МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт (Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика" Кафедра 806 "Вычислительная математика и программирование"

Лабораторные работы №8 По курсу «Операционные системы»

Студент: Попов А. Д.
Группа: М8О-208Б-23
Преподаватель: Живалев Е. А.
Дата:
Оценка:
Полимен

Описание и определение strace:

strace — это эффективный инструмент для диагностики и отладки, используемый в Unix-подобных операционных системах, таких как Linux. Он позволяет отслеживать и фиксировать системные вызовы, выполняемые процессом, а также сигналы, которые он обрабатывает. Системные вызовы представляют собой интерфейс взаимодействия между пользовательскими приложениями и ядром ОС. С помощью strace можно детально изучить, какие файлы открывает программа, какие сетевые соединения использует, какие ошибки возникают в процессе выполнения и многое другое. Этот инструмент особенно полезен для отладки, анализа производительности и изучения поведения процессов в реальном времени.

Репозиторий: https://github.com/aldpopov/OS_labs/tree/master/LW8

Листинг работы strace:

```
execve("./build/LW3/lab3", ["./build/LW3/lab3"], 0x7ffffa5e5c60 /* 48 vars
*/) = 0
brk (NULL)
                                     = 0x5dfa34fbf000
mmap (NULL, 8192, PROT READ| PROT WRITE, MAP PRIVATE | MAP ANONYMOUS, -1, 0)
0x729412a4b000
access("/etc/ld.so.preload", R OK)
                                         = -1 ENOENT (No such file or
openat(AT FDCWD, "/etc/ld.so.cache", O RDONLY|O CLOEXEC) = 3
fstat(3, {st mode=S IFREG|0644, st size=59691, ...}) = 0
mmap(NULL, 59691, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x729412a3c000
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O CLOEXEC) = 3
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832)
pread64(3,
fstat(3, {st mode=S IFREG|0755, st size=2125328, ...}) = 0
pread64(3,
= 784
           2170256, PROT READ,
                                 MAP PRIVATE | MAP DENYWRITE,
                                                              3, 0)
mmap(NULL,
0x729412800000
mmap(0x729412828000,
                                1605632,
                                                      PROT READ | PROT EXEC,
MAP PRIVATE | MAP FIXED | MAP DENYWRITE, 3, 0x28000) = 0x729412828000
mmap(0x7294129b0000, 323584, PROT READ, MAP PRIVATE | MAP FIXED | MAP DENYWRITE,
3, 0x1b0000) = 0x7294129b0000
mmap(0x7294129ff000,
                                 24576,
                                                     PROT READ | PROT WRITE,
MAP PRIVATE | MAP FIXED | MAP DENYWRITE, 3, 0x1fe000) = 0x7294129ff000
mmap(0x729412a05000)
                                 52624,
                                                     PROT READ | PROT WRITE,
MAP PRIVATE | MAP FIXED | MAP ANONYMOUS, -1, 0) = 0 \times 729412a05000
close(3)
                                     = 0
mmap (NULL, 12288, PROT READ | PROT WRITE, MAP PRIVATE | MAP ANONYMOUS, -1, 0) =
0x729412a39000
arch\ prctl(ARCH\ SET\ FS,\ 0x729412a39740) = 0
set tid address(0x729412a39a10)
set_robust_list(0x729412a39a20, 24)
rseq(0x729412a3a060, 0x20, 0, 0x53053053) = 0
mprotect(0x7294129ff000, 16384, PROT_READ) = 0
```

```
mprotect(0x5dfa34281000, 4096, PROT READ) = 0
mprotect(0x729412a83000, 8192, PROT READ) = 0
prlimit64(0,
                    RLIMIT STACK,
                                          NULL,
                                                    {rlim cur=8192*1024,
rlim max=RLIM64 INFINITY}) = 0
munmap(0x729412a3c000, 59691)
                                      = 0
getrandom("\x39\x56\xf4\x4c\x41\x85\xee\x97", 8, GRND NONBLOCK) = 8
brk (NULL)
                                      = 0x5dfa34fbf000
brk(0x5dfa34fe0000)
                                      = 0x5dfa34fe0000
fstat(0, \{st \ mode=S \ IFCHR | 0620, \ st \ rdev=makedev(0x88, \ 0), \ \ldots\}) = 0
read(0, "a\n", 1024)
unlink("/dev/shm/sem.semaphore write") = 0
unlink("/dev/shm/sem.semaphore read")
                                      = 0
openat(AT FDCWD, "/dev/shm/sem.semaphore write", O RDWR|O NOFOLLOW|O CLOEXEC)
= -1 ENOENT (No such file or directory)
getrandom("\x3a\x59\xc4\xd3\xcb\xb4\x1d\xc7", 8, GRND NONBLOCK) = 8
