Computer Science & Engineering Department
I. I. T. Kharagpur

Operating System: CS33007
3rd Year CSE: 5th Semester (Autumn 2006 - 2007)
Lecture III (**Relocation & Linking**)

Instructors: PDG and GB *Date:* 31st July, 2006

1. Consider the following C program:

```
#include <stdio.h>
#define MAX 100
#define N 5

int a1[MAX] = {10, 20, 30, 40, 50} ;
int a2[MAX] ;

void dataCopy(int) ;

int main(){
          int i ;

          dataCopy(N) ;
          for(i=0; i<N; ++i) printf("%d ", a2[i]) ;
          printf("\n") ;

// Print addresses:
          printf("\t\t\tmain: %p\n", main) ;
          printf("\t\t\tdataCopy: %p\n", dataCopy) ;
          printf("\ta1: %p\n", a1) ;
          printf("\ta2: %p\n", a2) ;
          printf("\t\t\tprintf: %p\n", printf) ;
          return 0 ;
}

void dataCopy(int n){
          int i ;

          for(i=0; i<n; ++i) a2[i] = a1[i] ;
}
```

2. Compile to executable module:

```
$ cc -Wall reloc1.c
$ file a.out
```

```
a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, dynamically linked (uses shared libs),
not stripped
```

3. Execute the code:

```
$ a.out
10 20 30 40 50
                        main: 0x8048368
                        dataCopy: 0x804843f
        a1: 0x80496c0
        a2: 0x8049880
                        printf: 0x80482b0
$
```

4. Change the order of the functions:

```
#include <stdio.h>
#define MAX 100
#define N 5

int a2[MAX] ;
int a1[MAX] = {10, 20, 30, 40, 50} ;

void dataCopy(int n){
          int i ;

          for(i=0; i<n; ++i) a2[i] = a1[i] ;
}

int main(){
          int i ;

          dataCopy(N) ;
          for(i=0; i<N; ++i) printf("%d ", a2[i]) ;
          printf("\n") ;

// Print addresses:
          printf("\t\t\tmain: %p\n", main) ;
          printf("\t\t\tdataCopy: %p\n", dataCopy) ;
          printf("\ta1: %p\n", a1) ;
          printf("\ta2: %p\n", a2) ;
          printf("\t\t\tprintf: %p\n", printf) ;
          return 0 ;
}
```

5. The output is -

```
$ cc -Wall reloc2.c
$ a.out
10 20 30 40 50
                        main: 0x804839a
                        dataCopy: 0x8048368
        a1: 0x80496c0
        a2: 0x8049880
                        printf: 0x80482b0
$
```

The starting addresses of '`main`' and '`dataCopy`' changes. But there is no change in the starting addresses of arrays.

6. Use '`readelf`' on '`a.out`'

```
$ cc -Wall reloc2.c
$ readelf -s a.out

Symbol table '.symtab' contains 73 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
    48: 080496c0   400 OBJECT  GLOBAL DEFAULT   22 a1
    52: 08048368    50 FUNC    GLOBAL DEFAULT   12 dataCopy
    56: 0804839a   212 FUNC    GLOBAL DEFAULT   12 main
    57: 08049880   400 OBJECT  GLOBAL DEFAULT   23 a2
    61: 080482b0    54 FUNC    GLOBAL DEFAULT  UND printf@@GLIBC_2.0
```

7. Machine code in '`a.out`'

```
$ objdump -d a.out
```

8. Split the program in two files: `reloc3.c`

```
#include <stdio.h>
#define MAX 100
#define N 5

int a2[MAX] ;
extern int a1[] ;
void dataCopy(int) ;

int main(){
          int i ;

          dataCopy(N) ;
          for(i=0; i<N; ++i) printf("%d ", a2[i]) ;
          printf("\n") ;
```

```
// Print addresses:
          printf("\t\t\tmain: %p\n", main) ;
          printf("\t\t\tdataCopy: %p\n", dataCopy) ;
          printf("\ta1: %p\n", a1) ;
          printf("\ta2: %p\n", a2) ;
          printf("\t\t\tprintf: %p\n", printf) ;
          return 0 ;
}
```

and the other one is

```
#include <stdio.h>
#define MAX 100
#define N 5

extern int a2[] ;
int a1[MAX] = {10, 20, 30, 40, 50} ;

void dataCopy(int n){
          int i ;

          for(i=0; i<n; ++i) a2[i] = a1[i] ;
}
```

9. Compile them to object module:

```
$ cc -Wall -c reloc3.c
$ cc -Wall -c reloc4.c
$ ls -l reloc*.o
-rw-rw-r--  1 goutam goutam 1324 Jul 28 16:54 reloc3.o
-rw-rw-r--  1 goutam goutam 1256 Jul 28 16:54 reloc4.o
$ file reloc*.o
reloc3.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV),
not stripped
reloc4.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), i
not stripped
$
```

10. Look into the symbol table:

```
$ readelf -s reloc3.o

Symbol table '.symtab' contains 13 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     8: 00000000   215 FUNC    GLOBAL DEFAULT    1 main
     9: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND dataCopy
    10: 00000020   400 OBJECT  GLOBAL DEFAULT  COM a2
    11: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND printf
    12: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND a1
```

'`dataCopy`', '`printf`' and '`a1`' are undefined in reloc3.o.

```
$ readelf -s reloc4.o

Symbol table '.symtab' contains 10 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     7: 00000000   400 OBJECT  GLOBAL DEFAULT    3 a1
     8: 00000000    50 FUNC    GLOBAL DEFAULT    1 dataCopy
     9: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND a2
```

'`a2`' is undefined in '`reloc4.o`'

11. `reloc3.o` has the machine code for '`main`' and the code starts from addrss zero.

    ```
    $ objdump -d reloc3.o
    ```

    Similarly `reloc4.o` has gthe machine code for 'dataCopy'. This code also starts from address zero.

    There are relocation points in both the codes. prepare the executable module and compare.

12. Compile to execution module

    ```
    $ cc reloc3.o reloc4.o
    $ ls -l reloc*.o a.out
    -rwxrwxr-x  1 goutam goutam 5584 Jul 28 17:11 a.out
    -rw-rw-r--  1 goutam goutam 1324 Jul 28 16:54 reloc3.o
    -rw-rw-r--  1 goutam goutam 1256 Jul 28 16:54 reloc4.o
    ```

13. Code from '`reloc3.o`' and '`a.out`'.

    ```
    00000000 <main>:
           0:  55                      push   %ebp
    08048368 <main>:
     8048368:  55                      push   %ebp
    ....................................................
         21:   e8 fc ff ff ff          call   22 <main+0x22>
    8048389:   e8 b2 00 00 00          call   8048440 <dataCopy>
    ....................................................
         5f:   e8 fc ff ff ff          call   60 <main+0x60>
    80483b0:   e8 fb fe ff ff          call   80482b0 <printf@plt>
    ```

14. Compile '`a.out`' by interchanging the position of the object modules.

    ```
    $ cc reloc4.o reloc3.o
    $ objdump -d a.out
    ```

15. Code from '`reloc3.o`' and '`a.out`' - different relocation

    ```
         21:   e8 fc ff ff ff          call   22 <main+0x22>
    80483bd:   e8 a6 ff ff ff          call   8048368 <dataCopy>
    ```