.

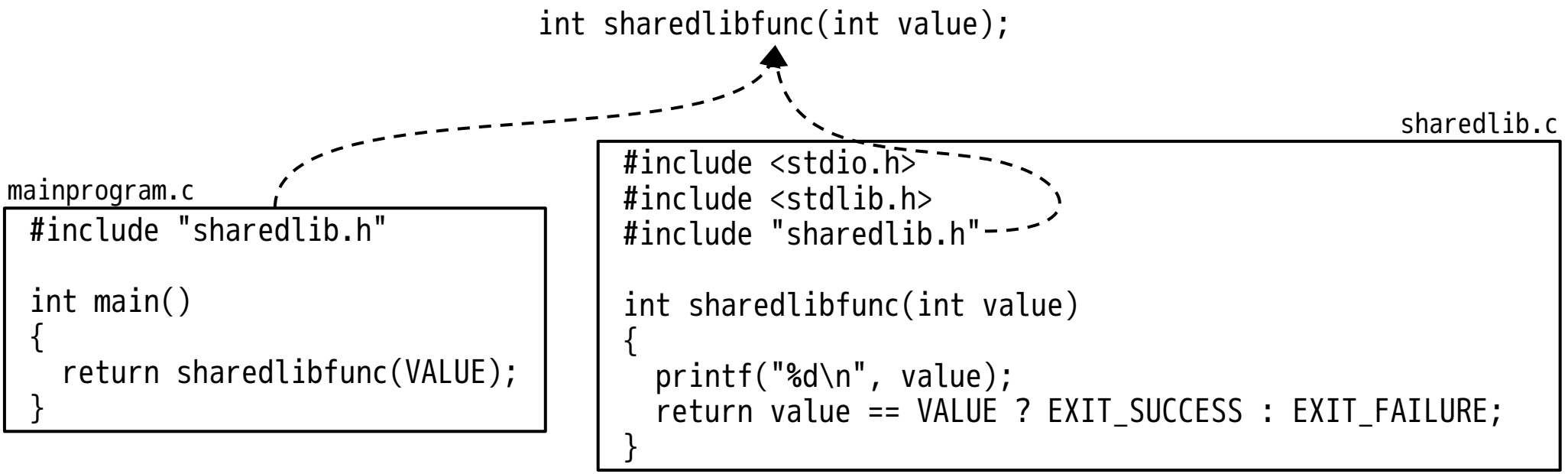
Library function in C language

```
$ cat sharedlib.c
#include <stdio.h>
#include <stdlib.h>
#include "sharedlib.h"

int sharedlibfunc(int value)
{
  printf("%d\n", value);
  return value == VALUE ? EXIT_SUCCESS : EXIT_FAILURE;
}
```

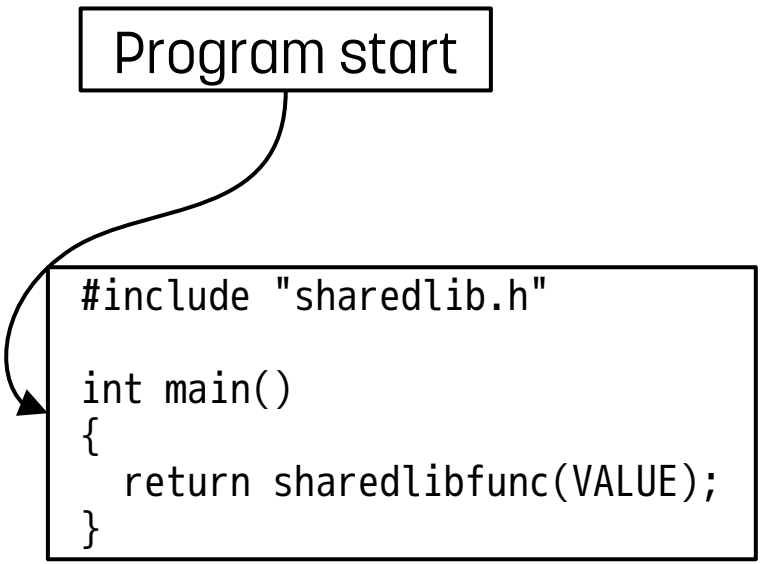
```
#define VALUE 3262
int sharedlibfunc(int value);
```

The function *sharedlibfunc* prints the argument and returns the result of comparing it with the expected value VALUE.

