Московский авиационный институт
(Национальный исследовательский университет)

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

Лабораторная работа № 1

Тема: Операционные системы

Студент: Семин Александр Витальевич

Группа: М8О-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

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

Используемые утилиты: strace.

2. Пример работы

```
1. execve("./a.out", ["./a.out"], 0x7ffe0bc34370 /* 60 vars */) = 0
2. brk(NULL)                                = 0x55c8469e7000
3. arch_prctl(0x3001 /* ARCH_??? */, 0x7fff0308dec0) = -1 EINVAL
   (Недопустимый аргумент)
4. access("/etc/ld.so.preload", R_OK)       = -1 ENOENT (Нет такого
   файла или каталога)
5. openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6. fstat(3, {st_mode=S_IFREG|0644, st_size=79688, ...}) = 0
7. mmap(NULL, 79688, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f85f6a96000
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\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]\2
   23\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) = 0x7f85f6a94000
14.  pread64(3,
   "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\2
   23\337"..., 68, 824) = 68
15.  mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
   = 0x7f85f6a71000
16.  mmap(0x7f85f6a78000,         69632,         PROT_READ|PROT_EXEC,
   MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f85f6a78000
17.  mmap(0x7f85f6a89000,              20480,              PROT_READ,
   MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f85f6a89000
18.  mmap(0x7f85f6a8e000,         8192,         PROT_READ|PROT_WRITE,
   MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f85f6a8e000
19.  mmap(0x7f85f6a90000,        13432,         PROT_READ|PROT_WRITE,
   MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f85f6a90000
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\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"..
., 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"..
., 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)
= 0x7f85f687f000
31. mprotect(0x7f85f68a4000, 1847296, PROT_NONE) = 0
32. mmap(0x7f85f68a4000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f85f68a4000
33. mmap(0x7f85f6a1c000, 303104, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f85f6a1c000
34. mmap(0x7f85f6a67000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f85f6a67000
35. mmap(0x7f85f6a6d000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f85f6a6d000
36. close(3) = 0
37. mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f85f687c000
38. arch_prctl(ARCH_SET_FS, 0x7f85f687c740) = 0
39. mprotect(0x7f85f6a67000, 12288, PROT_READ) = 0
40. mprotect(0x7f85f6a8e000, 4096, PROT_READ) = 0
41. mprotect(0x55c845c63000, 4096, PROT_READ) = 0
42. mprotect(0x7f85f6ad7000, 4096, PROT_READ) = 0
43. munmap(0x7f85f6a96000, 79688) = 0
44. set_tid_address(0x7f85f687ca10) = 56962
45. set_robust_list(0x7f85f687ca20, 24) = 0
46. rt_sigaction(SIGRTMIN, {sa_handler=0x7f85f6a78bf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f85f6a863c0},
NULL, 8) = 0
47. rt_sigaction(SIGRT_1, {sa_handler=0x7f85f6a78c90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO,
sa_restorer=0x7f85f6a863c0}, 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_IFREG|0664, st_size=0, ...}) = 0
51.  brk(NULL)                                = 0x55c8469e7000
52.  brk(0x55c846a08000)                      = 0x55c846a08000
53.  write(1,                                      "\320\235\320\265
   \321\203\320\272\320\260\320\267\320\260\320\275
   \320\272\320\273\321\216\321\207\n", 27) = 27
54.  exit_group(-1)                           = ?
55.  +++ exited with 255 +++
```

3. 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

4. Выводы

В процессе выполнения данной 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.