Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

Студент: Кажекин Д.А.
Группа: М8О-207Б-21
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/DKazhekin/OS

Постановка задачи

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

Задание

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.
Для уменьшения размеров отчета, проведу диагностику для второй ЛР.

Общие сведения о программе

Для диагностики работы программного обеспечения используется утилита strace.

В программе используются следующие системные вызовы:

1. arch_prctl - установить состояние треда, специфичное для архитектуры
2. access - проверить права доступа пользователя к файлу
3. openat, open – открывает файл
4. mmap, munmap - отражает файлы или устройства в памяти или снимает их отражение
5. stat, fstat, lstat - считывает статус файла
6. brk, sbrk - изменение размера сегмента данных
7. pipe - создает канал
8. clone - создать процесс-потомок
9. lseek - установить смещение для позиционирования операций чтения/записи
10. futex - системный вызов быстрых связей пространства пользователя
11. madvise - выдает предложения об использовании памяти
12. exit - обычное завершение работы программы
13. execve - выполняет программу, заданную параметром *filename*

Подробнее об access, pipe, open:

int access(const char * pathname, int mode**)** – проверяет, имеет ли процесс права на чтение или запись, или же просто проверяет, существует ли файл (или другой объект файловой

системы), с именем pathname. Если pathname является символьной ссылкой, то проверяются права доступа к файлу, на который она ссылается.

mode -- это маска, состоящая из одного или более флагов **R_OK**, **W_OK**, **X_OK** и **F_OK**.

R_OK, **W_OK** и **X_OK** запрашивают соответственно проверку существования файла и возможности его чтения, записи или выполнения. **F_OK** просто проверяет существование файла.

int pipe(int filedes[2]**)** – создает пару файловых описателей, указываюших на запись inode именованного канала, и помещает их в массив, на который указывает fildes. fildes[0] предназначен для чтения, а fildes[1] предназначен для записи

int open(const char *pathname, int flags**)** - вызов **open()** используется, чтобы преобразовать путь к файлу в описатель файла (небольшое неотрицательно целое число, которое используется с вызовами **read**, **write** и т.п. при последующем вводе-выводе). Если системный вызов завершается успешно, возвращенный файловый описатель является наименьшим описателем, который еще не открыт процессом. В результате этого вызова появляется новый открытый файл, не разделяемый никакими процессами (разделяемые открытые файлы могут возникнуть, когда посылается системный вызов **fork**(2)). Новый описатель файла будет оставаться открытым при выполнении функции **exec**(2) (смотри описание **fcntl**(2)). Указатель устанавливается в начале файла. Параметр *flags* - это флаги **O_RDONLY**, **O_WRONLY** или **O_RDWR**, открывающие файлы "только для чтения", "только для записи" и для чтения и записи соответственно.