Sequence of calling and returning

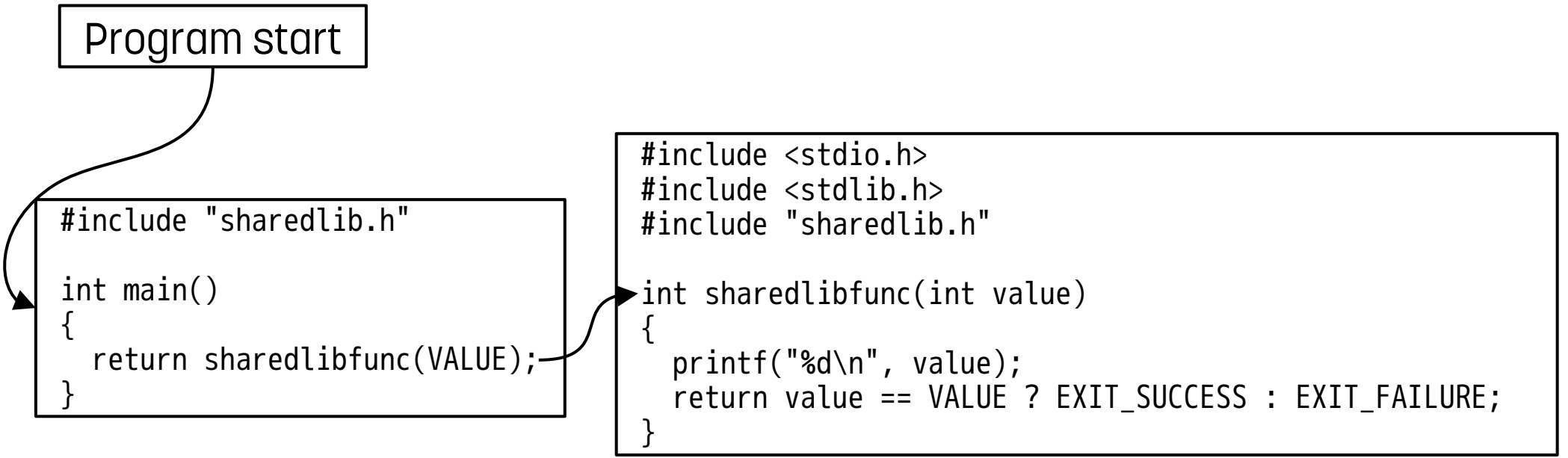


Sequence of calling and returning

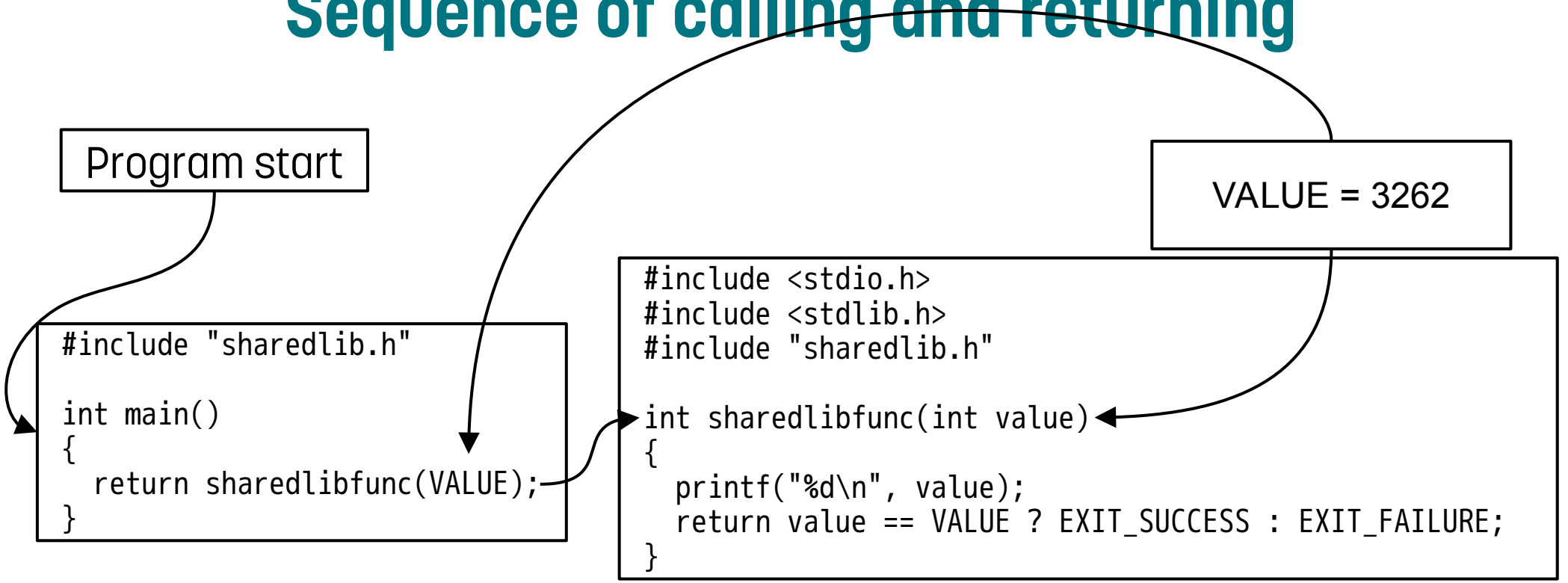



```c
#include <stdio.h>
#include <stdlib.h>
#include "sharedlib.h"

int sharedlibfunc(int value)
{
  printf("%d\n", value);
  return value == VALUE ? EXIT_SUCCESS : EXIT_FAILURE;
}
```

