

# Édition de liens

Ensimag 1A Apprentissage

2011

## 1 Programme unique

### 1.1 Listing d'assemblage

offset codop

```
.section .rodata

0000    25 73 20 25  sortie: .asciz "%s %s %c%c%c%c%c%c %s%c %s%c\n"
        73 20 25 63
        25 63 25 63
        25 63 20 25
        63 25 63 25
0022    72 64 73      car1:  .byte 'r','d','s'
0025    31 32 33 34  ch1:    .asciz "12345678"
        35 36 37 38
        00
002e    31 32 33 34  ch2:    .asciz "1234567"
        35 36 37 00

.data

0000    75 70 6F 65  car2:    .byte 'u','p','o','e'
0004    31 32 33 34  ch3:    .asciz "1234"
        00
0009    31 32 33 34  ch4:    .asciz "12345678911"
        35 36 37 38
        39 31 31 00

.globl main
0015    00 00 00      .text
0000    55            main:  pushl  %ebp
0001    89 E5         movl  %esp,%ebp
0003    31 C0         xorl  %eax,%eax
0005    A0 24 00 00   movb  car1+2, %al    # @ de s = car1+2 = 24
        00
000a    50            pushl  %eax
000b    68 09 00 00   pushl  $ch4
        00
0010    50            pushl  %eax
0011    68 25 00 00   pushl  $ch1
        00
```

```

0016    C9                leave
0018    C3                ret

```

## 1.2 Après la Compilation :

### Table des symboles

Defined symbol

<u>Section:offset</u>	<u>name</u>
-----------------------	-------------

```

.rodata:00000000  sortie
.rodata:00000022  car1
.rodata:00000025  ch1
.rodata:0000002e  ch2
.data:00000000    car2
.data:00000004    ch3
.data:00000009    ch4
.text:00000000    main

```

Undefined symbol

### Table des relogements (relocation table)

<u>Offset</u>	<u>name</u>	<u>section</u>	<u>value</u>
00000006	car1	.rodata	24
0000000C	ch4	.data	09
00000012	ch1	.rodata	25

## 1.3 binaire apres chargement :

Table des symboles

Defined symbol

<u>offset</u>	<u>name</u>
---------------	-------------

```

08048528  sortie # debut zone .rodata
0804854A  car1   # = sortie+22 = .rodata+22
0804854D  ch1
08048556  ch2
08049570  car2   # debut zone data
08049574  ch3    # = car2+4
08049579  ch4
08048460  main   # début zone .text

```

zone text

```

08048460 55                push %ebp
08048461 89 E5            mov %esp,%ebp
08048463 31 C0            xorl %eax,%eax
08048465 A0 4C 85 04      movb %0x0804854C, %al    # 08048528+24 ...
                                # en little-endian
                                08
0804846A 50                pushl %eax
0804846B 68 79 95 04      pushl %0x08049579
                                08
08048470 50                pushl %eax
08048471 68 4D 85 04      pushl %0x0804854D
                                08

```

```
08048476 C9          leave
08048477 C3          ret
```

## 1.4 Édition de liens avec plusieurs fichiers

Pour étudier l'édition de liens, nous prenons une application jouet composée de deux fichiers qui sont assemblés séparément. Une édition de liens (`ld`) est ensuite effectuée pour assembler les fichiers binaires en un seul fichier `ensemble.o`. Ce fichier est ensuite lié avec le système et chargé (`gcc`) pour constituer l'application jouet. Le `makefile` utilisé pour générer l'application est le suivant :

```
imprim.o: imprim.s
    as -a=imprim.l imprim.s -o imprim.o
hello.o: hello.s
    as -a=hello.l hello.s -o hello.o
ensemble.o: imprim.o hello.o
    ld -r imprim.o hello.o -o ensemble.o
ensemble: ensemble.o
    gcc ensemble.o -o ensemble
```

Le travail à effectuer consiste à identifier dans les listings d'assemblage (§1.5) les instructions devant être relogées et celles qui appellent des variables non définies dans le module. Dans un deuxième temps on essaiera de retrouver dans les sections §1.6, §1.7.2 et §1.8, comment l'édition de lien et le chargement ont modifié les instructions concernées. Les fichiers générés par l'assemblage et l'édition de liens sont au format ELF. Pour certains affichages, nous utilisons la commande `readelf` permettant de visualiser de façon sélective et «agréable» le contenu des fichiers binaires.

