Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

ДИАГНОСТИКА ПРОГРАММЫ

Студент: Харьков Павел Александрович
Группа: М8О–206Б–19
Вариант: 20
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021.

Постановка задачи

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

Задание

Необходимо продемонстрировать ключевые системные вызовы, которые используются в лабораторной работе №2. Для этого я буду использовать утилиту strace.

Вариант 20: Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 инвертируют строки. Процессы пишут результаты своей работы в стандартный вывод.

Вывод strace

```
pablo@Tolmhto: ~/OS/os_lab_2$ strace ./laba_2 < tests/test1
execve("./laba_2", ["./laba_2"], 0x7ffc9b12e670 /* 27 vars */) = 0
brk(NULL)                               = 0x55ef3b4ea000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc4c71f690) = -1 EINVAL (Invalid argument)
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=41434, ...}) = 0
mmap(NULL, 41434, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f218bd64000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\360q\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\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68,
880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f218bd62000
```

```
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\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f218bb70000
mprotect(0x7f218bb95000, 1847296, PROT_NONE) = 0
mmap(0x7f218bb95000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f218bb95000
mmap(0x7f218bd0d000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f218bd0d000
mmap(0x7f218bd58000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f218bd58000
mmap(0x7f218bd5e000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f218bd5e000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f218bd63540) = 0
mprotect(0x7f218bd58000, 12288, PROT_READ) = 0
mprotect(0x55ef3b1c2000, 4096, PROT_READ) = 0
mprotect(0x7f218bd9c000, 4096, PROT_READ) = 0
munmap(0x7f218bd64000, 41434)           = 0
pipe([3, 4])                            = 0
pipe([5, 6])                            = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f218bd63810) = 232
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f218bd63810) = 233
close(3)                                = 0
close(5)                                = 0
brk(NULL)                               = 0x55ef3b4ea000
brk(0x55ef3b50b000)                     = 0x55ef3b50b000
read(0, "f", 1)                         = 1
read(0, "\n", 1)                        = 1
write(4, "\1\0\0\0", 4)                 = 4
write(4, "f", 1)                        = 1
read(0, "s", 1)                         = 1
read(0, "\n", 1)                        = 1
write(6, "\1\0\0\0", 4)                 = 4
write(6, "s", 1)                        = 1
read(0, "h", 1)                         = 1
read(0, "e", 1)                         = 1
read(0, "l", 1)                         = 1
read(0, "l", 1)                         = 1
read(0, "o", 1)                         = 1
```

```
read(0, "\n", 1)                        = 1
write(4, "\5\0\0\0", 4)                  = 4
write(4, "hello", 5)                     = 5
…
read(0, "", 1)                          = 0
close(4)                                = 0
close(6)                                = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=233, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
exit_group(0)                           = ?
+++ exited with 0 +++
```

Описание системных вызовов

execve("./laba_2", ["./laba_2"], 0x7ffc9b12e670 / 27 vars */) = 0*

Исполняет программу ./laba_2 с ключом ./laba_2 и также передаются 27 переменных окружения.

brk(NULL)   = 0x55ef3b4ea000

Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти.

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, если не существует или нет прав на чтение.

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

Открывает /etc/ld.so.cache относительно каталога вызывающего процесса (AT_FDCWD) с правами доступа на чтение и закрытие при завершении процесс (O_RDONLY|O_CLOEXEC). Возвращает файловый дескриптор для файла.

fstat(3, {st_mode=S_IFREG|0644, st_size=41434, ...}) = 0

Заполняет структуру, указанную вторым аргументом, fstat информацией о файле с файловым дескриптором 3.

mmap(NULL, 41434, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f218bd64000

Создает отображение файла с файловым дескриптором 3 в память, начиная с адреса NULL, то есть ядро система само определит адрес, длины 41434 байт, с правами на чтение (PROT_READ), создает неразделяемое отражение с механизмом copy-on-write (MAP_PRIVATE), со смещением в файловом дескрипторе равным 0. Возвращает указатель на начало отраженной памяти = 0x7f218bd64000.

close(3) = 0

Закрывает файловый дескриптор.

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

Читает 832 байта из 3 файлового дескриптора. Возвращает количество прочитанных байт.

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

Читает 784 байт из 3 файлового дескриптора с начальным смещением равным 64. Смешение для файлового дескриптора не изменяется. Возвращает количество прочитанных байт.

mprotect(0x7f218bb95000, 1847296, PROT_NONE) = 0

Контролирует доступ к области памяти, начинающейся с адреса 0x7f218bb95000 и длиной 1847296 байт, доступ к памяти запрещен (PROT_NONE). Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV.

arch_prctl(ARCH_SET_FS, 0x7f218bd63540) = 0

Устанавливает специфичное для архитектуры состояние процесса или треда. Устанавливает 64 битную базу для регистра FS (ARCH_SET_FS) в значение 0x7f218bd63540.

munmap(0x7f218bd64000, 41434) = 0

Удаляет все отражения, начиная с адреса 0x7f218bd64000 длины 41434.

pipe([5, 6]) = 0

Создает пару файловых описателей, указывающих на запись inode именованного канала, и помещает их в массив. Файловый описатель равный 5 предназначен для чтения, а 6 для записи.

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID |CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f218bd63810) = 232

Создаёт новый процесс. Очищает TID для ребенка, но не для родителя, записывает TID ребенка в адрес 0x7f218bd63810. Создает сигнал для родителя SIGCHLD, вызываем при изменении статуса ребенка. Возвращает TID ребенка.

write(4, "\1\0\0\0", 4) = 4

Записывает в файловый дескриптор 4 строку \1\0\0\0 размером 4 байт. Возвращает количество записанных байт.

Выводы

Выполнив данную лабораторную работу, я узнал, что при помощи утилиты strace можно удобно просматривать системные вызовы программы – это позволит мне в будущем искать ошибки, которые могут возникнуть при написании кода. Также я познакомился со многими системными вызовами, которые позволяют создавать процессы, читать из файлов, писать в них и даже отображать в память программы.