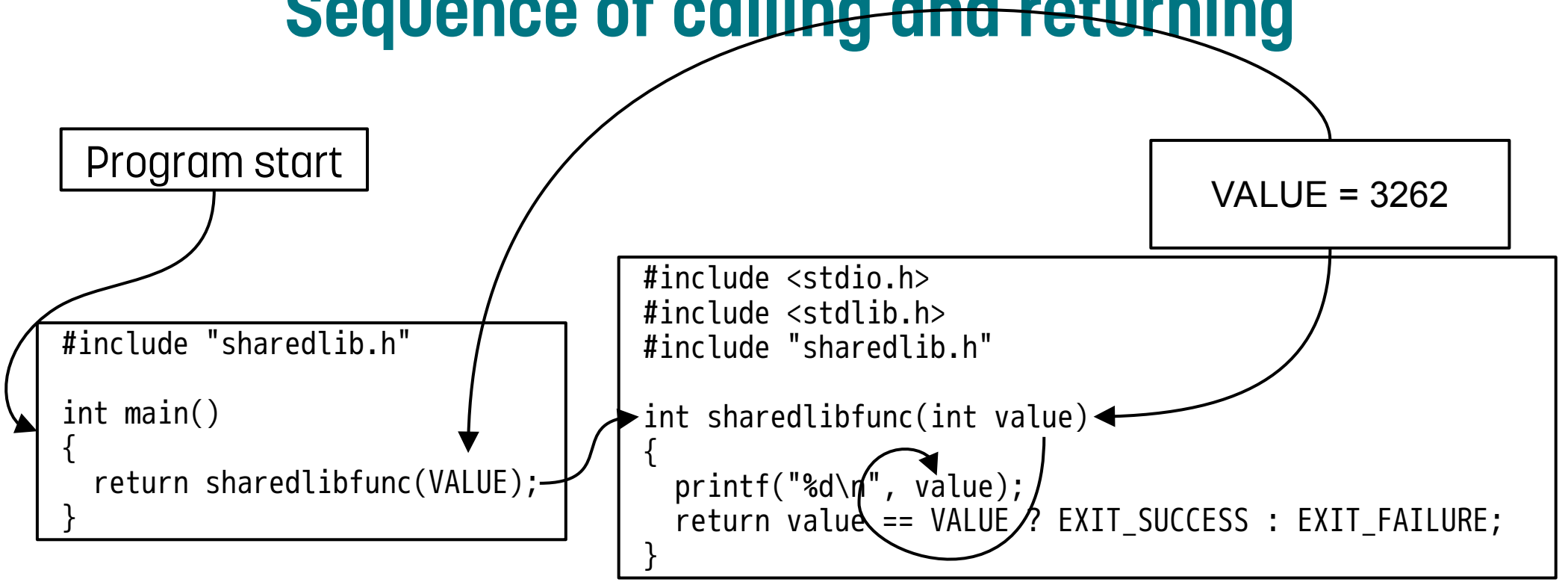
Sequence of calling and returning



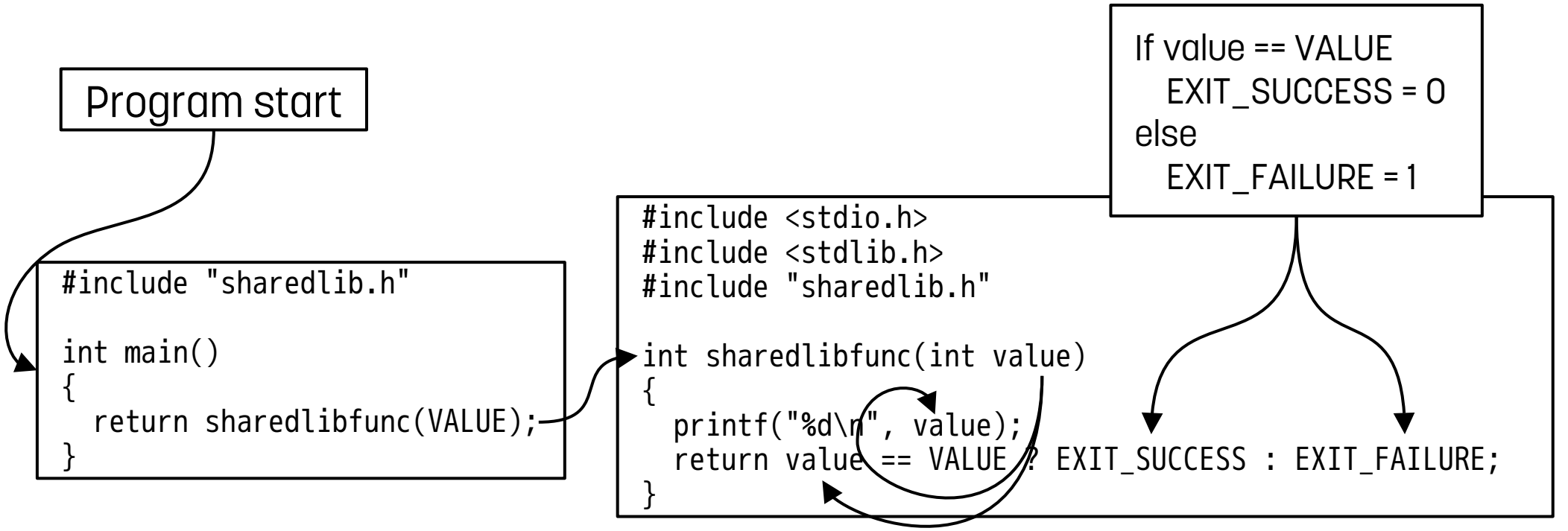
Sequence of calling and returning



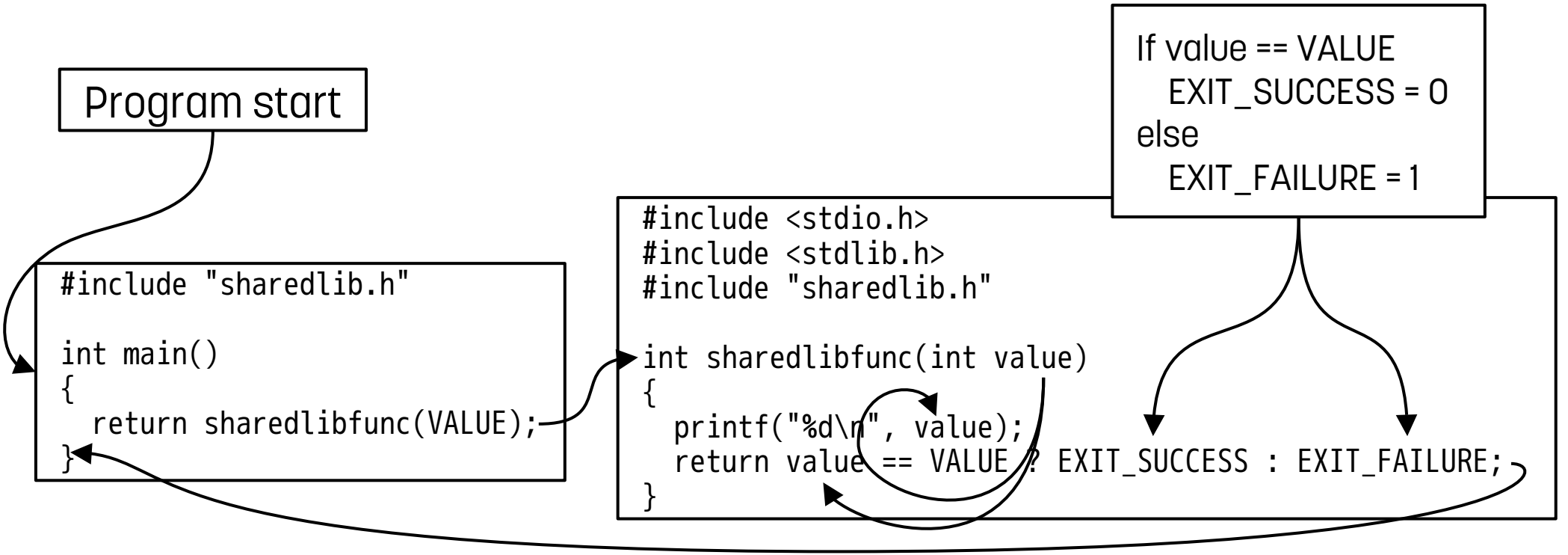
Sequence of calling and returning



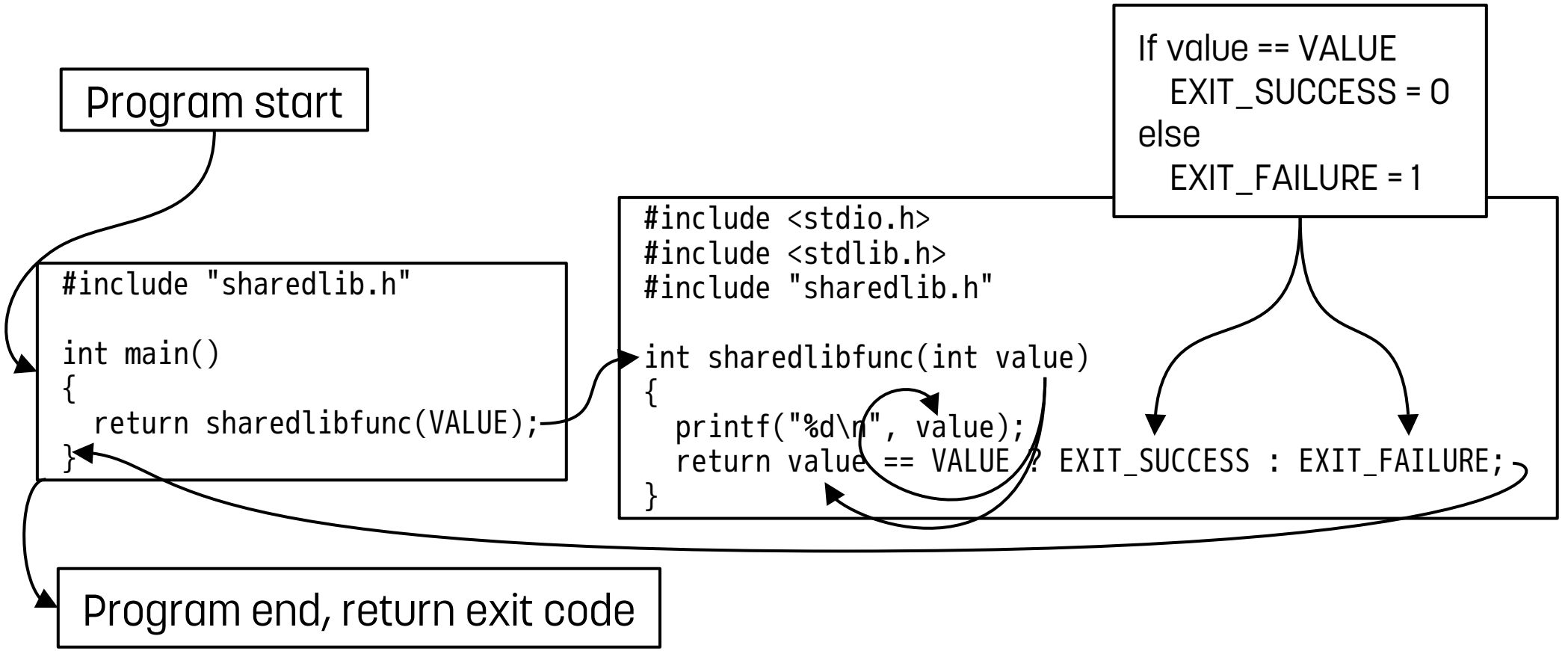
Sequence of calling and returning



Sequence of calling and returning



Sequence of calling and returning



Compile

```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o
```

Compile, create shared library

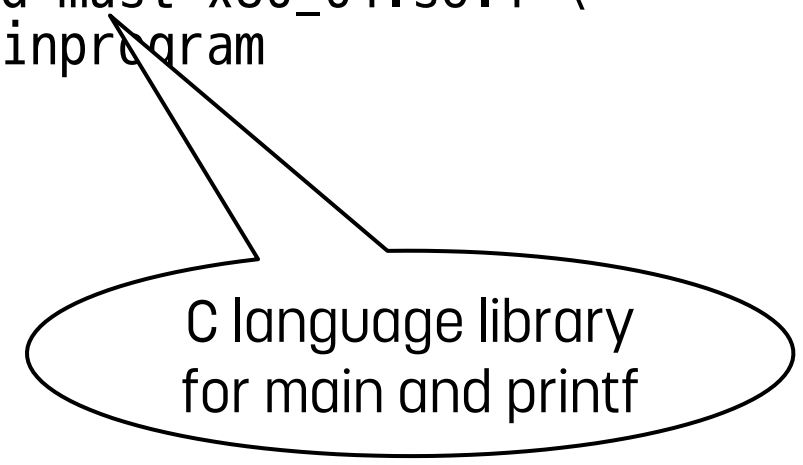
```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o
