



به نام خدا

آزمایشگاه سیستم عامل

# آشنایی، اجرا و اشکال زدایی هسته سیستم عامل xv6

(بخش سوم: اشکال زدایی)



## مقدمه

کد در حین اجرا ممکن است دارای اشکال باشد. یکی از ابزارهای تسریع اشکال زدایی، اشکال زداهای هستند. در همین راستا، اشکال زداهای متعددی برای سیستم عامل ها و معماری های مختلف ارائه شده است. ابزار اشکال زدای گنو<sup>۱</sup> (GDB)، یک اشکال زدای متداول در سیستم های یونیکسی بوده که در این بخش از آزمایش روش کار با آن به طور اجمالی بررسی خواهد شد. به عنوان مثال ممکن است یک کد بزرگ، هنگام اجرا دچار خرابی شود. استفاده از GDB می تواند دستور منجر به از کار افتادگی و زنجیره فراخوانی توابع منتهی به این دستور را نمایش دهد. جهت اشکال زدایی با GDB، در گام نخست باید سیستم عامل به صورتی بوت شود که قابلیت اتصال اشکال زدا به آن وجود داشته باشد. مراحل اتصال عبارت است از:

(۱) در یک ترمینال دستور `make qemu-gdb` اجرا گردد.

(۲) سپس در ترمینالی دیگر، فایل کد اجرایی به عنوان ورودی به GDB داده شود.

چنانچه در بخش دوم این آزمایش ذکر شد کد اجرایی شامل یک نیمه هسته و یک نیمه سطح کاربر بوده که نیمه هسته ثابت و نیمه سطح کاربر بسته به برنامه در حال اجرا بر روی پردازنده دائماً در حال تغییر است. به این ترتیب، به عنوان مثال، هنگام اجرای برنامه `cat`، کدهای اجرایی سیستم شامل کد هسته و کد برنامه `cat` خواهند بود. جهت اشکال زدایی بخش سطح کاربر، کافی است دستور `gdb _cat` و جهت اشکال زدایی بخش هسته دستور `gdb kernel` فراخوانی شود. دقت شود در هر دو حالت، هر دو کد سطح هسته و کاربر اجرا می شوند. اما اشکال زدا فقط روی یک کد اجرایی (سطح کاربر یا هسته) کنترل داشته و تنها قادر به انجام عملیات بر روی آن قسمت خواهد بود.

(۳) نهایتاً با وارد کردن دستور `target remote tcp::26000` در GDB، اتصال به سیستم عامل صورت خواهد گرفت.

با وارد کردن دستور `c` کد سیستم به اجرا ادامه خواهد داد. همچنین با فشردن دکمه `Ctrl + C` می توان به اشکال زدا بازگشت.

## اجرای اولیه اشکال زدا

برنامه `cat`، فایل ورودی را از دیسک خوانده و سپس محتوای آن را در ترمینال نمایش می دهد. خواندن از دیسک مستلزم انتقال از سطح کاربر به سطح هسته است. زیرا دسترسی به دستگاه های ورودی/خروجی، یک عملیات ممتاز است. بدین منظور یک تابع به نام `read()` در سطح کاربر وجود داشته که خود منجر به فراخوانی یک تابع در سطح هسته موسوم به `sys_read()` می گردد. چنانچه در پروژه بعدی آزمایشگاه خواهید دید، تابع دوم فراخوانی سیستمی نام داشته و یکی از فراخوانی های سیستمی پیاده سازی شده در هسته سیستم عامل است. در شکل زیر بخشی از برنامه `cat` که مربوط به خواندن از دیسک است نشان داده شده است.

