Московский авиационный институт
Национальный исследовательский университет

Операционные системы
Кафедра 806

Лабораторная работа №1

Тема: утилита диагностики strace

Студент: Минибаев Айдар
Группа: М8О-301Б-18
Преподаватель: Миронов Е.С.
Дата: .
Подпись: .

Москва 2020

1. Постановка задачи

Освоить утилиту диагностики strace и продемонстрировать её вывод на различных программах.

2. Описание

Вывод strace на максимально простой программе:

```c
int main() {
    return 0;
}
```

```
$ strace ./simple.out
execve("./simple.out", ["./simple.out"], 0x7fff892b5fe0 /* 51 vars */) = 0
brk(NULL)                               = 0x5596b08bb000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc5790b890) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=103640, ...}) = 0
mmap(NULL, 103640, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7faa5fde7000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7faa5fde5000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7faa5fbf3000
mprotect(0x7faa5fc18000, 1847296, PROT_NONE) = 0
mmap(0x7faa5fc18000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x25000) = 0x7faa5fc18000
mmap(0x7faa5fd90000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) =
0x7faa5fd90000
mmap(0x7faa5fddb000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e7000) = 0x7faa5fddb000
mmap(0x7faa5fde1000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7faa5fde1000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7faa5fde6540) = 0
mprotect(0x7faa5fddb000, 12288, PROT_READ) = 0
mprotect(0x5596aec29000, 4096, PROT_READ) = 0
mprotect(0x7faa5fe2e000, 4096, PROT_READ) = 0
munmap(0x7faa5fde7000, 103640)          = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Как видно, даже простейшая программа вызывает много системный вызывов.

Посмотрим, какие системные вызовы происходят при выделении памяти:

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#define LINE "\n\n==================================\n\n"

int main() {
    write(STDOUT_FILENO, LINE, sizeof(LINE) / sizeof(char));
    void *a = malloc(10);
    write(STDOUT_FILENO, LINE, sizeof(LINE) / sizeof(char));
    free(a);
    write(STDOUT_FILENO, LINE, sizeof(LINE) / sizeof(char));
    return 0;
}
```

```
$ strace ./malloc.out
execve("./malloc.out", ["./malloc.out"], 0x7ffdb2a2af40 /* 51 vars */) = 0
brk(NULL)                               = 0x561308683000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffef3573b40) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=103640, ...}) = 0
mmap(NULL, 103640, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe7d97e6000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe7d97e4000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe7d95f2000
mprotect(0x7fe7d9617000, 1847296, PROT_NONE) = 0
mmap(0x7fe7d9617000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x25000) = 0x7fe7d9617000
mmap(0x7fe7d978f000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) =
0x7fe7d978f000
mmap(0x7fe7d97da000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e7000) = 0x7fe7d97da000
mmap(0x7fe7d97e0000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fe7d97e0000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fe7d97e5540) = 0
mprotect(0x7fe7d97da000, 12288, PROT_READ) = 0
mprotect(0x561307385000, 4096, PROT_READ) = 0
mprotect(0x7fe7d982d000, 4096, PROT_READ) = 0
munmap(0x7fe7d97e6000, 103640)          = 0
```

```
write(1, "\n\n==============================="..., 38

================================

) = 38
brk(NULL)                               = 0x561308683000
brk(0x5613086a4000)                     = 0x5613086a4000
write(1, "\n\n==============================="..., 38

================================

) = 38
write(1, "\n\n==============================="..., 38

================================

) = 38
exit_group(0)                           = ?
+++ exited with 0 +++
```

Заметим, что free не делает каких-либо системных вызовов, так как он просто очищает одну из внутренних структур аллокатора glibc.

Рассмотрим программу со вводом числа:

```c
#include <stdio.h>
int main() {
    int a;
    scanf("%i", &a);
    return a;
}
```

```
$ strace ./input.out
execve("./input.out", ["./input.out"], 0x7fffe684e420 /* 51 vars */) = 0
brk(NULL)                               = 0x563e62ce3000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffda14f6350) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=103640, ...}) = 0
mmap(NULL, 103640, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0921e1e000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0921e1c000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) =
68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0921c2a000
```

```
mprotect(0x7f0921c4f000, 1847296, PROT_NONE) = 0
mmap(0x7f0921c4f000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x25000) = 0x7f0921c4f000
mmap(0x7f0921dc7000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) =
0x7f0921dc7000
mmap(0x7f0921e12000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e7000) = 0x7f0921e12000
mmap(0x7f0921e18000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f0921e18000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f0921e1d540) = 0
mprotect(0x7f0921e12000, 12288, PROT_READ) = 0
mprotect(0x563e61226000, 4096, PROT_READ) = 0
mprotect(0x7f0921e65000, 4096, PROT_READ) = 0
munmap(0x7f0921e1e000, 103640)          = 0
fstat(0, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x1), ...}) = 0
brk(NULL)                               = 0x563e62ce3000
brk(0x563e62d04000)                     = 0x563e62d04000
read(0, 42
"42\n", 1024)                   = 3
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Недопустимая операция смещения)
exit_group(42)                          = ?
+++ exited with 42 +++
```

scanf делает аналогичные вызовы, что и printf. Однако, в конце программы вызывается lseek, необходимый для деинициализации scanf.

3. Выводы:

Освоил работу с утилитой диагностики strace и научился использовать её для отладки системных вызовов в программах.