cc -shared sharedlib.o -o libsharedlib.so
```

Compile, create shared library, link

```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o

cc -shared sharedlib.o -o libsharedlib.so

cc -dynamic-linker /rootfs/usr/lib64/ld-musl-x86_64.so.1 \
   -L. -lsharedlib mainprogram.o -o mainprogram
```

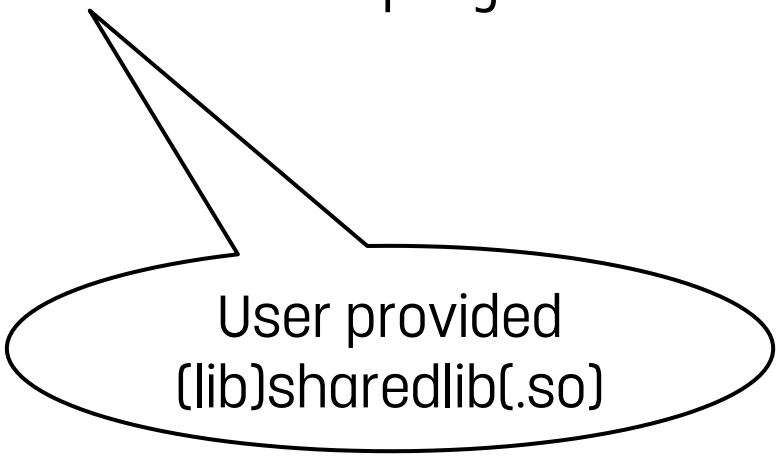
Compile, create shared library, link

```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o

cc -shared sharedlib.o -o libsharedlib.so

cc -dynamic-linker /rootfs/usr/lib64/ld-musl-x86_64.so.1 \
   -L. -lsharedlib mainprogram.o -o mainprogram
```