<sup>1</sup> GNU Debugger

```

7 void
8 cat(int fd)
9 {
10  int n;
11
12  while((n = read(fd, buf, sizeof(buf))) > 0) {
13      if (write(1, buf, n) != n) {
14          printf(1, "cat: write error\n");
15          exit();
16      }
17  }
18  if(n < 0){
19      printf(1, "cat: read error\n");
20      exit();
21  }
22 }

```

چنانچه مشاهده می شود در خط ۱۲، فایل محتوای فایل خوانده شده و در ترمینال چاپ می گردد. فراخوانی سیستمی `sys_read()` نیز که در فایل `sysfile.c` پیاده سازی شده است در شکل زیر نشان داده شده است.

```

69 int
70 sys_read(void)
71 {
72     struct file *f;
73     int n;
74     char *p;
75
76     if(argfd(0, 0, &f) < 0 || argint(2, &n) < 0 || argptr(1, &p, n) < 0)
77         return -1;
78     return fileread(f, p, n);
79 }

```

یک سازوکار مهم در اشکال زدایی نرم افزار نقطه توقف<sup>۲</sup> است. این ابزار برای توقف اجرای برنامه در یک نقطه از پیش تعیین شده، طراحی شده است. این نقطه، آدرسی از کد برنامه می باشد. این آدرس را می توان بر اساس نام تابع، شماره خط در فایل کد منبع یا مقدار عددی آن به اشکال زدا اطلاع داد. به عنوان مثال اگر به دلایلی قصد اشکال زدایی در تابع `read()` یا بخش سطح هسته آن یعنی `sys_read()` وجود داشته باشد، می توان با استفاده از دستورهای مختلفی که در جدول زیر نشان داده شده است، عمل نمود.

فراخوانی سیستمی <code>sys_read()</code>	تابع سطح کاربر <code>read()</code>	
b (sysfile.c):sys_read	3b (cat.c):cat	نام تابع
b sysfile.c:71	b cat.c:12	مکان در کد منبع
b *0x80104b10	b *0x98	مکان در حافظه

(۱) یک Breakpoint روی فراخوانی سیستمی `sys_read()` قرار داده<sup>۴</sup> و دستور `bt` را اجرا نمایید. در مورد خروجی این دستور و رابطه اجزای آن توضیح دهید.

<sup>۲</sup> Breakpoint

<sup>۳</sup> دقت شود در این شیوه تعیین Breakpoint، تنها می توان ابتدای تابع دربرگیرنده نقطه اشکال زدایی را تعیین نمود. لذا Breakpoint روی خط ۹ (و نه خط ۱۲) قرار می گیرد. همچنین بخش داخل پرانتز، اختیاری است.

<sup>۴</sup> دقت شود با توجه به این که `sys_read()` در هسته است، باید نقطه فراخوانی آن در سطح کاربر را متوقف نمایید. این نقطه در فایل `usys.S` قرار دارد. دقت نمایید `read()` فراخوانی سیستمی نیست، بلکه یک تابع کتابخانه ای است که فراخوانی سیستمی را فراخوانی می کند.

Breakpoint را می‌توان پیش از اجرای برنامه یا در زمان توقف اجرا (به سبب توقف روی یک Breakpoint قبلی یا توقف ناشی از Ctrl + C) قرار داد.

۲) یک حلقه بی‌نهایت در فراخوانی سیستمی `sys_read()` ایجاد نموده و نقطه اشکال در سطح هسته و کاربر را برای برنامه `cat` توسط GDB نشان دهید.

پس از توقف روی Breakpoint می‌توان با اجرای دستورهای `s(tep)` و `fin(ish)` به ترتیب به دستور بعدی، به درون دستور بعدی (اگر فراخوانی تابع باشد) و به خارج از تابع کنونی (یعنی بازگشت به تابع فراخواننده) منتقل شد. به عبارت دیگر، اجرا گام‌به‌گام قابل بررسی است. بدین معنی که پیش از اجرای خط جاری برنامه سطح کاربر یا هسته، امکان دستیابی به اطلاعات متغیرها و ثبات‌ها فراهم می‌باشد. به این ترتیب می‌توان برنامه را از جهت وجود حالات نادرست، بررسی نمود.

