Linux Kəsilmə Funksiyaları

Aşağıdakı məntdə Linux nüvəsinin kəsilmə funksiyalarının təyin olunması müzakirə olunur.

Metod.

1. Sistem yaddaşının surəti alınır.
2. Kəsilmə cədvəli yaddaşdan ikili fayla köçürülür.
3. C dilində yazılmış proqramın köməyi ilə ikili faylda olan hər-bir kəsilmənin ünvanı hesablanır və System.map faylından müvafiq ünvanlı funksiyanın adı axtarılır.

İstifadə olunan vasitələr.

1. VirtualBox proqramı
2. Virtual maşına yüklənmiş hər-hansı Linux distributivi
3. Linux nüvəsinin mənbə kodu

1 Sistem yaddaşının surətinin alınması.

Əvvəlcə Linux nüvəsini kompilyasiya edib virtual maşını yeni nüvə faylı ilə yükləyirik. Test məqsədi üçün istifadə olunan distributiv **ArchLinux**, virtual maşının adı **arch**, fiziki yaddaşın ölçüsü **512 Mb** -dır. Virtual maşın yeni nüvə ilə yükləndikdən sonra host maşından terminalı yükləyib virtual maşının fiziki yaddaşının (sistem) sürətini almaq üçün aşağıdakı əmri daxil edirik:

VBoxManage debugvm arch dumpguestcore --filename fiziki_yaddash_elf

Nəticədə virtual maşının fiziki yaddaşı, cpu -nun registrləri v.s. məlumatlar elf formatında olan **fiziki_yaddash_elf** adlı fayla yazılacaq. Daha sonra bu fayldan fiziki yaddaşı almaq üçün aşağıdakı kimi etməliyik.

1. Əvvəlcə **objdump** proqramı ilə elf faylının strukturunu örgənirik.

```
 objdump -h  fiziki_yaddash_elf

core1_64:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 note0         00000440  0000000000000000  0000000000000000  000002a8  2**0
                  CONTENTS, READONLY
  1 load1         20000000  0000000000000000  0000000000000000  000006e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
  2 load2         00600000  0000000000000000  00000000e0000000  200006e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
  3 load3         00000000  0000000000000000  00000000f0000000  206006e8  2**0
                  ALLOC, READONLY
```

```
  4 load4         00080000  0000000000000000  00000000f0080000  206006e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
  5 load5         00400000  0000000000000000  00000000f0400000  206806e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
  6 load6         00004000  0000000000000000  00000000f0800000  20a806e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
  7 load7         00000000  0000000000000000  00000000f0804000  20a846e8  2**0
                  ALLOC, READONLY
  8 load8         00000000  0000000000000000  00000000fec00000  20a846e8  2**0
                  ALLOC, READONLY
  9 load9         00000000  0000000000000000  00000000fee00000  20a846e8  2**0
                  ALLOC, READONLY
 10 load10        00010000  0000000000000000  00000000ffff0000  20a846e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY
```

Burada 2 -ci hissənin ölçüsü 0x20000000 bayt (512 MB), sürüşməsi isə 0x6e8 (1768) baytdır. Aşağıdakı əmr ilə elf formatlı fayldan fiziki yaddaş alırıq.

dd if=fiziki_yaddash_elf of=fiziki_yaddash bs=8 skip=221 count=67108864

Burada 8*221 = 1768, 8*67108864 = 512 MB.

Beləliklə bu əmri icra etdikdən sonra sistem yaddaşı elf faylından alınaraq f**iziki_yaddash** adlı fayla yazılacaq.

2 Kəsilmə cədvəlinin yaddaşdan fayla köçürülməsi.

Əvvəlcə kəsilmə cədvəlinin fiziki yaddaşdakı ünvanını örgənməliyik. Kəsilmə cədvəlinin ünvanı CPU -nun **idtr** reqistrində yerləşdiyindən aşağıdakı əmrlə həmin ünvanı örgənirik:

```
VBoxManage debugvm arch getregisters idtr

idtr = 0xffffffff81b91000:0fff
```

Kəsilmələr cədvəlinin virtual ünvanı 0xffffffff81b91000 -dır. Bu ünvanı fiziki ünvana çüvirmək üçün ondan 0xffffffff80000000 çıxmalıyıq. Nəticədə kəsilmə cədvəlinin fiziki ünvanını alarıq **0x1b91000**. 64 bitlik intel x86 arxitekturasında hər-bir kəsilmə üçün 16 bayt yer ayrılır, Cəmi 256 kəsilmə təyin olunur. Kəsilmələr cədvəlinin ölçüsü 256*16 = 4096 bayt (1 səhifə, page) olur. Həmin cədvəli fiziki yaddaşın dump faylından almaq üçün aşağıdakı əmri daxil edirik:

dd if= f**iziki_yaddash of=kesilme_cedveli count=4096 skip=28905472 bs=1**

Burada 0x1b91000 = 28905472.