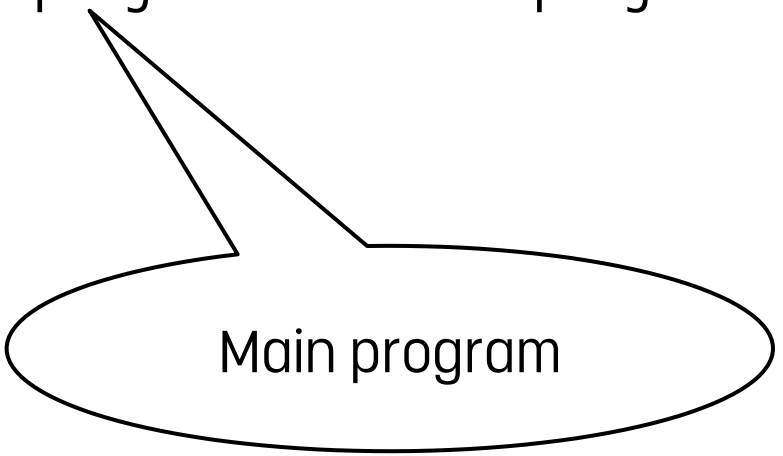
Compile, create shared library, link

```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o

cc -shared sharedlib.o -o libsharedlib.so

cc -dynamic-linker /rootfs/usr/lib64/ld-musl-x86_64.so.1 \
   -L. -lsharedlib mainprogram.o -o mainprogram
```



Compile, create shared library, link

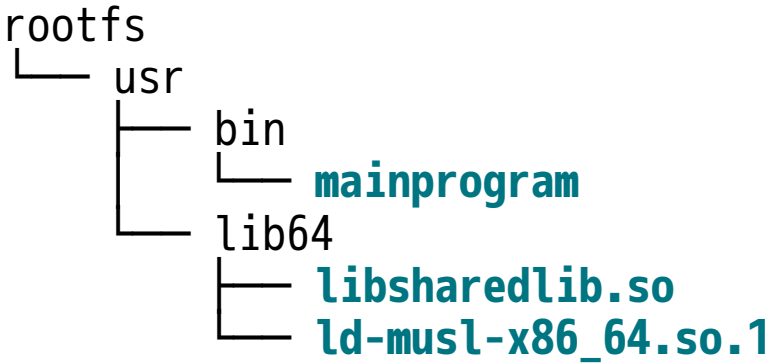
```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o

cc -shared sharedlib.o -o libsharedlib.so

cc -dynamic-linker /rootfs/usr/lib64/ld-musl-x86_64.so.1 \
   -L. -lsharedlib mainprogram.o -o mainprogram
```



Compile, create shared library, link and install

```
cc -c mainprogram.c -o mainprogram.o
cc -fPIC -c sharedlib.c -o sharedlib.o

cc -shared sharedlib.o -o libsharedlib.so

cc -dynamic-linker /rootfs/usr/lib64/ld-musl-x86_64.so.1 \
   -L. -lsharedlib mainprogram.o -o mainprogram

cp mainprogram /rootfs/usr/bin
cp libsharedlib.so /rootfs/usr/lib64
```

Program and library in the directory tree

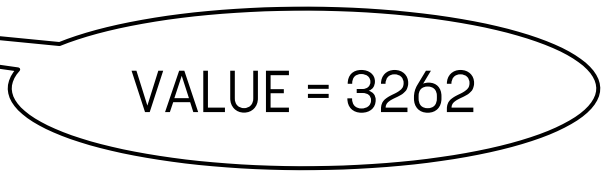


Running the program

```
$ /rootfs/usr/bin/mainprogram
3262
$ echo $?
0
```

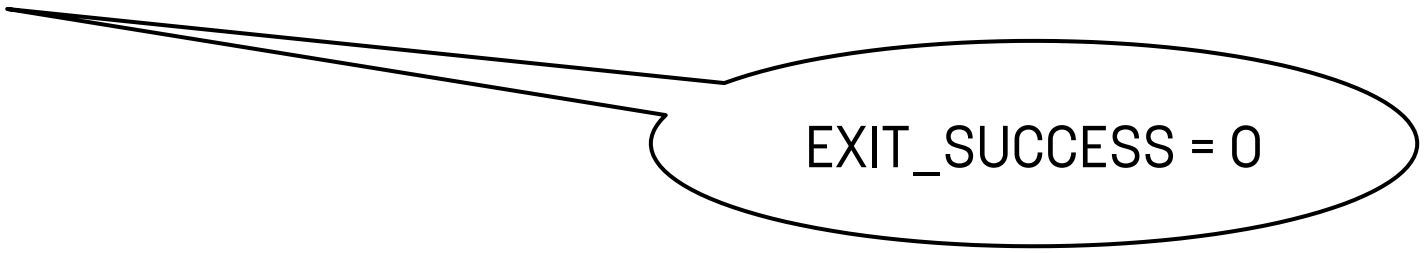
Running the program

```
$ /rootfs/usr/bin/mainprogram
3262
$ echo $?
0
```



Running the program

```
$ /rootfs/usr/bin/mainprogram
3262
$ echo $?
0
```



Scenario #2: In the compliance officer's room

The compliance officer receives the program *mainprogram* for analysis...

