Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

# Приобретение практических навыков диагностики работы программного обеспечения.

Студент: Попов Илья Павлович

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: 11.12.2021

Оценка: 5

Подпись: _____

Москва, 2021

# Постановка задачи

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются.

Используемые утилиты: strace.

# Листинг программы

Strace — это утилита Linux, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессом и операционной системой. Использование данной утилиты позволяет понять, что процесс пытается сделать в данное время. Strace может быть полезен при отладке программ.

Для удобства работы с протоколом утилиты можно использовать следующие ключи:

- -o file – Перенаправить протокол утилиты в файл file
- -e trace=filters – Указать выражения, по которым будут фильтроваться системные вызовы. Например -e trace=write,%process задаёт фильтрацию по системным вызовам write и по группе системных вызовов, связанных с межпроцессорным взаимодействием.
- -f – Отслеживать системные вызовы в дочерних процессах
- -y – Заменить в протоколе все файловые дескрипторы на имена соответствующих им файлов (где возможно).
- -p file – Отслеживать только обращения к файлу file
- -k – Отображать стек вызовов

# Примеры работы

**Лабораторная работа 3 – взаимодействие между потоками**

1. execve("./a.out", ["./a.out"], 0x7fff8d1140b0 /* 58 vars */) = 0
2. brk(NULL) = 0x55891d890000
3. arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcb9cda400) = -1 EINVAL (Invalid argument)
4. access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
5. openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6. fstat(3, {st_mode=S_IFREG|0644, st_size=71280, ...}) = 0
7. mmap(NULL, 71280, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f117df0d000
8. close(3) = 0
9. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
10. read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832) = 832
11. pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68
12. fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
13. mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f117df0b000

14. pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68
15. mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f117dee8000
16. mmap(0x7f117deef000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f117deef000
17. mmap(0x7f117df00000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f117df00000
18. mmap(0x7f117df05000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f117df05000
19. mmap(0x7f117df07000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f117df07000
20. close(3) = 0
21. openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
22. 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
23. 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\0\0"..., 784, 64) = 784
24. 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
25. pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
26. fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
27. 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\0\0"..., 784, 64) = 784
28. 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
29. pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68
30. mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f117dcf6000
31. mprotect(0x7f117dd1b000, 1847296, PROT_NONE) = 0
32. mmap(0x7f117dd1b000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f117dd1b000
33. mmap(0x7f117de93000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f117de93000
34. mmap(0x7f117dede000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f117dede000
35. mmap(0x7f117dee4000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f117dee4000
36. close(3) = 0
37. mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f117dcf3000
38. arch_prctl(ARCH_SET_FS, 0x7f117dcf3740) = 0
39. mprotect(0x7f117dede000, 12288, PROT_READ) = 0
40. mprotect(0x7f117df05000, 4096, PROT_READ) = 0
41. mprotect(0x55891d137000, 4096, PROT_READ) = 0
42. mprotect(0x7f117df4c000, 4096, PROT_READ) = 0
43. munmap(0x7f117df0d000, 71280) = 0
44. set_tid_address(0x7f117dcf3a10) = 7914

```
45. set_robust_list(0x7f117dcf3a20, 24)     = 0
46. rt_sigaction(SIGRTMIN, {sa_handler=0x7f117deefbf0, sa_mask=[],
    sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f117defd3c0}, NULL, 8) = 0
47. rt_sigaction(SIGRT_1, {sa_handler=0x7f117deefc90, sa_mask=[],
    sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f117defd3c0}, NULL, 8) = 0
48. rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
49. prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
50. fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
51. brk(NULL)                           = 0x55891d890000
52. brk(0x55891d8b1000)                 = 0x55891d8b1000
53. write(1, "\320\227\320\260\320\264\320\260\320\271\321\202\320\265
    \320\276\320\263\321\200\320\260\320\275\320\270\321\207\320\265\320"..., 71) = 71
54. write(1, "Usage: ./a.out
    <\320\276\320\263\321\200\320\260\320\275\320\270\321\207\320\265"..., 72) = 72
55. write(1, "\n", 1)                   = 1
56. exit_group(-1)                      = ?
57. +++ exited with 255 +++
```

# Вывод

В процессе выполнения данной лабораторной работы я познакомился с утилитой отслеживания системных вызовов strace Linux.

Используя ее, можно понять, к каким файлам обращается программа, какие сетевые порты она использует, какие ресурсы ей нужны, а также какие ошибки возвращает ей система. Это помогает разобраться в особенностях работы программы и лучше понять причину ошибки.

Несмотря на то, что эта утилита редко используется при дебаге лабораторных работ, я уверен, что опыт работы с ней пригодится мне в дальнейшей работе с более объемными проектами.