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: Епанешников Владислав Сергеевич

Группа: М80 – 206Б-19

Преподаватель: Миронов Евгений Сергеевич

Дата: 21.04.2021

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.

vladislav@Alekseev:/mnt/c/VScode/3sem/OS/Vlad$ strace -f ./main < test.txt

execve("./a.out", ["./a.out", "test.txt", "res.txt"], 0x7ffd1abdac28 /\* 50 vars \*/) = 0

brk(NULL) = 0x555bdb126000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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=88176, ...}) = 0

mmap(NULL, 88176, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4ece6b2000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000b\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4ece6b0000

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4ece282000

mprotect(0x7f4ece29c000, 2093056, PROT\_NONE) = 0

mmap(0x7f4ece49b000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f4ece49b000

mmap(0x7f4ece49d000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4ece49d000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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\260\34\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4ecde91000

mprotect(0x7f4ece078000, 2097152, PROT\_NONE) = 0

mmap(0x7f4ece278000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f4ece278000

mmap(0x7f4ece27e000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4ece27e000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4ece6ad000

arch\_prctl(ARCH\_SET\_FS, 0x7f4ece6ad740) = 0

mprotect(0x7f4ece278000, 16384, PROT\_READ) = 0

mprotect(0x7f4ece49b000, 4096, PROT\_READ) = 0

mprotect(0x555bda0ec000, 4096, PROT\_READ) = 0

mprotect(0x7f4ece6c8000, 4096, PROT\_READ) = 0

munmap(0x7f4ece6b2000, 88176) = 0

set\_tid\_address(0x7f4ece6ada10) = 1613

set\_robust\_list(0x7f4ece6ada20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f4ece287cb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f4ece294890}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f4ece287d50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f4ece294890}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

openat(AT\_FDCWD, "test.txt", O\_RDWR) = 3

openat(AT\_FDCWD, "res.txt", O\_RDWR|O\_CREAT, 0600) = 4

fstat(3, {st\_mode=S\_IFREG|0664, st\_size=64, ...}) = 0

ftruncate(4, 64) = 0

read(3, " Hello wOrld!\nBye Bye"..., 64) = 64

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

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4ece6ada10) = 1614

strace: Process 1614 attached

[pid 1613] clone( <unfinished ...>

[pid 1614] set\_robust\_list(0x7f4ece6ada20, 24) = 0

[pid 1614] nanosleep({tv\_sec=1, tv\_nsec=0}, <unfinished ...>

[pid 1613] <... clone resumed> child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4ece6ada10) = 1615

strace: Process 1615 attached

[pid 1613] nanosleep({tv\_sec=2, tv\_nsec=0}, <unfinished ...>

[pid 1615] set\_robust\_list(0x7f4ece6ada20, 24) = 0

[pid 1615] nanosleep({tv\_sec=1, tv\_nsec=0}, <unfinished ...>

[pid 1614] <... nanosleep resumed> 0x7ffd8e1886d0) = 0

[pid 1615] <... nanosleep resumed> 0x7ffd8e1886d0) = 0

[pid 1614] mmap(NULL, 64, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0 <unfinished ...>

[pid 1615] mmap(NULL, 64, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0 <unfinished ...>

[pid 1614] <... mmap resumed> ) = 0x7f4ece6c7000

[pid 1615] <... mmap resumed> ) = 0x7f4ece6c7000

[pid 1614] munmap(0x7f4ece6c7000, 64) = 0

[pid 1614] nanosleep({tv\_sec=1, tv\_nsec=0}, <unfinished ...>

[pid 1615] brk(NULL) = 0x555bdb126000

[pid 1615] brk(0x555bdb147000) = 0x555bdb147000

[pid 1615] --- SIGSEGV {si\_signo=SIGSEGV, si\_code=SEGV\_MAPERR, si\_addr=0x7f4ece6c6ff8} ---

[pid 1615] +++ killed by SIGSEGV (core dumped) +++

[pid 1613] <... nanosleep resumed> {tv\_sec=0, tv\_nsec=793520618}) = ? ERESTART\_RESTARTBLOCK (Interrupted by signal)

[pid 1613] --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_DUMPED, si\_pid=1615, si\_uid=1000, si\_status=SIGSEGV, si\_utime=0, si\_stime=0} ---