Beləliklə kəsilmələr cədvəli **kesilme_cedveli** faylına yazılmış olacaq.

3 Kəsilmələr cədvəlinin dump faylından kəsilmə funksiyalarının alınması

Kəsilmələr cədvəlində hər-birinin ölçüsü 16 bayt olan cəmi 256 sayda kəsilmə barəsində məlumat yerləşir. Bu cədvəldə hər bir 16 baytlıq element bir kəsilməni təyin edir və özündə müvafiq nömrəli kəsilməyə aid müxtəlif məlumatlar saxlayır. Bu məlumatlardan bizə ən lazım olanı kəsilmənin yaddaşdakı ünvanıdır. 64 bitlik intel x86 arxitekturasında kəsilmənin ünvanına 8 bayt yer ayrılır aşağıdakı kimi:

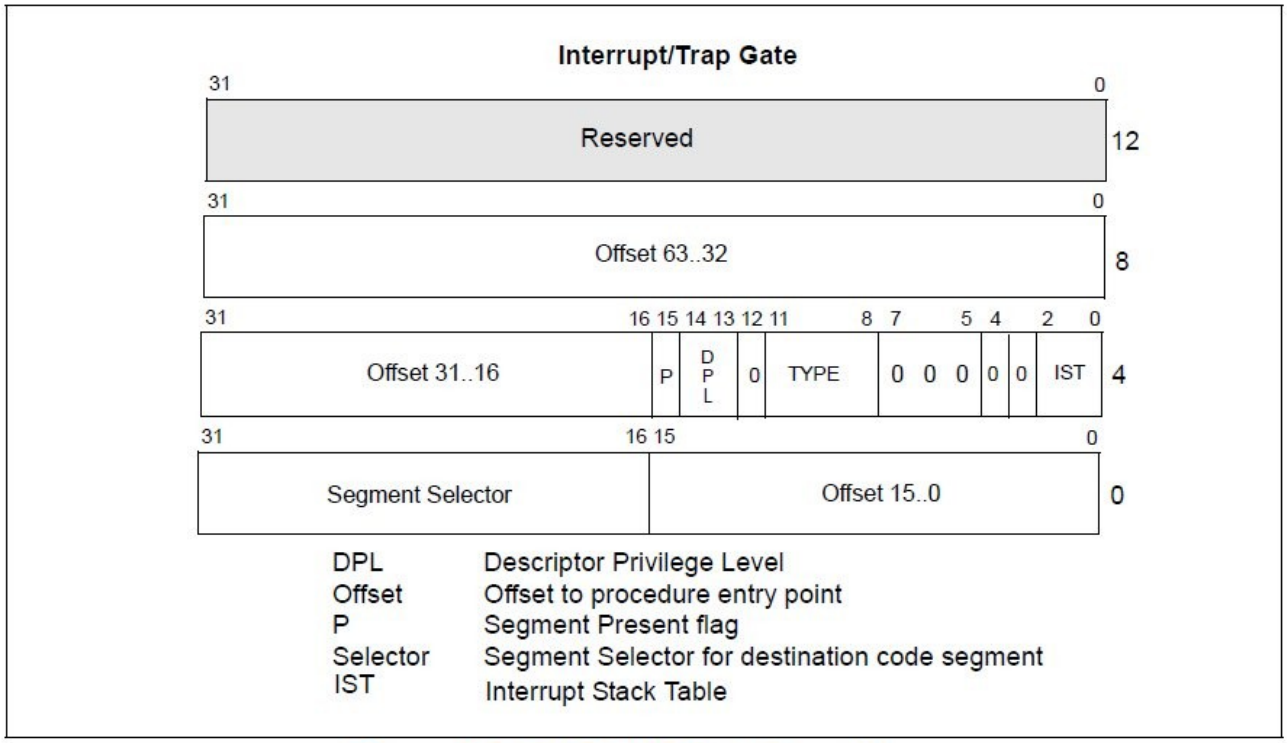

Figure 5-7. 64-Bit IDT Gate Descriptors

Mənbə :

Intel® 64 and IA-32 Architectures
Software Developer's Manual
Volume 3A:
System Programming Guide, Part 1