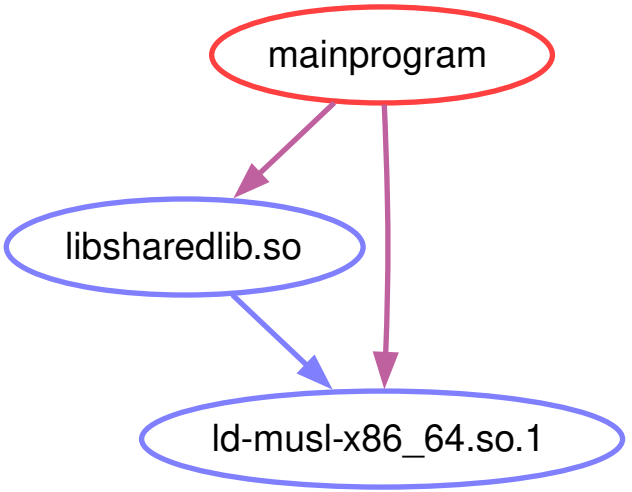
- ... but knows nothing about the previous steps. In particular the compliance officer does not know what other components are needed that may create a common work with the program. But this is required for licensing.

The compliance officer receives the program *mainprogram* for analysis...

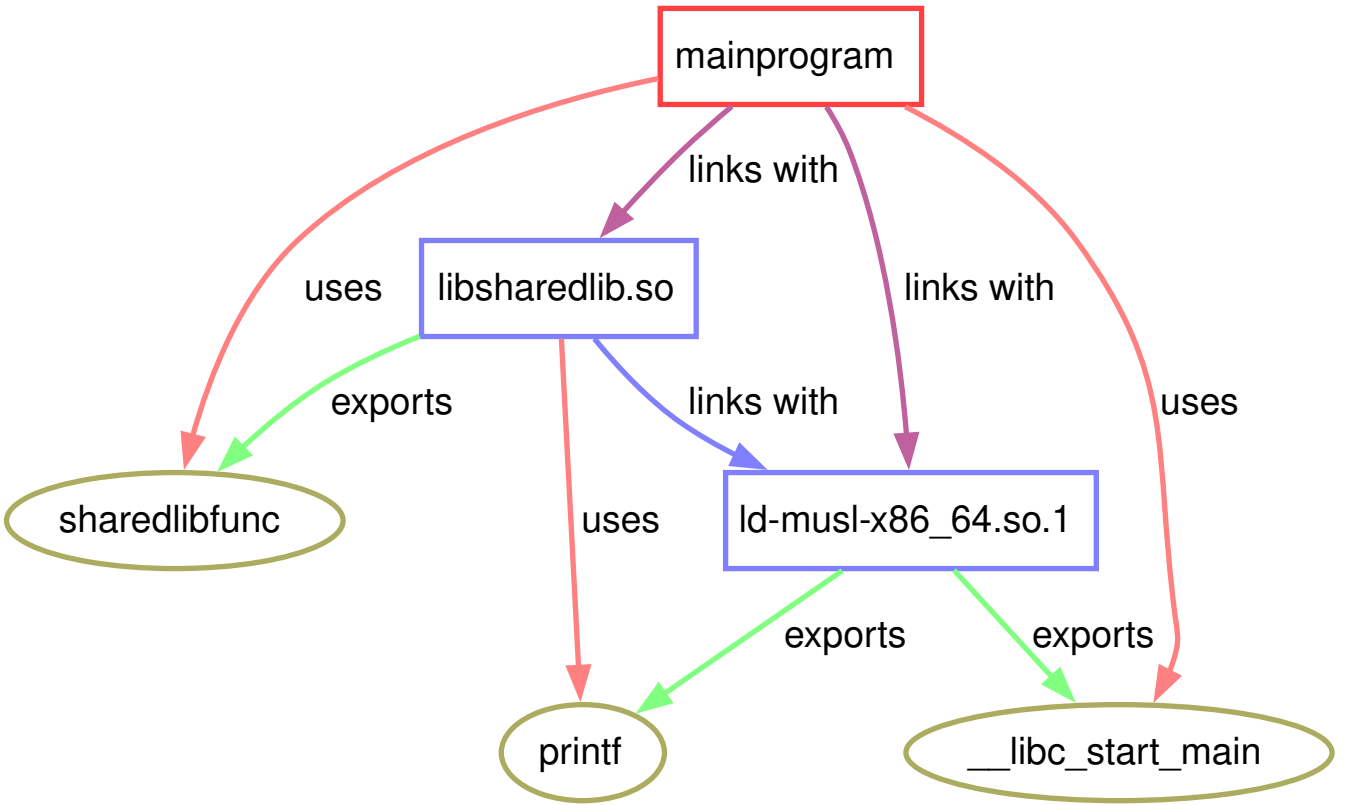
- ... but knows nothing about the previous steps. In particular the compliance officer does not know what other components are needed that may create a common work with the program. But this is required for licensing.
- The OSADL Callgraph tool can be used to analyze the link dependencies of the program:

```
./generatecypher -d /rootfs -t usr/bin/mainprogram
```

