

Compilers Laboratory: CS39003

C++ Program Using Library Function

```
#include <iostream>
using namespace std;
int main()                                // second0.c++
{
    cout << "My second program\n";
    return 0;
}
```

C++ Program Using System Call

```
#include <unistd.h>
#define LEN 19
int main()                                // second1.c++
{
    char str[LEN] = "My second program\n";
    write(1, str, LEN);    // STDOUT_FILENO=1
    _exit(0);
}
```

Assembly Language Translation

```
.file "second1.c++"
.text
.globl main
.type main, @function

main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
subq $32, %rsp           # 32-byte stack-frame
movq %fs:40, %rax        # Segment addressing
```

```
movq %rax, -8(%rbp)          # M[rbp-8] <-- rax
xorl %eax, %eax              # Clear eax
movl $1931508045, -32(%rbp)
    # 0111 0011 0010 0000 0111 1001 0100 1101
    # 73 20 79 4D - "s yM"
movl $1852793701, -28(%rbp)
    # 0110 1110 0110 1111 0110 0011 0110 0101
    # 6E 6F 63 65 - "noce"
movl $1919950948, -24(%rbp)
    # 0111 0010 0111 0000 0010 0000 0110 0100
    # 72 70 20 64 - "rp d"
movl $1634887535, -20(%rbp)
    # 0110 0001 0111 0010 0110 0111 0110 1111
    # 61 72 67 6F - "argo"
movw $2669, -16(%rbp)
    # 0000 1010 0110 1101
    # 0A 6D - "\nm"
```



```

movb $0, -14(%rbp)
      # 0000 0000
      # 00 - '\0'
leaq -32(%rbp), %rax      # rax <-- (rbp - 32) (str)
movl $19, %edx           # edx <-- 19 (LEN)
movq %rax, %rsi          # esi <-- rax (str)
movl $1, %edi            # edi <-- 1 (stdout)
call write               # call write
movl $0, %edi            # edi <-- 0
call _exit               # call exit
.cfi_endproc

```

.LFE0:

```

.size main, .-main
.ident "GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3"
.section .note.GNU-stack,"",@progbits

```

Using x86-64 Software Interrupt

```
#include <asm/unistd.h>
#include <syscall.h>
#define STDOUT_FILENO 1

.file "second3.S"
.section .rodata
L1:
    .string "My Second program\n"
L2:
.text
.globl _start
```

_start:

```
movl $(SYS_write), %eax # eax <-- 1 (write) parameters to write
movq $(STDOUT_FILENO), %rdi # rdi <-- 1 (stdout)
movq $L1, %rsi # rsi <-- starting address of string
movq $(L2-L1), %rdx # rdx <-- L2 - L1 string length
syscall # software interrupt
# user process requesting OS for service
movl $(SYS_exit), %eax # eax <-- 60 (exit) parameters to exit
movq $0, %rdi # rdi <-- 0
syscall # software interrupt
ret # return
```


Preprocessor - Assembler - Linker

```
$ /lib/cpp second3.S second3.s
```

```
$ as -o second3.o second3.s
```

```
$ ld second3.o
```

```
$ ./a.out
```

My second program

Simple Library: Printing an Integer

```
#define BUFF 20                                     // filename → printInt.c++
void print_int(int n) {
    char buff[BUFF], zero='0';
    int i=0, j, k, bytes;
    if(n == 0) buff[i++]=zero;
    else{
        if(n < 0) {
            buff[i++]='-';
            n = -n;
        }
        while(n){
            int dig = n%10;
            buff[i++] = (char)(zero+dig);
            n /= 10;
        }
    }
}
```

```

        if(buff[0] == '-') j = 1;
        else j = 0;
        k=i-1;
        while(j<k){
            char temp=buff[j];
            buff[j++] = buff[k];
            buff[k--] = temp;
        }
    }
    buff[i]='\n';
    bytes = i+1;
    __asm__ __volatile__ (
        "movl $1, %%eax \n\t"
        "movq $1, %%rdi \n\t"
        "syscall \n\t"
        :
        : "S"(buff), "d"(bytes)
    ) ; // $4: write, $1: on stdin
}