Aşağıdakı C proqramı vastəsilə kəsilmələr faylından kəsilmələr oxunur, onların ünvanı hesablanır, sətir formasına çevrilir və System.map faylından həmin ünvanlı funksiya axtarılaraq çap edilir.

```c
/*
 kesilme.c
 Ahmed Sadikhov ahmed.sadikhov@gmail.com
 License GPL V3.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

void son(char *s){
  printf("%s\n",s);
  exit(0);
};


struct gate_desc {
        unsigned short low;
        unsigned short seg;
        unsigned char  d1;
        unsigned char  d2;
        unsigned short mid;
        unsigned int   high;
        unsigned int   zero1;
} gate_desc;


struct unv{
      unsigned short low;
      unsigned short mid;
      unsigned int high;
     } unv;


char* get_idt_nm(char *unv, char *bufer1){

  int k;
  char *argv[3], bufer[100];

  int p1[2], p2[2];

  argv[0] = malloc(strlen("grep"));
  argv[1] = malloc(strlen(unv));
  argv[2] = malloc(strlen("System.map"));
  argv[3] = NULL;

  strcpy(argv[0],"/bin/grep");
  strcpy(argv[1],unv);
  strcpy(argv[2],"System.map");

  if (pipe(p1)==-1) son("kanal yaradila bilmedi");
  if (pipe(p2)==-1) son("kanal yaradila bilmedi");

   k=fork();
```

```
  if (k==0){  /*child*/
    close(p1[1]);
    close(p2[0]);
    close(1);
    dup(p2[1]);
    execve("/bin/grep",argv,NULL);
  }
  else {
 /* parent */
    close(p1[0]);
    close(p2[1]);
    memset(bufer,'\0',100);
    read(p2[0],bufer,100);
    strcpy(bufer1,bufer);
    /* printf(bufer1); */
    waitpid(k,NULL,0);
  }
}



int main( int argc, char *argv[]){

  int fd,k,i=0;

  char bufer[100],bufer1[100];

  unsigned long *ptr = malloc(100);

  if (argc<2) son("istifade ./kesilme kesilme_fayli");

  fd = open(argv[1], O_RDONLY);

  if (fd<0) son("file acilmir");

  while((k=read(fd,&gate_desc,16))!=0){
    unv.low  = gate_desc.low;
    unv.mid  = gate_desc.mid;
    unv.high = gate_desc.high;
    memcpy(ptr,&unv,sizeof(unv));
    memset(bufer,'\0',100);
    memset(bufer1,'\0',100);
    sprintf(bufer,"%lx",*ptr);
    get_idt_nm(bufer,bufer1);
    i++;
    if (strcmp(bufer1,"")==0) continue;
    printf("Idt func %d %s",i,bufer1);
    }
 }
```

Proqramı gcc kompilyatoru ilə kompilyasiya edək

gcc keslime.c -o kesilme

Kəsilmələri almaq üçün aşağıdakı əmri daxil edirik:

./kesilme kesilme_cedveli

Nəticədə kəsilmə funksiyalarının nömrələri, ünvanları və adları çap olunacaq:

```
Idt func 1 ffffffff81570780 T divide_error
Idt func 2 ffffffff815686e0 T debug
Idt func 3 ffffffff81568af0 T nmi
Idt func 4 ffffffff81568720 T int3
Idt func 5 ffffffff815707a0 T overflow
Idt func 6 ffffffff815707c0 T bounds
Idt func 7 ffffffff815707e0 T invalid_op
Idt func 8 ffffffff81570800 T device_not_available
Idt func 9 ffffffff81570820 T double_fault
Idt func 10 ffffffff81570850 T coprocessor_segment_overrun
Idt func 11 ffffffff81570870 T invalid_TSS
Idt func 12 ffffffff815708a0 T segment_not_present
Idt func 13 ffffffff81568760 T stack_segment
Idt func 14 ffffffff81568800 T general_protection
Idt func 15 ffffffff81568830 T page_fault
Idt func 16 ffffffff815708d0 T spurious_interrupt_bug
Idt func 17 ffffffff815708f0 T coprocessor_error
Idt func 18 ffffffff81570910 T alignment_check
Idt func 19 ffffffff81568890 T machine_check
Idt func 20 ffffffff81570940 T simd_coprocessor_error
Idt func 33 ffffffff81570280 T irq_entries_start
Idt func 129 ffffffff81571040 T ia32_syscall
Idt func 240 ffffffff81570680 T apic_timer_interrupt
Idt func 245 ffffffff81570700 T mce_self_interrupt
Idt func 247 ffffffff81570760 T irq_work_interrupt
Idt func 248 ffffffff815706a0 T x86_platform_ipi
Idt func 250 ffffffff815706c0 T threshold_interrupt
Idt func 251 ffffffff815706e0 T thermal_interrupt
Idt func 255 ffffffff81570720 T error_interrupt
Idt func 256 ffffffff81570740 T spurious_interrupt
```

Əhməd Sadıxov 20.12.2012
ahmed.sadikhov@gmail.com