## 1.5 Listings d'assemblage

### 1.5.1 Fichier `hello.l`

```
1                                .section .rodata
2 0000 48656C6C          message: .asciz "Hello, world! \n"
2          6F2C2077
2          6F726C64
2          21200A00
3                                .text
4                                .globl main
5 0000 55              main:  pushl %ebp
6 0001 89E5            movl %esp, %ebp
7 0003 68000000        pushl $message
7          00
8 0008 E8FCFFFF        call imprim
8          FF
9 000d 83C404          addl $4, %esp
10 0010 C9              leave
11 0011 C3              ret
12 0012 89F6
DEFINED SYMBOLS
```

```

                hello.s:2      .rodata:00000000 message
                hello.s:5      .text:00000000 main
UNDEFINED SYMBOLS
imprim
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(89)readelf -r hello.o
Relocation section '.rel.text' at offset 0x298 contains 2 entries:
  Offset      Info  Type           Symbol's Value  Symbol's Name
  00000004    00401 R_386_32       00000000      .rodata
  00000009    00702 R_386_PC32     00000000      imprim

```

## 1.5.2 Fichier imprim.1

```

                                # Deplacement parametre sur pile
1                                .equ CHAINE,8
2                                .section .rodata
3 0000 257300      sortie: .asciz "%s"
4                                .text
5                                .globl imprim
6 0000 55          imprim: pushl %ebp
7 0001 89E5        movl %esp, %ebp
8 0003 53          pushl %ebx
                        # Adr. chaine
9 0004 8B5D08      movl CHAINE(%ebp), %ebx
10 0007 53         pushl %ebx
11 0008 68000000   pushl $sortie
11          00
12 000d E8FCFFFF   call printf
12          FF
13 0012 83C408     addl $8, %esp
14 0015 5B         popl %ebx
15 0016 C9        leave
16 0017 C3        ret
DEFINED SYMBOLS
                imprim.s:1      *ABS*:00000008 CHAINE
                imprim.s:3      .rodata:00000000 sortie
                imprim.s:6      .text:00000000 imprim
UNDEFINED SYMBOLS
printf
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(90)readelf -r imprim.o
Relocation section '.rel.text' at offset 0x2a8 contains 2 entries:
  Offset      Info  Type           Symbol's Value  Symbol's Name
  00000009    00501 R_386_32       00000000      .rodata
  0000000e    00802 R_386_PC32     00000000      printf

```

## 1.6 Binaire généré par l'édition de liens

### 1.6.1 Section .text

```
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(92)objdump -d ensemble.o
ensemble.o:      file format elf32-i386
Disassembly of section .text:
00000000 <imprim>:
   0:  55                push    %ebp
   1:  89 e5             mov     %esp,%ebp
   3:  53                push    %ebx
   4:  8b 5d 08          mov     0x8(%ebp),%ebx
   7:  53                push    %ebx
   8:  68 00 00 00 00    push    $0x0 # pushl $sortie
  d:  e8 fc ff ff ff    call   e <imprim+0xe># call printf
12:  83 c4 08          add     $0x8,%esp
15:  5b                pop     %ebx
16:  c9                leave
17:  c3                ret
00000018 <main>:
18:  55                push    %ebp
19:  89 e5             mov     %esp,%ebp
1b:  68 03 00 00 00    push    $0x3 # pushl $message
20:  e8 fc ff ff ff    call   21 <main+0x9># call imprim
25:  83 c4 04          add     $0x4,%esp
28:  c9                leave
29:  c3                ret
2a:  89 f6             mov     %esi,%esi
```