Демонстрация работы программы

```
strace -f ./parent
execve("./parent", ["./parent"], 0x7ffcf8cb55b8 /* 48 vars */) = 0
brk(NULL)                               = 0x5585f64ec000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdb1d08720) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=82021, ...}) = 0
mmap(NULL, 82021, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f40fb19f000
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\300A\2\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", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880) = 68

fstat(3, {st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f40fb19d000

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", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880) = 68

mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f40fafab000

mmap(0x7f40fafcd000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f40fafcd000

mmap(0x7f40fb145000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) = 0x7f40fb145000

mmap(0x7f40fb193000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f40fb193000

mmap(0x7f40fb199000, 13920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f40fb199000

close(3) = 0

arch_prctl(ARCH_SET_FS, 0x7f40fb19e540) = 0

mprotect(0x7f40fb193000, 16384, PROT_READ) = 0

mprotect(0x5585f512e000, 4096, PROT_READ) = 0

mprotect(0x7f40fb1e1000, 4096, PROT_READ) = 0

munmap(0x7f40fb19f000, 82021) = 0

pipe([3, 4]) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f40fb19e810) = 57396

strace: Process 57396 attached

[pid 57395] fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

[pid 57395] brk(NULL <unfinished ...>

[pid 57396] read(3, <unfinished ...>

[pid 57395] <... brk resumed>) = 0x5585f64ec000

[pid 57395] brk(0x5585f650d000) = 0x5585f650d000

[pid 57395] write(1, "Enter the filename:\n", 20Enter the filename:

) = 20

[pid 57395] fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

[pid 57395] read(0, test.txt

"test.txt\n", 1024) = 9

[pid 57395] close(3)   = 0

[pid 57395] write(4, "\10\0\0\0", 4)   = 4

[pid 57396] <... read resumed>"\10\0\0\0", 4) = 4

[pid 57395] write(4, "test.txt\0\302\22\365\205U\0\0\360\207\320\261\375\177\0\0\0\"(\203R8\226{", 32 <unfinished ...>

[pid 57396] read(3, <unfinished ...>

[pid 57395] <... write resumed>)   = 32

[pid 57395] close(4 <unfinished ...>

[pid 57396] <... read resumed>"test.txt", 8) = 8

[pid 57395] <... close resumed>)   = 0

[pid 57396] close(4 <unfinished ...>

[pid 57395] mknod("myfifo", S_IFIFO|0777 <unfinished ...>

[pid 57396] <... close resumed>)   = 0

[pid 57395] <... mknod resumed>)   = -1 EEXIST (Файл существует)

[pid 57396] close(4 <unfinished ...>

[pid 57395] openat(AT_FDCWD, "myfifo", O_WRONLY <unfinished ...>

[pid 57396] <... close resumed>)   = -1 EBADF (Неправильный дескриптор файла)

[pid 57396] execve("./child", ["./child", "test.txt\200\302\22\365\205U"], 0x7ffdb1d08808 /* 48 vars */) = 0

[pid 57396] brk(NULL)   = 0x55c866a5f000

[pid 57396] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe08e2b410) = -1 EINVAL (Недопустимый аргумент)

[pid 57396] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

[pid 57396] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 4

[pid 57396] fstat(4, {st_mode=S_IFREG|0644, st_size=82021, ...}) = 0

[pid 57396] mmap(NULL, 82021, PROT_READ, MAP_PRIVATE, 4, 0) = 0x7f4c3d37c000

[pid 57396] close(4)   = 0

[pid 57396] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 4

[pid 57396] read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300A\2\0\0\0\0"..., 832) = 832

[pid 57396] pread64(4, "\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

[pid 57396] pread64(4, "\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

[pid 57396] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880) = 68

[pid 57396] fstat(4, {st_mode=S_IFREG|0755, st_size=2029592, ...}) = 0

[pid 57396] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f4c3d37a000

[pid 57396] pread64(4, "\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

[pid 57396] pread64(4, "\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

[pid 57396] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880)
= 68

[pid 57396] mmap(NULL, 2037344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) =
0x7f4c3d188000

[pid 57396] mmap(0x7f4c3d1aa000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x22000) = 0x7f4c3d1aa000

[pid 57396] mmap(0x7f4c3d322000, 319488, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x19a000) = 0x7f4c3d322000

[pid 57396] mmap(0x7f4c3d370000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1e7000) = 0x7f4c3d370000

[pid 57396] mmap(0x7f4c3d376000, 13920, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f4c3d376000

[pid 57396] close(4)   = 0

[pid 57396] arch_prctl(ARCH_SET_FS, 0x7f4c3d37b540) = 0

[pid 57396] mprotect(0x7f4c3d370000, 16384, PROT_READ) = 0

[pid 57396] mprotect(0x55c866485000, 4096, PROT_READ) = 0

[pid 57396] mprotect(0x7f4c3d3be000, 4096, PROT_READ) = 0

[pid 57396] munmap(0x7f4c3d37c000, 82021) = 0

[pid 57396] openat(AT_FDCWD, "myfifo", O_RDONLY) = 4

[pid 57396] read(4, <unfinished ...>

[pid 57395] <... openat resumed>)   = 3

[pid 57395] write(3, "\4\0\0\0", 4)   = 4

[pid 57396] <... read resumed>"\4\0\0\0", 4) = 4

[pid 57395] write(1, "Enter 4 different digits\n", 25 <unfinished ...>

[pid 57396] read(4, Enter 4 different digits

 <unfinished ...>

[pid 57395] <... write resumed>)   = 25

[pid 57395] read(0, 3 4 5 6

"3 4 5 6\n", 1024) = 8

[pid 57395] write(3, "\0\0@@", 4)   = 4

[pid 57396] <... read resumed>"\0\0@@", 4) = 4

[pid 57395] write(3, "\0\0\200@", 4 <unfinished ...>

[pid 57396] read(4, <unfinished ...>

[pid 57395] <... write resumed>)   = 4

[pid 57396] <... read resumed>"\0\0\200@", 4) = 4

[pid 57395] write(3, "\0\0\240@", 4 <unfinished ...>

[pid 57396] read(4, <unfinished ...>

[pid 57395] <... write resumed>)   = 4

[pid 57396] <... read resumed>"\0\0\240@", 4) = 4

[pid 57395] write(3, "\0\0\300@", 4 <unfinished ...>

[pid 57396] read(4, <unfinished ...>

[pid 57395] <... write resumed>)   = 4

[pid 57396] <... read resumed>"\0\0\300@", 4) = 4

[pid 57395] close(3)   = 0

[pid 57395] lseek(0, -1, SEEK_CUR <unfinished ...>

[pid 57396] brk(NULL <unfinished ...>

[pid 57395] <... lseek resumed>)   = -1 ESPIPE (Недопустимая операция смещения)

[pid 57396] <... brk resumed>)   = 0x55c866a5f000

[pid 57395] exit_group(0 <unfinished ...>

[pid 57396] brk(0x55c866a80000 <unfinished ...>

[pid 57395] <... exit_group resumed>)   = ?

[pid 57396] <... brk resumed>)   = 0x55c866a80000

[pid 57396] openat(AT_FDCWD, "test.txt\200\302\22\365\205U", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 5

[pid 57396] fstat(5, <unfinished ...>

[pid 57395] +++ exited with 0 +++

<... fstat resumed>{st_mode=S_IFREG|0664, st_size=0, ...}) = 0

write(5, "The answer is: 18.00\n", 21) = 21

close(5)   = 0

close(4)   = 0

exit_group(0)   = ?

Выводы

Данная лабораторная работа была очень полезной. Я приобрел практические навыки в диагностике работы программного обеспечения.