۳) دو تفاوت دستورهای `x` و `print` را توضیح دهید. چگونه می‌توان محتوای یک ثبات خاص را چاپ کرد؟ (راهنمایی: می‌توانید از دستور `help` استفاده نمایید: `help x` و `help print`)

### آشنایی با قابلیت‌های سطح پایین‌تر

اشکال‌زدایی برنامه در سطوح مختلفی قابل انجام است. با توجه به این که ممکن است بخشی از کد سطح بالا به دنبال بهینه‌سازی‌های کامپایلری اجرا نشده یا ترتیب اجرای آن تغییر کند، نیاز به اشکال‌زدایی در سطح کد اسمبلی خواهد بود. زیرا کدی که در واقع اجرا می‌گردد، همین کد است. ضمن این که ممکن است برخی از فایل‌های کد منبع، خود به دلایلی از جمله بهینه‌سازی یا عدم پشتیبانی زبان برنامه‌نویسی به صورت کد اسمبلی نوشته شده باشند. جهت آشنایی با این قابلیت، ابتدا یک Breakpoint در خط ۳۶ برنامه `cat` قرار دهید. سپس با وارد کردن ورودی مناسب، این خط را اجرا نمایید. جهت سهولت در مشاهده روند اجرا دستور `layout src` را در اشکال‌زدا اجرا نمایید. مشاهده خواهید نمود که مطابق شکل زیر، خط ۳۶ برنامه، علامت‌گذاری شده است.

```

cat.c
28
29     if(argc <= 1){
30         cat(0);
31         exit();
32     }
33
34     for(i = 1; i < argc; i++){
35         if((fd = open(argv[i], 0)) < 0){
B> 36             printf(1, "cat: cannot open %s\n", argv[i]);
37             exit();
38         }
39         cat(fd);
40         close(fd);
41     }
42     exit();
43 }
44
45
46

remote Thread 1 In: main
(gdb)

```

در این حالت، اجرای دستور `layout asm` کد اسمبلی همین بخش از برنامه را نشان می‌دهد که در شکل زیر قابل مشاهده است.

```

B+> 0x69 <main+105> push    %eax
      0x6a <main+106> pushl   (%ebx)
      0x6c <main+108> push    $0x82b
      0x71 <main+113> push    $0x1
      0x73 <main+115> call    0x4b0 <printf>
      0x78 <main+120> call    0x362 <exit>
      0x7d <main+125> sub     $0xc,%esp
      0x80 <main+128> push    $0x0
      0x82 <main+130> call    0x90 <cat>
      0x87 <main+135> call    0x362 <exit>
      0x8c             xchg    %ax,%ax
      0x8e             xchg    %ax,%ax
      0x90 <cat>        push    %ebp
      0x91 <cat+1>      mov     %esp,%ebp
      0x93 <cat+3>      push    %esi
      0x94 <cat+4>      push    %ebx
      0x95 <cat+5>      mov     0x8(%ebp),%esi
      0x98 <cat+8>      jmp     0xb7 <cat+39>
      0x9a <cat+10>     lea     0x0(%esi),%esi
remote Thread 2 In: main
(gdb)

```

مشاهده می کنید چهار دستور `push`، پیش از فراخوانی `printf` (توسط دستور `call`) وجود دارد که وظیفه قرار دادن پارامترها و مقادیر برخی ثبات‌ها در پشته را بر عهده دارند. جهت انتقال به دستور بعدی در سطح کد اسمبلی نمی توان از دستور `s` یا `n` استفاده کرد. زیرا تمامی این خطوط کد اسمبلی، مربوط به یک خط از کد سی (خط ۳۶) هستند.

(۴) دستورهای معادل `s` و `n` در سطح کد اسمبلی را نام ببرید.

(۵) برنامه `cat` محتوای یک فایل را در ترمینال می نویسد. برای نوشتن در ورودی/خروجی نیاز به دسترسی به سرویس های هسته سیستم عامل است. لذا فراخوانی سیستمی `sys_write` رخ خواهد داد. یک `Breakpoint` روی آن قرار دهید. سپس دستور `bt` را وارد نمایید. خروجی آن چیست؟ توضیح دهید.