### 1.6.2 Table des symboles

Ndx est le numéro de la section dans laquelle est défini le symbole. Pour le fichier binaire ensemble.o, les sections référencées ci-dessous sont 1 pour .text, 3 pour .rodata, 4 pour .data et 5 pour .bss. ABS indique un symbole absolu (constante) tandis que UND indique un symbole indéfini.

```
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(93)readelf -s ensemble.o
Symbol table '.symtab' contains 15 entries:
   Num:      Value          Size Type      Bind     Vis      Ndx Name
   0: 00000000           0 NOTYPE   LOCAL   DEFAULT  UND
   1: 00000000           0 SECTION LOCAL   DEFAULT    1
   .....
   8: 00000000           0 SECTION LOCAL   DEFAULT    8
   9: 00000008           0 NOTYPE   LOCAL   DEFAULT  ABS CHAINE
  10: 00000000           0 NOTYPE   LOCAL   DEFAULT    3 sortie
  11: 00000003           0 NOTYPE   LOCAL   DEFAULT    3 message
  12: 00000000           0 NOTYPE   GLOBAL  DEFAULT  UND printf
  13: 00000018           0 NOTYPE   GLOBAL  DEFAULT    1 main
```

```
14: 00000000      0 NOTYPE GLOBAL DEFAULT      1 imprim
```

### 1.6.3 Table des relogements

```
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(95)readelf -r ensemble.o
Relocation section '.rel.text' at offset 0x21c contains 4 entries:
```

Offset	Info	Type	Symbol's Value	Symbol's Name
00000009	00301	R_386_32	00000000	.rodata
0000000e	00c02	R_386_PC32	00000000	printf
0000001c	00301	R_386_32	00000000	.rodata
00000021	00e02	R_386_PC32	00000000	imprim

## 1.7 Binaire après chargement

Si on affiche l'intégralité du fichier binaire `ensemble` — en utilisant une commande telle que `objdump`, `hexdump` ou `od` — nous pourrions constater qu'il contient une table de l'ensemble des symboles, le code système permettant de lancer et d'arrêter le programme utilisateur ainsi que le programme utilisateur, ainsi que les différentes sections de données `.rodata`, `.data` et `.bss`. Pour des raisons de lisibilité, tous les constituants du fichier binaire `ensemble` sont donnés séparément.

### 1.7.1 Table des symboles

De nombreux symboles font référence au code «système» qui est lié au code utilisateur. On ne donne ci-dessous que l'extrait de la table des symboles concernant le programme utilisateur.

```
Symbol table '.symtab' contains 80 entries:
```

Num:	Value	Size	Type	Bind	Vis	Ndx	Name
0:	00000000	0	NOTYPE	LOCAL	DEFAULT	UND	
1:	080480f4	0	SECTION	LOCAL	DEFAULT	1	
.....							
33:	00000000	0	FILE	LOCAL	DEFAULT	ABS	init.c
.....							
57:	00000008	0	NOTYPE	LOCAL	DEFAULT	ABS	CHAINE
58:	080484f8	0	NOTYPE	LOCAL	DEFAULT	14	sortie
59:	080484fb	0	NOTYPE	LOCAL	DEFAULT	14	message
.....							
67:	08048478	0	NOTYPE	GLOBAL	DEFAULT	12	main
.....							
77:	08048460	0	NOTYPE	GLOBAL	DEFAULT	12	imprim