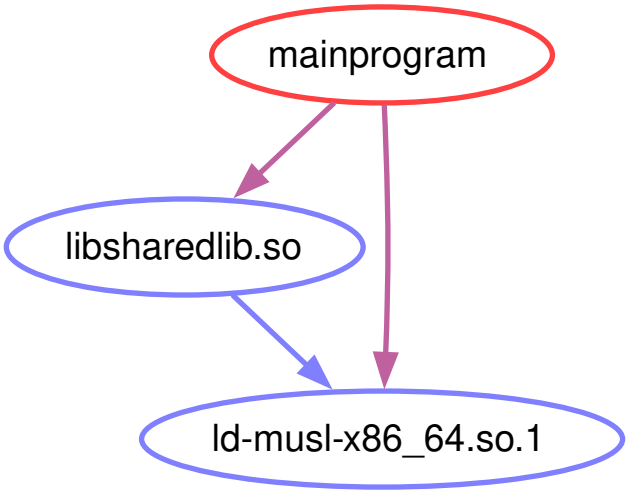
Callgraph of a program with libraries



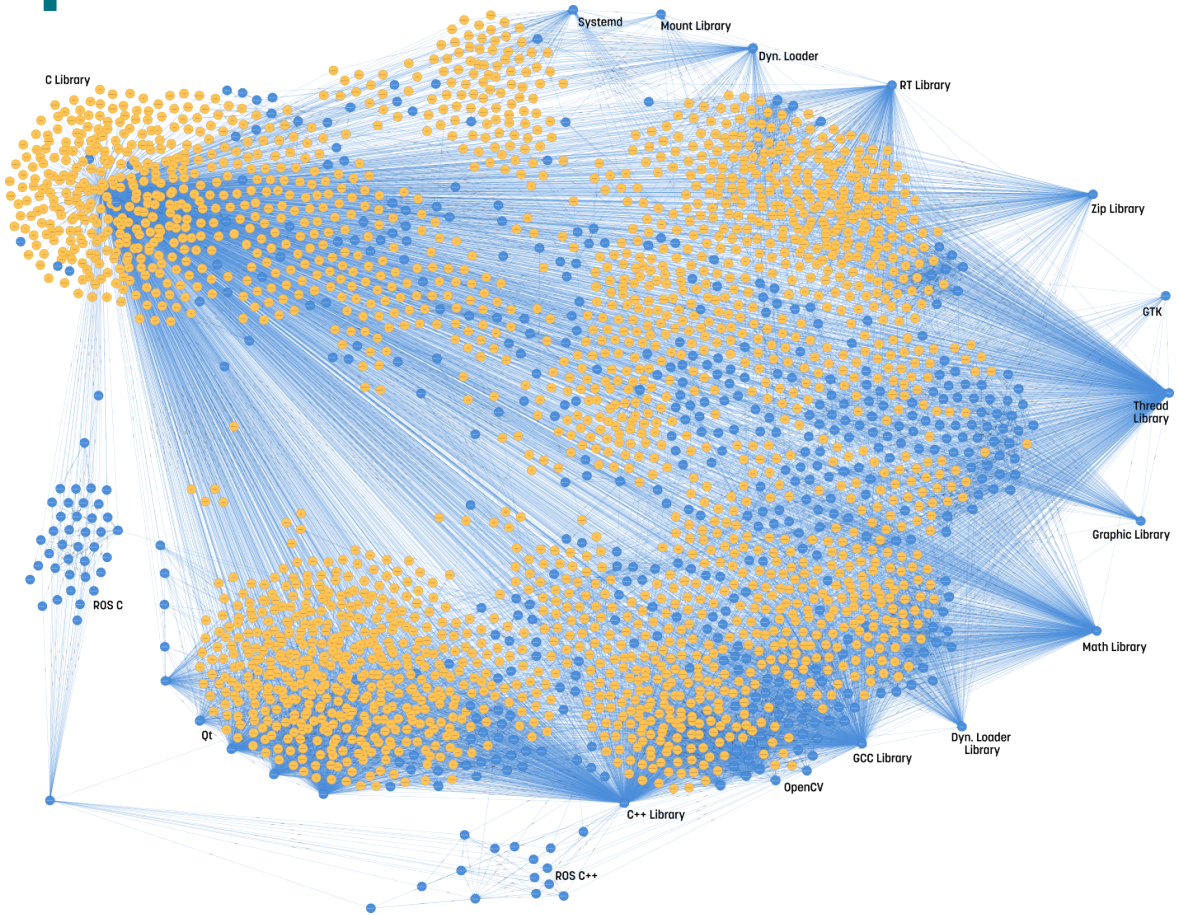
Callgraph of a program with libraries and symbols



Callgraph of a program with libraries



Callgraphs of an entire embedded system



Library tracing using *ltrace* (all in user space)

```
$ ltrace /rootfs/usr/bin/mainprogram
__libc_start_main(0x401136, 1, 0x7ffc730d9818, 0x401000 <unfinished ...>
sharedlibfunc(3262, 0x7ffc730d9818, 0x7ffc730d9828, 03262
)                        = 0
+++ exited (status 0) +++
```

Library tracing using *ltrace* (all in user space)

```
$ ltrace /rootfs/usr/bin/mainprogram
__libc_start_main(0x401136, 1, 0x7ffc730d9818, 0x401000 <unfinished ...>
sharedlibfunc(3262, 0x7ffc730d9818, 0x7ffc730d9828, 03262
)                        = 0
+++ exited (status 0) +++
```

Call a function of the C library

Library tracing using *ltrace* (all in user space)

```
$ ltrace /rootfs/usr/bin/mainprogram
__libc_start_main(0x401136, 1, 0x7ffc730d9818, 0x401000 <unfinished ...>
sharedlibfunc(3262, 0x7ffc730d9818, 0x7ffc730d9828, 03262
)                                 = 0
+++ exited (status 0) +++
```

Call a function of the C library

Call a function of the user provided library

System tracing using *strace* (kernel interface)

```
execve("/rootfs/usr/bin/mainprogram", ["/rootfs/usr/bin/mainprogram"], 0x7ffe2a7a4b40 /* 90 vars */) = 0
arch_prctl(ARCH_SET_FS, 0x7f595fdd7ae8) = 0
set_tid_address(0x7f595fdd7f50)         = 329069
brk(NULL)                               = 0x57f3000
brk(0x57f5000)                          = 0x57f5000
mmap(0x57f3000, 4096, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x57f3000
open("/rootfs/usr/lib/ld-musl-x86_64.so.1", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 3
fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
fstat(3, {st_mode=S_IFREG|0755, st_size=838368, ...}) = 0
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\336\320\7\0\0\0\0\0"..., 960) = 960
mmap(NULL, 835584, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f595fc3b000
mmap(0x7f595fc50000, 430080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0x15000) = 0x7f595fc50000
mmap(0x7f595fcb9000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED, 3, 0x7e000) = 0x7f595fcb9000
mmap(0x7f595fd02000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0xc6000) = 0x7f595fd02000
mmap(0x7f595fd04000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f595fd04000
close(3)                                = 0
munmap(0x7f595fc3b000, 835584)
```

