Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №8 по курсу

«Операционные системы»

Группа: М80-206Б-22

Студент: Волков А.Д.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 29.12.2023

Москва, 2023

Постановка задачи

При выполнении лабораторных работ по курсу ОС необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР. По итогам выполнения всех лабораторных работ отчет по данной ЛР должен содержать краткую сводку по исследованию написанных программ.

Общий метод и алгоритм решения

strace - это утилита, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессом и операционной системой (ядром). Эти вызовы могут быть перехвачены и прочитаны. Это позволяет лучше понять, что процесс пытается сделать в заданное время. Перехватывая эти вызовы, мы можем добиться лучшего понимания поведения процессов, особенно если что-то идет не так.

У этой утилиты есть полезные опции, одна из таких -f, которая позволяет отслеживать, от какого процесса идет системный вызов. Еще есть опция -d, которая показывает некоторые отладочные данные самого strace для стандартной ошибки.

Strace:

JIP 1

```
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
pipe2([7, 8], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 8438 attached
, child_tidptr=0x7f870e207a10) = 8438
[pid  8437] clone(child_stack=NULL,
 flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
strace: Process 8439 attached
```

JIP 2

```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f02c2dff910
, parent_tid=0x7f02c2dff910, exit_signal=0, stack=0x7f02c25ff000, stack_size=0x7fff00,
tls=0x7f02c2dff640}strace: Process 55278 attached
 => {parent_tid=[55278]}, 88) = 55278
[pid 55277] <... clone3 resumed> => {parent_tid=[55279]}, 88) = 55279
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f02c25fe910
, parent_tid=0x7f02c25fe910, exit_signal=0, stack=0x7f02c1dfe000, stack_size=0x7fff00,
tls=0x7f02c25fe640}strace: Process 55280 attached
 => {parent_tid=[55280]}, 88) = 55280
[pid 55277] <... clone3 resumed> => {parent_tid=[55281]}, 88) = 55281
```

JIP 3

```
openat(AT_FDCWD, "file", O_RDWR|O_CREAT, 0666) = 3
openat(AT_FDCWD, "count", O_RDWR|O_CREAT, 0666) = 4
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fdff32f7000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fdff32bd000
[pid 60534] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd86ed000
[pid 60534] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
[pid 60534] <... mmap resumed>)         = 0x7fbdd86b3000
[pid 60534] munmap(0x7fbdd86b3000, 4096 <unfinished ...>
[pid 60534] <... munmap resumed>)       = 0
[pid 60534] munmap(0x7fbdd86ed000, 4096 <unfinished ...>
[pid 60534] <... munmap resumed>)       = 0
[pid 60535] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f2340fb4000
[pid 60535] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f2340f7a000
[pid 60535] munmap(0x7f2340f7a000, 4096) = 0
[pid 60535] munmap(0x7f2340fb4000, 4096) = 0
munmap(0x7fdff32bd000, 4096)            = 0
munmap(0x7fdff32f7000, 4096)            = 0
```

JIP 4

```
openat(AT_FDCWD, "./librealization1.so", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "./librealization2.so", O_RDONLY|O_CLOEXEC) = 3
```

JIP 5-7

```
[pid 9946] socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE <unfinished ...>
```

```
     [pid 9946] bind(9, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12
<unfinished ...>
      [pid 9946] socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP <unfinished ...>
      [pid 9946] setsockopt(9, SOL_SOCKET, SO_REUSEADDR, [1], 4 <unfinished ...>
[pid 9946] bind(9, {sa_family=AF_INET, sin_port=htons(4041),
sin_addr=inet_addr("127.0.0.1")}, 16 <unfinished ...>
[pid 9954] socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP <unfinished ...>
[pid 9954] getsockopt(10, SOL_SOCKET, SO_ERROR, <unfinished ...>
     [pid 9952] sendto(10, "\4&\5READY\vSocket-Type\0\0\0\3REQ\10Iden"..., 40, 0, NULL, 0
<unfinished ...>
     [pid 9954] sendto(10, "\4\31\5READY\vSocket-Type\0\0\0\3REP", 27, 0, NULL, 0
<unfinished ...>
     [pid 9950] socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE) = 11 [pid 9950]
bind(11, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 0 [pid 9950]
getsockname(11, {sa_family=AF_NETLINK, nl_pid=9950, nl_groups=00000000}, [12]) = 0
     [pid 9950] sendto(11, [{nlmsg_len=20, nlmsg_type=RTM_GETLINK,
nlmsg_flags=NLM_F_REQUEST|NLM_F_DUMP, nlmsg_seq=1703809046, nlmsg_pid=0},
{ifi_family=AF_UNSPEC, ...}], 20, 0, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
12) = 20
     [pid 9950] socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP) = 11
     [pid 9950] setsockopt(11, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
[pid 9957] socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP <unfinished ...>
[pid 9955] poll([{fd=8, events=POLLIN}], 1, -1 <unfinished ...>
     [pid 9957] getsockopt(10, SOL_SOCKET, SO_ERROR, [0], [4]) = 0
     [pid 9957] setsockopt(10, SOL_TCP, TCP_NODELAY, [1], 4) = 0
     [pid 9954] sendto(12, "\4&\5READY\vSocket-Type\0\0\0\3REQ\10Iden"..., 40, 0, NULL, 0
<unfinished ...>
[pid 9957] sendto(10, "\4\31\5READY\vSocket-Type\0\0\0\3REP", 27, 0, NULL, 0 <unfinished ...>
     [pid 9957] poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>
     [pid 9955] poll([{fd=8, events=POLLIN}], 1, 0 <unfinished ...>
[pid 9954] sendto(12, "\1\0\0\0052 pid", 9, 0, NULL, 0 <unfinished ...>
```

Вывод

В ходе выполнения лабораторных работ, я понял, что strace очень полезная утилита, тк может очень сильно помочь, в случае, когда в программе преобладают системные вызовы и необходимо отследить каждый из них. В принципе на протяжения всего семестра мы этим и занимались, писали программы, смотрели, что, какие системные вызовы делали, а самые значимые объясняли на защитах.