### 1.7.2 Section .text

```
chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(98)objdump -d ensemble
ensemble:      file format elf32-i386
Disassembly of section .init:
080482e4 <_init>:
80482e4:      55                push    %ebp
```

```

80482e5:      89 e5          mov     %esp,%ebp
80482e7:      83 ec 08        sub     $0x8,%esp
80482ea:      e8 95 00 00 00  call    8048384 <call_gmon_start>
.....
804833c:      ff 25 48 95 04 08  jmp     *0x8049548 # Adresse printf
.....
08048460 <imprim>:
8048460:      55             push    %ebp
8048461:      89 e5          mov     %esp,%ebp
8048463:      53             push    %ebx
8048464:      8b 5d 08        mov     0x8(%ebp),%ebx
8048467:      53             push    %ebx
8048468:      68 f8 84 04 08  push    $0x80484f8 # $sortie
804846d:      e8 ca fe ff ff  call    804833c <_init+0x58> # printf
8048472:      83 c4 08        add     $0x8,%esp
8048475:      5b             pop     %ebx
8048476:      c9             leave
8048477:      c3             ret
08048478 <main>:
8048478:      55             push    %ebp
8048479:      89 e5          mov     %esp,%ebp
804847b:      68 fb 84 04 08  push    $0x80484fb # $message
8048480:      e8 db ff ff ff  call    8048460 <imprim>
8048485:      83 c4 04        add     $0x4,%esp
8048488:      c9             leave
8048489:      c3             ret
804848a:      89 f6          mov     %esi,%esi
804848c:      90             nop
804848d:      90             nop
804848e:      90             nop
804848f:      90             nop
.....
080484d0 <_fini>:
80484d0:      55             push    %ebp
80484d1:      89 e5          mov     %esp,%ebp
80484d3:      53             push    %ebx
80484d4:      52             push    %edx

```

### 1.7.3 Section .rodata

```

chassin@ensilogic ~/LOGBASE/PROGRAMMES/EDLINK(107)readelf -x 14 ensemble
Hex dump of section '.rodata':
 0x080484f0 6f6c6c65 48007325 00020001 00000003 .....%s.Hello
 0x08048500                000a20 21646c72 6f77202c , world! ..

```

## 1.8 Binaire exécuté

```
(gdb) x/30i imprim
0x8048460 <imprim>:      push    %ebp
0x8048461 <imprim+1>:    mov     %esp,%ebp
0x8048463 <imprim+3>:    push    %ebx
0x8048464 <imprim+4>:    mov     0x8(%ebp),%ebx
0x8048467 <imprim+7>:    push    %ebx
0x8048468 <imprim+8>:    push    $0x80484f8
0x804846d <imprim+13>:   call   0x804833c <printf>
0x8048472 <imprim+18>:   add     $0x8,%esp
0x8048475 <imprim+21>:   pop     %ebx
0x8048476 <imprim+22>:   leave
0x8048477 <imprim+23>:   ret
0x8048478 <main>:        push    %ebp
0x8048479 <main+1>:        mov     %esp,%ebp
0x804847b <main+3>:        push    $0x80484fb
0x8048480 <main+8>:        call   0x8048460 <imprim>
0x8048485 <main+13>:     add     $0x4,%esp
0x8048488 <main+16>:     leave
0x8048489 <main+17>:     ret
0x804848a <main+18>:     mov     %esi,%esi
0x804848c <main+20>:     nop
0x804848d <main+21>:     nop
0x804848e <main+22>:     nop
0x804848f <main+23>:     nop
0x8048490 <__do_global_ctors_aux>:      push    %ebp
```

```
(gdb) x/50xb imprim
0x8048460 <imprim>:
    0x55    0x89    0xe5    0x53    0x8b    0x5d    0x08    0x53
0x8048468 <imprim+8>:
    0x68    0xf8    0x84    0x04    0x08    0xe8    0xca    0xfe
0x8048470 <imprim+16>:
    0xff    0xff    0x83    0xc4    0x08    0x5b    0xc9    0xc3
0x8048478 <main>:
    0x55    0x89    0xe5    0x68    0xfb    0x84    0x04    0x08
0x8048480 <main+8>:
    0xe8    0xdb    0xff    0xff    0xff    0x83    0xc4    0x04
0x8048488 <main+16>:
    0xc9    0xc3    0x89    0xf6    0x90    0x90    0x90    0x90
0x8048490 <__do_global_ctors_aux>:      0x55    0x89
```