[pid 1613] restart\_syscall(<... resuming interrupted nanosleep ...>) = 0

[pid 1613] wait4(1615, [{WIFSIGNALED(s) && WTERMSIG(s) == SIGSEGV && WCOREDUMP(s)}], WNOHANG, NULL) = 1615

[pid 1613] wait4(1614, 0x7ffd8e188768, WNOHANG, NULL) = 0

[pid 1613] mmap(NULL, 64, PROT\_READ, MAP\_SHARED, 4, 0) = 0x7f4ece6c7000

[pid 1613] fstat(1, {st\_mode=S\_IFCHR|0666, st\_rdev=makedev(1, 3), ...}) = 0

[pid 1614] <... nanosleep resumed> 0x7ffd8e1886d0) = 0

[pid 1613] ioctl(1, TCGETS <unfinished ...>

[pid 1614] close(3 <unfinished ...>

[pid 1613] <... ioctl resumed> , 0x7ffd8e188560) = -1 ENOTTY (Inappropriate ioctl for device)

[pid 1614] <... close resumed> ) = 0

[pid 1614] close(4) = 0

[pid 1613] brk(NULL) = 0x555bdb126000

[pid 1613] brk(0x555bdb147000) = 0x555bdb147000

[pid 1613] munmap(0x7f4ece6c7000, 64) = 0

[pid 1614] exit\_group(0 <unfinished ...>

[pid 1613] nanosleep({tv\_sec=2, tv\_nsec=0}, <unfinished ...>

[pid 1614] <... exit\_group resumed>) = ?

[pid 1614] +++ exited with 0 +++

<... nanosleep resumed> {tv\_sec=1, tv\_nsec=999911259}) = ? ERESTART\_RESTARTBLOCK (Interrupted by signal)

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=1614, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

restart\_syscall(<... resuming interrupted nanosleep ...>) = 0

close(3) = 0

close(4) = 0

write(1, " HELLO WORLD!\nBYE BYE\n HOT CHILL"..., 65) = 65

exit\_group(0) = ?

+++ exited with 0 +++

1. **Описание**
2. execve("./a.out", ["./a.out", "test.txt", "res.txt"], 0x7ffd1abdac28 /\* 50 vars \*/) = 0
3. Исполняет программу ./a.out с ключами ", ["./a.out", "test.txt", "res.txt"]. Возвращает 0 – успешное выполнение.
4. brk(NULL) = 0x555bdb126000
5. Устанавливает конец сегмента данных в значение NULL, возвращает указатель на начало новой области памяти = 0x555bdb126000.
6. access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)
7. access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)
8. Проверяет /etc/ld.so.nohwcap на существование (F\_OK), возвращает -1 – не существует /etc/ld.so.nohwcap, errno устанавливается в ENOENT (компонент пути не существует или является "висячей" символической ссылкой). 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).
9. openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3
10. 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.
11. fstat(3, {st\_mode=S\_IFREG|0644, st\_size=88176, ...}) = 0
12. Заполняет структуру указанную вторым аргументом fstat информацией об файле с файловым дескриптором 3. Возвращает 0 – успешное выполнение.
13. mmap(NULL, 88176, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4ece6b2000
14. Создает отображение файла с файловым дескриптором 3 в память, начиная с адресса 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 = 0x7f4ece6b2000.
15. close(3) = 0
16. Закрывает файл с файловым дескриптором 3. Возвращает 0 – успешное выполнение.
17. read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000b\0\0\0\0\0\0"..., 832) = 832
18. Читает 832 байта данных из файла с файловым дескриптором 3 в буффер указанный вторым аргументом. Возвращает число успешно считанных байт = 832.
19. mprotect(0x7f4ece29c000, 2093056, PROT\_NONE) = 0
20. Контролирует доступ к области памяти начинающейся с адресса 0x7f4ece29c000 длины 2093056 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.
21. arch\_prctl(ARCH\_SET\_FS, 0x7f4ece6ad740) = 0
22. Устанавливает специфичное для архитектуры состояние. Устанавливает 64 битную базу для регистра FS (ARCH\_SET\_FS) в значение 0x7f4ece6ad740. Возвращает 0 – успешное выполнение.
23. munmap(0x7f4ece6b2000, 88176) = 0
24. Снимает отражение из заданной области памяти, 0x7f4ece6b2000 указатель на начало памяти, длина = 88176 байт. Возвращает 0 – успешное выполнение.
25. set\_tid\_address(0x7f4ece6ada10) = 1613
26. Устанавливает у вызывающего процесса значение clear\_child\_tid равным 0x7f4ece6ada10. Возвращает id = 1613 вызывающего процесса.
27. set\_robust\_list(0x7f4ece6ada20, 24) = 0
28. Устнавливает начало списка надёжных фьютексов в 0x7f4ece6ada20, размером 24 байта. Возвращает 0 – успешное выполнение.
29. rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f4ece287cb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f4ece294890}, NULL, 8) = 0
30. Изменяет действие, предпринимаемое процессом при получении сигнала SIGRTMIN. Возвращает 0 – успешное выполнение.
31. rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0
32. Выбор и/или изменение маски сигнала SIG\_UNBLOCK вызывающего потока. Возвращает 0 – успешное выполнение.
33. prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0
34. Устанавливает лимит ресурсов для процесса с pid = 0, RLIMIT\_STACK – максимальный размер стека процессов. NULL – новый лимит (const struct rlimit\*), {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY} – новый лимит, rlim\_cur – текущий размер, rlim\_max – максимальный размер, RLIM64\_INFINITY – нет ограничения по размеру. Возвращает 0 – успешное выполнение.
35. ftruncate(4, 64) = 0
36. Изменяет длину для файла с файловым дескриптором 4 на длину = 64 байт. Если размер увеличивается новые данный заполняются нулевыми байтами, если размер уменьшается данные обрезаются. Возвращает 0 – успешное выполнение.
37. write(4, " Hello wOrld!\nBye Bye"..., 64) = 64
38. Записывает 64 байта из буффера (второй аргумент) в файл с файловым дескриптором 4. Возвращает число успешно записанных байт = 64.