```

Printing an Integer

```
#ifndef _MYPRINTINT_H
#define _MYPRINTINT_H
void print_int(int);
#endif
```

//printInt.h

```
#include <iostream>
using namespace std;
#include "printInt.h"
```

```
int main()
```

// mainPrintInt.c++

```
{
    int n;
    cout << "Enter an integer: ";
    cin >> n;
    print_int(n);
    return 0;
}
```

Creating a Library

```
$ g++ -Wall -c printInt.c++
```

```
$ ar -rcs libprintInt.a printInt.o
```

```
$ g++ -Wall -c mainPrintInt.c++
```

```
$ g++ mainPrintInt.o -L. -lprintInt
```

```
$ ./a.out
```

```
Enter an integer: -123
```

```
-123
```

```
$
```

Make file

An utility program that automatically decides which part of a large software is required to be recompiled.

Target: Prerequisites

Command

- **Target:** name of a file generated by a program e.g. main.o or certain action e.g. clean.
- **Prerequisites:** files required to create the target e.g. main.c++, xyz.h etc.
- **Command:** that creates the target e.g. `c++ -Wall main.c++`.

A Simple Makefile

```
a.out:  mainPrintInt.o libprintInt.a
        c++ mainPrintInt.o -L. -lprintInt

mainPrintInt.o: mainPrintInt.cpp printInt.h
        c++ -Wall -c mainPrintInt.cpp

libprintInt.a:  printInt.o
        ar -rcs libprintInt.a printInt.o

printInt.o:     printInt.cpp printInt.h
        c++ -Wall -c printInt.cpp

clean:
        rm a.out mainPrintInt.o libprintInt.a printInt.o
```

Usage of Makefile

\$ make clean

rm a.out mainPrintInt.o libprintInt.a printInt.o

\$ make

c++ -Wall -c mainPrintInt.c++

c++ -Wall -c printInt.c++

ar -rcs libprintInt.a printInt.o

c++ mainPrintInt.o -L. -lprintInt

Creating Library

```
$ cp libprintInt.a /usr/lib
```

```
$ c++ mainPrintInt.o -lprintInt
```

Creating Shared Library

Following are steps for creating a shared library:

```
$ g++ -Wall -fPIC -c printInt.c
```

```
$ g++ -shared -Wl,-soname,libprintInt.so -o libprintInt.so printInt.o
```

Perform the following steps as superuser.

```
$ cp libprintInt.so /usr/lib/
```

```
$ ldconfig -n /usr/lib/
```

The soft-link libprint int.so.1 is created under /usr/lib. Final compilation:

```
$ g++ mainPrintInt.o -lprintInt
```

The new ./a.out does not contain the code of print_int(). But it contains code for the corresponding plt (procedure linkage table).

Disassembled second3.o

\$ objdump -d second3.o

second3.o: file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <_start>:

0:	b8 01 00 00 00	mov \$0x1,%eax
5:	48 c7 c7 01 00 00 00	mov \$0x1,%rdi
c:	48 c7 c6 00 00 00 00	mov \$0x0,%rsi
13:	48 c7 c2 13 00 00 00	mov \$0x13,%rdx
1a:	0f 05	syscall
1c:	b8 3c 00 00 00	mov \$0x3c,%eax
21:	48 c7 c7 00 00 00 00	mov \$0x0,%rdi
28:	0f 05	syscall
2a:	c3	retq

Disassembled a.out

\$ objdump -d a.out

a.out: file format elf64-x86-64

Disassembly of section .text:

000000000400078 <_start>:

400078:	b8 01 00 00 00	mov \$0x1,%eax
40007d:	48 c7 c7 01 00 00 00	mov \$0x1,%rdi
400084:	48 c7 c6 a3 00 40 00	mov \$0x4000a3,%rsi
40008b:	48 c7 c2 13 00 00 00	mov \$0x13,%rdx
400092:	0f 05	syscall
400094:	b8 3c 00 00 00	mov \$0x3c,%eax
400099:	48 c7 c7 00 00 00 00	mov \$0x0,%rdi
4000a0:	0f 05	syscall
4000a2:	c3	retq