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: Тумаков Данила Владимирович

Группа: М80 – 206Б-19

Преподаватель: Соколов Андрей Алексеевич

Дата: \_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **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.

Используемые утилиты: strace.

1. **Описание используемых утилит**

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

1. **Примеры использования утилит**

Ниже приведены протоколы strace лабораторной работы 2.

danila@LAPTOP-5N1LT0S0:/mnt/c/VUZ/OS/lab\_2/src$ strace -f ./a.out < ../test/test4

execve("./a.out", ["./a.out"], 0x7ffff907d618 /\* 21 vars \*/) = 0

brk(NULL) = 0x565268938000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffd1a08e60) = -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=31852, ...}) = 0

mmap(NULL, 31852, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f89734c6000

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\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 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) = 0x7f89734c4000

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\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f89732d2000

mprotect(0x7f89732f7000, 1847296, PROT\_NONE) = 0

mmap(0x7f89732f7000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f89732f7000

mmap(0x7f897346f000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f897346f000

mmap(0x7f89734ba000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f89734ba000

mmap(0x7f89734c0000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f89734c0000

close(3) = 0

arch\_prctl(ARCH\_SET\_FS, 0x7f89734c5540) = 0

mprotect(0x7f89734ba000, 12288, PROT\_READ) = 0

mprotect(0x5652674e3000, 4096, PROT\_READ) = 0

mprotect(0x7f89734fb000, 4096, PROT\_READ) = 0

munmap(0x7f89734c6000, 31852) = 0

read(0, "f", 1) = 1

read(0, "1", 1) = 1

read(0, "\n", 1) = 1

openat(AT\_FDCWD, "f1", O\_WRONLY|O\_CREAT|O\_TRUNC, 0106560) = 3

read(0, "f", 1) = 1

read(0, "2", 1) = 1

read(0, "\n", 1) = 1

openat(AT\_FDCWD, "f2", O\_WRONLY|O\_CREAT|O\_TRUNC, 0106560) = 4

pipe([5, 6]) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 79 attached

, child\_tidptr=0x7f89734c5810) = 79

[pid 79] close(6 <unfinished ...>

[pid 78] close(5 <unfinished ...>

[pid 79] <... close resumed>) = 0

[pid 78] <... close resumed>) = 0

[pid 79] read(5, <unfinished ...>

[pid 78] pipe([5, 7]) = 0

[pid 78] clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 80 attached

, child\_tidptr=0x7f89734c5810) = 80

[pid 80] close(6 <unfinished ...>

[pid 78] close(5 <unfinished ...>

[pid 80] <... close resumed>) = 0

[pid 78] <... close resumed>) = 0

[pid 80] close(7 <unfinished ...>

[pid 78] read(0, <unfinished ...>

[pid 80] <... close resumed>) = 0

[pid 80] read(5, <unfinished ...>

[pid 78] <... read resumed>"m", 1) = 1

[pid 78] write(6, "m", 1) = 1

[pid 79] <... read resumed>"m", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] write(3, "m", 1 <unfinished ...>

[pid 78] <... read resumed>" ", 1) = 1

[pid 79] <... write resumed>) = 1

[pid 78] write(6, " ", 1 <unfinished ...>

[pid 79] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] <... read resumed>" ", 1) = 1

[pid 79] write(3, " ", 1) = 1

[pid 78] <... read resumed>"n", 1) = 1

[pid 79] read(5, <unfinished ...>

[pid 78] write(6, "n", 1) = 1

[pid 79] <... read resumed>"n", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] write(3, "n", 1 <unfinished ...>

[pid 78] <... read resumed>" ", 1) = 1

[pid 79] <... write resumed>) = 1

[pid 78] write(6, " ", 1 <unfinished ...>

[pid 79] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 79] <... read resumed>" ", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] write(3, " ", 1 <unfinished ...>

[pid 78] <... read resumed>"o", 1) = 1

[pid 79] <... write resumed>) = 1

[pid 78] write(6, "o", 1 <unfinished ...>

[pid 79] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 79] <... read resumed>"o", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] read(5, <unfinished ...>

[pid 78] <... read resumed>" ", 1) = 1

[pid 78] write(6, " ", 1) = 1

[pid 79] <... read resumed>" ", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 79] write(3, " ", 1) = 1

[pid 78] <... write resumed>) = 1

[pid 80] <... read resumed>" ", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 80] <... read resumed>" ", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 80] write(4, " ", 1 <unfinished ...>

[pid 78] <... read resumed>"k", 1) = 1

[pid 80] <... write resumed>) = 1

[pid 78] write(7, "k", 1 <unfinished ...>

[pid 80] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 80] <... read resumed>"k", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 80] write(4, "k", 1 <unfinished ...>

[pid 78] <... read resumed>" ", 1) = 1

[pid 80] <... write resumed>) = 1

[pid 78] write(7, " ", 1 <unfinished ...>

[pid 80] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 80] <... read resumed>" ", 1) = 1

[pid 78] read(0, <unfinished ...>

[pid 80] write(4, " ", 1 <unfinished ...>

[pid 78] <... read resumed>"l", 1) = 1

[pid 80] <... write resumed>) = 1

[pid 78] write(7, "l", 1 <unfinished ...>

[pid 80] read(5, <unfinished ...>

[pid 78] <... write resumed>) = 1

[pid 78] read(0, <unfinished ...>

[pid 80] <... read resumed>"l", 1) = 1

[pid 80] write(4, "l", 1 <unfinished ...>

[pid 78] <... read resumed>"\n", 1) = 1

[pid 78] write(7, "\n", 1) = 1

[pid 78] read(0, "", 1) = 0

[pid 80] <... write resumed>) = 1

[pid 78] close(6 <unfinished ...>

[pid 80] read(5, <unfinished ...>

[pid 78] <... close resumed>) = 0

[pid 79] <... read resumed>"", 1) = 0

[pid 80] <... read resumed>"\n", 1) = 1

[pid 78] close(7 <unfinished ...>

[pid 79] close(3 <unfinished ...>

[pid 78] <... close resumed>) = 0

[pid 79] <... close resumed>) = 0

[pid 80] write(4, "\n", 1 <unfinished ...>

[pid 78] exit\_group(0 <unfinished ...>

[pid 79] close(5 <unfinished ...>

[pid 78] <... exit\_group resumed>) = ?

