Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

Лабораторная работа № 1

Студент: Марочкин И.А.
Группа: М8О-206Б-19
Преподаватель: Соколов А.А.
Дата: 24.04.2021
Оценка:

Москва, 2020

Цель работы

Приобретение практических навыков:

- Диагностики работы программного обеспечения на примере 4 лабораторной работы.

Задание

Провести диагностику работы 4 лабораторной работы при помощи strace, объяснить результат работы strace.

Вариант:

В файле записаны команды вида: “число число число<endline>”. Дочерний процесс производит деление первого числа в команде на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. числа имеют тип float.

Вывод strace

```
execve("./laba", ["./laba"], 0x7ffd0f13de78 /* 47 vars */) = 0
brk(NULL)                               = 0x20db000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc6346b990) = -1 EINVAL (Invalid
argument)  // специфичное состояние треда
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=84981, ...}) = 0
mmap(NULL, 84981, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7faf83d53000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/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 \203\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", 32, 848)
= 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0J\177\263t\t\177\271'\373\223\323^\371\213\250
\222"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=3222048, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7faf83d51000
```

```
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
mmap(NULL, 1876640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7faf83b86000
mprotect(0x7faf83bac000, 1683456, PROT_NONE) = 0
mmap(0x7faf83bac000, 1372160, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7faf83bac000
mmap(0x7faf83cfb000, 307200, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x175000) = 0x7faf83cfb000
mmap(0x7faf83d47000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c0000) = 0x7faf83d47000
mmap(0x7faf83d4d000, 12960, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7faf83d4d000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7faf83b84000
arch_prctl(ARCH_SET_FS, 0x7faf83d52580) = 0
mprotect(0x7faf83d47000, 12288, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x7faf83d93000, 4096, PROT_READ) = 0
munmap(0x7faf83d53000, 84981)           = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x3), ...}) = 0
brk(NULL)                               = 0x20db000
brk(0x20fc000)                          = 0x20fc000
brk(NULL)                               = 0x20fc000
fstat(0, {st_mode=S_IFREG|0644, st_size=9, ...}) = 0
read(0, "test1.txt", 4096)              = 9
read(0, "", 4096)                       = 0
openat(AT_FDCWD, "test1.txt", O_RDONLY) = 3
getpid()                                = 26758
clock_gettime(CLOCK_MONOTONIC, {tv_sec=20369, tv_nsec=30061271}) = 0
openat(AT_FDCWD, "/tmp/OS_laba4_tmpFile.tyOfYu", O_RDWR|O_CREAT|O_EXCL, 0600)
= 4
unlink("/tmp/OS_laba4_tmpFile.tyOfYu")  = 0
write(4, "\0\0\0\0\0\0\0\0\0\0", 10)     = 10
mmap(NULL, 10, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7faf83d67000
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7faf83d52850) = 26759
strace: Process 26759 attached
[pid 26758] wait4(-1,  <unfinished ...>
[pid 26759] dup2(3, 0)                  = 0
[pid 26759] write(1, "Enter file name: Count of number"..., 38Enter file
name: Count of numbers = 0
) = 38
[pid 26759] close(4)                    = 0
[pid 26759] close(3)                    = 0
[pid 26759] exit_group(0)               = ?
[pid 26759] +++ exited with 0 +++
<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 26759
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=26759, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
```

```
write(1, "Enter file name: Result from chi"..., 46Enter file name: Result
from child = 0.000000
) = 46
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Объяснение вывода strace

execve("./laba", ["./laba"], 0x7ffd0f13de78 /* 47 vars */) = 0
Исполняет программу ./laba с ключами ", ["./laba"]. Возвращает 0 – успешное выполнение.

brk(NULL) = 0x20db000
Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти = 0x20db000.

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
Проверяет /etc/ld.so.preload на существование и на наличие прав на чтение (R_OK), возвращает -1 – или не существует /etc/ld.so.preload или нет прав на чтение, errno устанавливается в ENOENT (компонент пути не существует или является "висячей" символической ссылкой).

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
Открывает /etc/ld.so.cache относительно дескриптора указанного каталога - AT_FDCWD (относительно текущего рабочего каталога вызывающего процесса) с правами доступа - O_RDONLY|O_CLOEXEC (на чтение и устанавливает флаг close-on-exec на новом файловом дескрипторе). Возвращает новый файловый дескриптор 3.

fstat(3, {st_mode=S_IFREG|0644, st_size=88176, ...}) = 0
Заполняет структуру указанную вторым аргументом fstat информацией об файле с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

mmap(NULL, 84981, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7faf83d53000
Создает отображение файла с файловым дескриптором 3 в память, начиная с адресса NULL (система сама выбирает), размер = 88176 байт, с правами

защиты памяти на чтение PROT_READ, задает тип отражаемого объекта MAP_PRIVATE - создает неразделяемое отражение с механизмом copy-on-write, запись в эту область памяти не влияет на файл, не определено, являются или нет изменения в файле после вызова mmap видимыми в отраженном диапазоне. Возвращает указатель на начало отраженной памяти = 0x7faf83d53000.

close(3) = 0
Закрывает файл с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \203\2\0\0\0\0\0"..., 832) = 832
Читает 832 байта данных из файла с файловым дескриптором 3 в буффер указанный вторым аргументом. Возвращает число успешно считанных байт = 832.

mprotect(0x7faf83bac000, 1683456, PROT_NONE) = 0
Контролирует доступ к области памяти начинающейся с адресса 0x7f4ece29c000 длины 2093056 байт, доступ к памяти запрещен - PROT_NONE. Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV. Возвращает 0 – успешное завершение.

arch_prctl(ARCH_SET_FS, 0x7faf83d52580) = 0
Устанавливает специфичное для архитектуры состояние. Устанавливает 64 битную базу для регистра FS (ARCH_SET_FS) в значение 0x7faf83d52580. Возвращает 0 – успешное выполнение.

munmap(0x7faf83d53000, 84981) = 0
Снимает отражение из заданной области памяти, 0x7faf83d53000 указатель на начало памяти, длина = 84981 байт. Возвращает 0 – успешное выполнение.

write(4, "\0\0\0\0\0\0\0\0\0\0", 10) = 10

Записывает 10 байт из буфера (второй аргумент) в файл с файловым дескриптором 4. Возвращает число успешно записанных байт = 10.

clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7faf83d52850) = 26759
Создает процесс-потомок с фалагами - CLONE_CHILD_CLEARTID (очищает id), CLONE_CHILD_SETTID (устанавливает id), SIGCHLD (сигнал о изменении статуса дочернего процесса), задает положение стека для процесса-потомка = NULL, задает указатель на id = 0x7faf83d52850. Возвращает pid процесса-потомка.

[pid 26758] wait4(-1, <unfinished ...>
В процессе с pid = 26758 ждет завершение работы процесса.

Так как в strace указан флаг –f, информация о процессах выводится подробно, рядом с системным вызовом указывается pid процесс, где происходит системный вызов.

Вывод

При помощи strace можно удобно анализировать работу программы, смотреть на различные системные вызовы их параметры, что каждый вызов возвращает. Также можно следить за системными вызовами по процессам. Все это помогает быстро искать неполадки в работе программы и устранять их.