# Department of Computer Science & Engineering

## *Compilers Laboratory: CS39003*

*Autumn Semester: 2014 - 2015*

*30th July, 2014*

## C Program Using Library Function

```
#include <stdio.h>
int main() // second0.c
{
    printf("My second program\n");
    return 0;
}
```

# Assembly Language Translation

```
        .file   "second0.c"
        .section        .rodata
.LC0:
        .string "My second program"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        andl    $-16, %esp
        subl    $16, %esp
```

```
movl    $.LC0, (%esp)
call    puts
movl    $0, %eax
leave
ret
```

## C Program Using System Call

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

## Assembly Language Translation

```
  .file   "second1.c"
  .section  .rodata
.LC0:
  .string  "My second program\n"
  .text
.globl main
  .type  main, @function
main:
  pushl  %ebp
  movl  %esp, %ebp
  andl  $-16, %esp
  subl  $32, %esp
```

```
movl  $.LC0, 28(%esp)
movl  $19, 8(%esp)
movl  28(%esp), %eax
movl  %eax, 4(%esp)
movl  $1, (%esp)
call  write
movl  $0, (%esp)
call  _exit
```

## Using Software Interrupt: x386

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

.file "second2.S"
.section        .rodata
L1:
    .string "My second program\n"
L2:

.text
.globl _start
```

```
_start:
        movl  $(SYS_write), %eax
        movl  $(STDOUT_FILENO), %ebx
        movl  $L1, %ecx
        movl  $(L2-L1), %edx
        int   $128
        movl  $(SYS_exit), %eax
        movl  $0, %ebx
        int   $128
```

## Preprocessor - Assembler - Linker

```
$ /lib/cpp -m32 second2.S second2.s
$ as --32 -o second2.o second2.s
$ ld -m elf_i386 second2.o
$ ./a.out
My second program
```

Note: `-m32`, `--32`, `-m elf_i386` are required when 32-bit x386 code is generated in a x86-64 environment.

## Using Software Interrupt: x86-64

```
#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
movq  $(STDOUT_FILENO), %rdi
movq  $L1, %rsi
movq  $(L2-L1), %rdx
syscall
movl  $(SYS_exit), %eax
movq  $0, %rdi
syscall
ret
```

## 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
void printInt(int n){ // printInt.c
    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 $4, %%eax \n\t"
          "movl $1, %%ebx \n\t"
          "int $128 \n\t"
          :
          :"c"(buff), "d"(bytes)
    ) ;  // $4: write, $1: on stdin
}
```

## Printing an Integer: print_int.h

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

## Printing an Integer: main

```
#include <stdio.h>
#include "printInt.h"
int main()
{
    int n;

    printf("Enter an integer: ");
    scanf("%d", &n);
    printInt(n);
    return 0;
}
```

## Creating a Library

```
$ cc -Wall -m32 -c printInt.c
$ ar -rcs libprintInt.a printInt.o
$ cc -Wall -m32 -c mainPrintInt.c
$ cc  -m32 mainPrintInt.o -L. -lprintInt
$ ./a.out
Enter an integer: -123
-123
$
```

# A Simple Makefile

```
a.out: mainPrintInt.o libprintInt.a
        cc -m32 mainPrintInt.o -L. -lprintInt

mainPrintInt.o: mainPrintInt.c printInt.h
                cc -Wall -m32 -c mainPrintInt.c

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

printInt.o:    printInt.c printInt.h
               cc -Wall -m32 -c printInt.c
```

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

## Object File

```
$ file second2.o
second2.o: ELF 32-bit LSB relocatable, Intel
80386, version 1 (SYSV), not stripped
$ file printInt.o
printInt.o: ELF 32-bit LSB relocatable, Intel
80386, version 1 (SYSV), not stripped
$ file mainPrintInt.o
mainPrintInt.o: ELF 32-bit LSB relocatable,
Intel 80386, version 1 (SYSV), not stripped
```

## Executable File

```
$ ld second2.o
```

a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, not stripped

## Executable File

```
$ make clean
$ make
$ file a.out
```
a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped

## Executable File

```
$ file a.out
```
a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.15, not stripped

## Disassembled second2.o

```
$ objdump -d second2.o
Disassembly of section .text:
00000000 <_start>:
   0: b8 04 00 00 00          mov    $0x4,%eax
   5: bb 01 00 00 00          mov    $0x1,%ebx
   a: b9 00 00 00 00          mov    $0x0,%ecx
   f: ba 12 00 00 00          mov    $0x12,%edx
  14: cd 80                   int    $0x80
  16: b8 01 00 00 00          mov    $0x1,%eax
  1b: bb 00 00 00 00          mov    $0x0,%ebx
  20: cd 80                   int    $0x80
```

## Disassembled a.out (second2.o)

```
$ objdump -d a.out
Disassembly of section .text:
08048054 <_start>:
 8048054: b8 04 00 00 00   mov    $0x4,%eax
 8048059: bb 01 00 00 00   mov    $0x1,%ebx
 804805e: b9 76 80 04 08   mov $0x8048076,%ecx
 8048063: ba 12 00 00 00   mov    $0x12,%edx
 8048068: cd 80            int    $0x80
 804806a: b8 01 00 00 00   mov    $0x1,%eax
 804806f: bb 00 00 00 00   mov    $0x0,%ebx
 8048074: cd 80            int    $0x80
```

## Disassemble mainPrintInt.o

```
$ objdump -d mainPrintInt.o

Disassembly of section .text:
00000000 <main>:
   0: 55                    push   %ebp
   1: 89 e5                 mov    %esp,%ebp
   3: 83 e4 f0              and    $0xfffffff0,%esp
   6: 83 ec 20              sub    $0x20,%esp
   9: b8 00 00 00 00        mov    $0x0,%eax
   e: 89 04 24              mov    %eax,(%esp)
  11: e8 fc ff ff ff        call   12 <main+0x12>
  16: b8 13 00 00 00        mov    $0x13,%eax
  1b: 8d 54 24 1c           lea    0x1c(%esp),%edx
```

```
1f: 89 54 24 04          mov    %edx,0x4(%esp)
23: 89 04 24             mov    %eax,(%esp)
26: e8 fc ff ff ff       call   27 <main+0x27>
2b: 8b 44 24 1c          mov    0x1c(%esp),%eax
2f: 89 04 24             mov    %eax,(%esp)
32: e8 fc ff ff ff       call   33 <main+0x33>
37: b8 00 00 00 00       mov    $0x0,%eax
3c: c9                   leave
3d: c3                   ret
```

**Note**

We may copy the library to a standard directory as a superuser. In that case specifying the library path is not necessary.

```
# cp libprintInt.a /usr/lib
# cc mainPrintInt.o -lprintInt
```

## Shared Library

Following are steps for creating a shared library:

```
$ cc -Wall -m32 -fPIC -c printInt.c
$
cc -m32 -shared -Wl,-soname,libprintInt.so
 -o libprintInt.so printInt.o
```

Perform the following steps as superuser.

## Shared Library

```
# cp libprintInt.so /usr/lib/
# ldconfig -n /usr/lib/
```

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

```
$ cc -m32 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).