[pid 79] <... close resumed>) = 0

[pid 78] +++ exited with 0 +++

[pid 79] exit\_group(0 <unfinished ...>

[pid 80] <... write resumed>) = 1

[pid 79] <... exit\_group resumed>) = ?

[pid 80] read(5, <unfinished ...>

[pid 79] +++ exited with 0 +++

<... read resumed>"", 1) = 0

close(4) = 0

close(5) = 0

exit\_group(0) = ?

+++ exited with 0 +++

1. **Описание**
2. execve("./a.out", ["./a.out"], 0x7ffff907d618 /\* 21 vars \*/) = 0
3. Исполняет программу ./a.out с ключом ", ["./a.out"]. Возвращает 0 – успешное выполнение.
4. brk(NULL) = 0x555bdb126000

Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти = 0x555bdb126000.

1. access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)
2. Проверяет /etc/ld.so.preload на существование и на наличие прав на чтение (R\_OK), возвращает -1 – или не существует /etc/ld.so.preload или нет прав на чтение, errno устанавливается в ENOENT (компонент пути не существует или является "висячей" символической ссылкой).
3. openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

Открывает /etc/ld.so.cache относительно дескриптора указанного каталога - AT\_FDCWD (относительно текущего рабочего каталога вызывающего процесса) с правами доступа - O\_RDONLY|O\_CLOEXEC (на чтение и устанавливает флаг close-on-exec на новом файловом дескрипторе). Возвращает новый файловый дескриптор 3.

1. fstat(3, {st\_mode=S\_IFREG|0644, st\_size=31852, ...}) = 0

Заполняет структуру указанную вторым аргументом fstat информацией об файле с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

1. mmap(NULL, 31852, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f89734c6000

Создает отображение файла с файловым дескриптором 3 в память, начиная с адреса NULL (система сама выбирает), размер = 88176 байт, с правами защиты памяти на чтение PROT\_READ, задает тип отражаемого объекта MAP\_PRIVATE - создает неразделяемое отражение с механизмом copy-on-write, запись в эту область памяти не влияет на файл, не определено, являются или нет изменения в файле после вызова mmap видимыми в отраженном диапазоне. Возвращает указатель на начало отраженной памяти = 0x7f89734c6000.

1. close(3) = 0

Закрывает файл с файловым дескриптором 3. Возвращает 0 – успешное выполнение.

1. 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 данных из файла с файловым дескриптором 3 в буфер указанный вторым аргументом. Возвращает число успешно считанных байт = 832.

1. 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 байтов из файла с файловым дескриптором 3, начиная со смещения 64, в буфер . Текущая позиция файла не изменяется. Возвращает число успешно считанных байт = 784.

1. mprotect(0x7f89732f7000, 1847296, PROT\_NONE) = 0

Контролирует доступ к области памяти начинающейся с адреса 0x7f89732f7000 длины 1847296 байт, доступ к памяти запрещен - PROT\_NONE. Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV. Возвращает 0 – успешное завершение.

1. arch\_prctl(ARCH\_SET\_FS, 0x7f89734c5540) = 0

Устанавливает специфичное для архитектуры состояние. Устанавливает 64 битную базу для регистра FS (ARCH\_SET\_FS) в значение 0x7f89734c5540. Возвращает 0 – успешное выполнение.

1. munmap(0x7f89734c6000, 31852) = 0

Снимает отражение из заданной области памяти, 0x7f89734c6000 указатель на начало памяти, длина = 31852 байт. Возвращает 0 – успешное выполнение.

1. write(4, " Hello wOrld!\nBye Bye"..., 64) = 64

Записывает 64 байта из буфера (второй аргумент) в файл с файловым дескриптором 4. Возвращает число успешно записанных байт = 64.

1. clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 79 attached, child\_tidptr=0x7f89734c5810) = 79

Создает процесс-потомок с флагами - CLONE\_CHILD\_CLEARTID (очищает id), CLONE\_CHILD\_SETTID (устанавливает id), SIGCHLD (сигнал о изменении статуса дочернего процесса), задает положение стека для процесса-потомка = NULL, задает указатель на id = 0x7f89734c5810. Возвращает pid процесса-потомка.

1. pipe([5, 6]) = 0

pipe создает пару файловых описателей [5, 6]. При удачном завершении вызова возвращаемое значение равно нулю. При ошибке возвращается -1, а переменной errno присваивается номер ошибки.

1. [pid 79] – Процесс с id=79 выполняется.
2. Т.к. я в strace указал флаг –f, информация о процессах выводилась подробно, рядом с системным вызовом указывался pid процесс, где происходит системный вызов.
3. **Выводы**

Выполнив данную лабораторную работу, я узнал, что утилита strace помогает просматривать системные вызовы. Это позволяет мне 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.