System tracing using *strace* (kernel interface)

```
execve("/rootfs/usr/bin/mainprogram", ["/rootfs/usr/bin/mainprogram"], 0x7ffe2a7a4b40 /* 90 vars */) = 0
arch_prctl(ARCH_SET_FS, 0x7f595fdd7f50)  = 0
set_tid_address(0x7f595fdd7f50)          = 329069
brk(NULL)                                = 0x57f3000
brk(0x57f5000)                           = 0x57f5000
mmap(0x57f3000, 4096, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x57f3000
open("/rootfs/usr/lib/ld-musl-x86_64.so.1", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 3
fcntl(3, F_SETFD, FD_CLOEXEC)            = 0
fstat(3, {st_mode=S_IFREG|0755, st_size=838368, ...}) = 0
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\336\320\7\0\0\0\0\0"..., 960) = 960
mmap(NULL, 835584, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f595fc3b000
mmap(0x7f595fc50000, 430080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0x15000) = 0x7f595fc50000
mmap(0x7f595fcb9000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED, 3, 0x7e000) = 0x7f595fcb9000
mmap(0x7f595fd02000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0xc6000) = 0x7f595fd02000
mmap(0x7f595fd04000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f595fd04000
close(3)                                 = 0
munmap(0x7f595fc3b000, 835584)           = 0
```

Execute the program

System tracing using *strace* (kernel interface)

```
execve("/rootfs/usr/bin/mainprogram", ["/rootfs/usr/bin/mainprogram"], 0x7ffe2a7a4b40 /* 90 vars */) = 0
arch_prctl(ARCH_SET_FS, 0x7f595fdd7f50) = 0
set_tid_address(0x7f595fdd7f50)         = 329069
brk(NULL)                               = 0x57f3000
brk(0x57f5000)                          = 0x57f5000
mmap(0x57f3000, 4096, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x57f3000
open("/rootfs/usr/lib/ld-musl-x86_64.so.1", O_RDONLY|O_LARGEFILE|O_CLOEXEC
fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
fstat(3, {st_mode=S_IFREG|0755, st_size=838368, ...}) = 0
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\336\320\7\0\0\0
mmap(NULL, 835584, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f595fc3b000
mmap(0x7f595fc50000, 430080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0x15000) = 0x7f595fc50000
mmap(0x7f595fcb9000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED, 3, 0x7e000) = 0x7f595fcb9000
mmap(0x7f595fd02000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0xc6000) = 0x7f595fd02000
mmap(0x7f595fd04000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f595fd04000
close(3)                                = 0
munmap(0x7f595fc3b000, 835584)
```

Execute the program

Open and read the C library

System tracing using *strace* (cont'd)

```
open("/rootfs/usr/lib/libsharedlib.so", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 3
fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
fstat(3, {st_mode=S_IFREG|0775, st_size=16560, ...}) = 0
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 960) = 960
mmap(NULL, 20480, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f595fd02000
mmap(0x7f595fd03000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0x1000) = 0x7f595fd03000
mmap(0x7f595fd04000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED, 3, 0x2000) = 0x7f595fd04000
mmap(0x7f595fd05000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x2000) = 0x7f595fd05000
close(3)                                = 0
```

System tracing using *strace* (cont'd)

```
open("/rootfs/usr/lib/libsharedlib.so", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 3
fcntl(3, F_SETFD, FD_CLOEXEC)                   = 0
fstat(3, {st_mode=S_IFREG|0775, st_size=16560, ...}) = 0
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 960) = 960
mmap(NULL, 20480, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f595fd02000
mmap(0x7f595fd03000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0x1000) = 0x7f595fd03000
mmap(0x7f595fd04000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED, 3, 0x2000) = 0x7f595fd04000
mmap(0x7f595fd05000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x3000) = 0x7f595fd05000
close(3)                                        = 0
```

Open and read the user provided library

System tracing using *strace* (cont'd)

```
ioctl(1, TIOCGWINSZ, {ws_row=39, ws_col=120, ws_xpixel=0, ws_ypixel=0}) = 0
writev(1, [{iov_base="3262", iov_len=4}, {iov_base="\n", iov_len=1}], 2) = 5
exit_group(0)                           = ?
+++ exited with 0 +++
```

System tracing using *strace* (cont'd)

```
ioctl(1, TIOCGWINSZ, {ws_row=39, ws_col=120, ws_xpixel=0, ws_ypixel=0}) = 0
writev(1, [{iov_base="3262", iov_len=4}, {iov_base="\n", iov_len=1}], 2) = 5
exit_gro (0)                            = ?
+++ exited      0 +++
```

Write "3262"and "\n" (new line) to the default output channel (1)

System tracing using *strace* (cont'd)

```
ioctl(1, TIOCGWINSZ, {ws_row=39, ws_col=120, ws_xpixel=0, ws_ypixel=0}) = 0
writev(1, [{iov_base="3262", iov_len=4}, {iov_base="\n", iov_len=1}], 2) = 5
exit_gro (0)                            = ?
+++ exited      0 +++
```

Write "3262"and "\n" (new line) to the default output channel (1)

Return the number of written characters (4 + new line)

Conclusions

- **Function calls** are a common mechanism for moving frequently used operations into a shared set of software libraries to save development resources and storage space. They create derivation between the calling and the called component.

Conclusions

- **Function calls** are a common mechanism for moving frequently used operations into a shared set of software libraries to save development resources and storage space. They create derivation between the calling and the called component.
- **Callgraphs** are used to investigate link dependencies of binary executables. It is a static analysis that can be done independently from the run-time system.

Conclusions

- **Tracing** is used to obtain an in-depth analysis of the library calling interface. It is a dynamic analysis that must be run on the hardware of the system under investigation.

Conclusions

- **Tracing** is used to obtain an in-depth analysis of the library calling interface. It is a dynamic analysis that must be run on the hardware of the system under investigation.
- Usually, **Open Source** components are not investigated with respect to link dependency, **but only proprietary software** that may be linked to Open Source libraries. This is a recommended procedure to detect and prevent license incompatibilities.