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4ece6ada10) = 1614

1. Создает процесс-потомок с фалагами - 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 = 0x7f4ece6ada10. Возвращает pid процесса-потомка.
3. [pid 1614] nanosleep({tv\_sec=1, tv\_nsec=0}, <unfinished ...>
4. Останавливает работу процесса с pid = 1614 на 1 сек. <unfinished ...> - означает начало вызова.
5. <... nanosleep resumed> 0x7ffd8e1886d0) = 0 - означает конец вызова и возвращает 0 – успешное выполнение.
6. [pid 1613] restart\_syscall(<... resuming interrupted nanosleep ...>) = 0
7. В процессе с pid = 1613 перезапускает системный вызов - <... resuming interrupted nanosleep ...>. Возвращает 0 – возвращаемое значение перезапущенного системного вызова - <... resuming interrupted nanosleep ...>.
8. [pid 1613] wait4(1615, [{WIFSIGNALED(s) && WTERMSIG(s) == SIGSEGV && WCOREDUMP(s)}], WNOHANG, NULL) = 1615
9. В процессе с pid = 1613 ждет завершение работы процесса с pid = 1615, статус - , [{WIFSIGNALED(s) && WTERMSIG(s) == SIGSEGV && WCOREDUMP(s)}], немедленно возвращает управление вызывающему процессу, если нет дочерних процессов, попадающих под действие функции при помощи WNOHANG, rusage\* = NULL – информация не будет записана в struct rusage. Возвращает pid =1615 дочернего процесса, завершившего работу.
10. Т.к. я в strace указал флаг –f, информация о процессах выводилась подробно, рядом с системным вызовом указывался pid процесс, где происходит системный вызов:
11. [pid 1613] restart\_syscall(<... resuming interrupted nanosleep ...>) = 0
12. Системный вызов в процессе с pid = 1613.
13. **Выводы**

Утилиты strace и ltrace – простые и надежные инструменты для отслеживания системных вызовов. Они полезны в тех случаях, когда программисту нужно отладить и тестировать программу. Выполнив данную лабораторную работу, я узнал, что при помощи утилиты strace можно удобно просматривать системные вызовы. Это позволит мне в будущем искать ошибки, которые возникнут 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.