newfstatat(AT FDCWD,
                             "/dev/shm/sem.wWTIG7",
                                                             0x7fff4f4c1640,
AT SYMLINK NOFOLLOW) = -1 ENOENT (No such file or directory)
openat (AT FDCWD,
                                                      "/dev/shm/sem.wWTIG7",
O RDWR O CREAT O EXCLO NOFOLLOW O CLOEXEC, 0666) = 3
write(3,
mmap(NULL, 32, PROT READ|PROT WRITE, MAP SHARED, 3, 0) = 0x729412a4a000
link("/dev/shm/sem.wWTIG7", "/dev/shm/sem.semaphore write") = 0
fstat(3, {st mode=S IFREG|0664, st size=32, ...}) = 0
unlink("/dev/shm/sem.wWTIG7")
                                      = 0
close(3)
openat(AT FDCWD, "/dev/shm/sem.semaphore read", O RDWR|O NOFOLLOW|O CLOEXEC)
= -1 ENOENT (No such file or directory)
getrandom("\xf9\xa8\x98\x25\x1c\x68\x8f\x59", 8, GRND NONBLOCK) = 8
newfstatat(AT FDCWD,
                           "/dev/shm/sem.7rScGO",
                                                             0x7fff4f4c1640,
AT SYMLINK NOFOLLOW) = -1 ENOENT (No such file or directory)
openat (AT FDCWD,
                                                      "/dev/shm/sem.7rScGO",
O RDWR|O CREAT|O EXCL|O NOFOLLOW|O CLOEXEC, 0666) = 3
write (3.
mmap(NULL, 32, PROT READ|PROT WRITE, MAP SHARED, 3, 0) = 0x729412a49000
link("/dev/shm/sem.7rScGO", "/dev/shm/sem.semaphore read") = 0
fstat(3, {st mode=S_IFREG|0664, st_size=32, ...}) = 0
unlink("/dev/shm/sem.7rScGO")
                                      = 0
close(3)
                                       = 0
openat (AT FDCWD,
                                                  "/dev/shm/virtual memory",
O RDWR | O CREAT | O NOFOLLOW | O CLOEXEC, 0666) = 3
ftruncate(3, 10240)
                                      = 0
openat(AT FDCWD, "a", O RDONLY)
                                      = 4
clone(child stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child\ tidptr=0x729412a39a10)\ =\ 6481
mmap(NULL, 10240, PROT READ|PROT WRITE, MAP SHARED, 3, 0) = 0x729412a46000
futex(0x729412a49000, FUTEX WAIT BITSET|FUTEX CLOCK REALTIME, 0, NULL,
FUTEX BITSET MATCH ANY) = 0
fstat(1, \{st \ mode=S \ IFCHR | 0620, \ st \ rdev=makedev(0x88, \ 0), \ \ldots\}) = 0
write(1, "26.80\n", 6)
futex(0x729412a4a000, FUTEX WAKE, 1)
futex(0x729412a49000, FUTEX WAIT BITSET|FUTEX CLOCK REALTIME, 0,
                                                                      NULL.
FUTEX BITSET MATCH ANY) = 0
write(1, "0.00\n", 5)
                                      = 5
futex(0x729412a4a000, FUTEX WAKE, 1)
                                      = 1
write(1, "7.51\n", 5)
                                      = 5
futex(0x729412a4a000, FUTEX WAKE, 1)
munmap(0x729412a46000, 1024)
                                      = 0
wait4(6481, [\{WIFEXITED(s) \&\& WEXITSTATUS(s) == 0\}], 0, NULL) = 6481
```

```
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=6481, si_uid=1000, si_status=0, si_utime=0, si_stime=0} --- munmap(0x729412a49000, 32) = 0 munmap(0x729412a4a000, 32) = 0 unlink("/dev/shm/sem.semaphore_write") = 0 unlink("/dev/shm/sem.semaphore_read") = 0 close(3) = 0 unlink("/dev/shm/virtual_memory") = 0 close(4) = 0 exit_group(0) = ? +++ exited with 0 +++
```

Комментарии к листингу:

Если обобщить вывод, то он разделён на 3 последовательных этапа.

Сначала с помощью различных системных вызовов инициализируется программа, загружаются нужные библиотеки, устанавливаются соответствующие значения.

Затем программа начинает выполнение. Происходит вызов кода из программы — считывается название некоторого файла. Затем идёт создание семафоров, открытие файла, создание дочернего процесса и вывод значений

Последним этапом программа завершает работу: родительский процесс ждёт завершения дочернего, удаляются все семафоры, разделяемая память, освобождается отображенная память и программа выходит.

Вывод:

В рамках данной лабораторной работы инструмент strace был применен для детального анализа выполнения программы. Были изучены все системные вызовы, их параметры и возвращаемые значения. В процессе работы удалось глубже понять логику программы, а также убедиться в отсутствии критических ошибок, что свидетельствует о её стабильной и корректной работе.