(۶) در مورد خروجی `layout src` و `layout asm` توضیح دهید. (راهنمایی: دستور نخست خروجی `layout asm`، شماره فراخوانی سیستمی `sys_write` را در ثبات `eax` قرار می دهد. در واقع طبق واسطه باینری برنامه های کاربردی<sup>۵</sup> (ABI)، این ثبات، باید بدین صورت مقداردهی شود. دو دستور بعدی چه نقشی دارند؟)

جهت انتقال به مد عادی اشکال زدایی ابتدا باید کلیدهای `Ctrl + x` و سپس کلید `C` فشار داد.

## اشکال زدایی بر اساس داده

تا اینجا اشکال زدایی بر اساس قرار دادن نقاط توقف روی آدرس های برنامه بررسی شد. یک روش متفاوت اشکال زدایی، اشکال زدایی بر اساس داده ها بوده که به کمک قطعه کد زیر در ابتدای فایل فرضی `foo.c` قابل بیان است.

```

int i;
for (i = 0; i < 1000000; i++)
    x = rand();

```

<sup>5</sup> Application Binary Interface

اگر فرض شود مقدار ۱- برای  $X$  یک مقدار نامعتبر است، چگونه می‌توان وقوع این حالت را به کمک اشکال‌زدا تشخیص داد؟ البته یک روش، تغییر در کد و افزودن شرطی جهت تشخیص این حالت است. اما تشخیص بدون تغییر در کد توسط دستور زیر میسر می‌شود.

`b foo.c:2 if (x = -1)`

اما اگر متغیر  $X$ ، یک متغیر سراسری بوده و در چندین فایل کد منبع مورد دسترسی قرار بگیرد، چه باید کرد؟ در این حالت تشخیص با تغییر در کد نیز ساده نخواهد بود. در شرایط پیچیده‌تر، ممکن است این دسترسی‌ها به طور غیرمستقیم و توسط اشاره‌گرها رخ دهد. در این حالت بهتر است به جای علامت‌گذاری خط کد (توسط Breakpoint) داده‌ها را علامت‌گذاری نمود. بدین‌منظور watchها طراحی شده‌اند که خود چندین نوع دارند. ساده‌ترین آن‌ها با استفاده از دستور زیر قابل تعریف است.

`watch *0x12345678`

در این‌جا نوشتن در این آدرس حافظه منجر به توقف اجرا در نقطه دسترسی می‌گردد. می‌توان watch را روی متغیرهای محلی هم گذاشت. البته باید به حوزه تعریف متغیرها نیز دقت نمود. دقت کنید موارد استفاده watch تنها به نکات مذکور در بالا ختم نمی‌شود. در کدهای همروند یا موازی اشکال‌زدایی بسیار دشوارتر از کدهای ترتیبی است. اگر دسترسی به متغیری به صورت همروند یا موازی صورت پذیرد و مثلاً متغیری مقدار نادرستی بگیرد، استفاده از watch جهت تشخیص حالت اشکال و عامل آن بسیار راهگشا خواهد بود. به‌خصوص که در این شرایط، تکرارپذیری آزمایش نیز به سادگی ممکن نبوده و بسیاری از اجراها منجر به خروجی صحیح می‌گردند.

## نکات مهم

- برای تحویل پروژه ابتدا یک مخزن خصوصی در سایت GitLab ایجاد نموده و سپس پروژه خود را در آن Push کنید.
- سپس اکانت UT\_OS\_TA را با دسترسی Maintainer به مخزن خود اضافه کنید. کافی است در محل بارگذاری در سایت درس، آدرس مخزن، شناسه آخرین Commit و گزارش پروژه را بارگذاری نمایید.
- همه اعضای گروه باید به پروژه‌ی آپلود شده توسط گروه خود مسلط باشند و لزوماً نمره افراد یک گروه با یکدیگر برابر نیست.
- در صورت مشاهده هرگونه مشابهت بین کدها یا گزارش دو گروه، نمره ۰ به هر دو گروه تعلق می‌گیرد.
- تمامی سؤالات را در کوتاه‌ترین اندازه ممکن در گزارش خود پاسخ دهید.