Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету «Операционные системы» №8

Студент: Пирязев М.А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 22.12.2023

Оценка:

Подпись:

Москва 2023 г.

Оглавление

Цель работы 3
Постановка задачи 3
Общие сведения об утилите..... 3
Пример использования утилиты..... 4
Вывод 9

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

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.

Общие сведения об утилите

Утилита strace используется для отслеживания системных вызовов и сигналов, которые происходят во время выполнения программы. Она позволяет анализировать взаимодействие программы с операционной системой, отображая вызовы системных функций, их аргументы и возвращаемые значения. Strace полезна для отладки, профилирования и анализа производительности программ, а также для выявления проблем взаимодействия с операционной системой. У данной утилиты существует огромное количество ключей, однако самым удобным для выполнения лабораторной работы мне показался -o, этот ключ превращает большой список логов программы в скромную табличку системных вызовов, разграниченных по времени. Так же выходные данные этой утилиты можно направить в файл, указав его название, что довольно удобно.

Пример использования утилиты

В качестве обрабатываемого кода утилитой была использована лабораторная работа №2.

```
execve("./main", ["./main", "10"], 0x7ffc8adc3c38 /* 29 vars */) = 0
brk(NULL)                               = 0x5591fa544000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc3a33adb0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6874ee000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=18843, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18843, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc6874e9000
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\0P\237\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 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
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
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fc6872c1000
mmap(0x7fc6872e9000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7fc6872e9000
mmap(0x7fc68747e000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7fc68747e000
mmap(0x7fc6874d6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x214000) = 0x7fc6874d6000
mmap(0x7fc6874dc000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
= 0x7fc6874dc000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6872be000
arch_prctl(ARCH_SET_FS, 0x7fc6872be740) = 0
set_tid_address(0x7fc6872bea10)         = 3885
set_robust_list(0x7fc6872bea20, 24)     = 0
rseq(0x7fc6872bf0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fc6874d6000, 16384, PROT_READ) = 0
```

mprotect(0x5591f88b2000, 4096, PROT_READ) = 0

mprotect(0x7fc687528000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7fc6874e9000, 18843) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x3), ...}, AT_EMPTY_PATH) = 0

getrandom("\xbb\x76\x57\xe1\x06\xe2\xfe\x24", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x5591fa544000

brk(0x5591fa565000) = 0x5591fa565000

write(1, "Amount_of_threads is 10\n", 24Amount_of_threads is 10

) = 24

rt_sigaction(SIGRT_1, {sa_handler=0x7fc6873528f0, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fc687303520}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc686abd000

mprotect(0x7fc686abe000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc6872bd910,
parent_tid=0x7fc6872bd910, exit_signal=0, stack=0x7fc686abd000, stack_size=0x7fff00, tls=0x7fc6872bd640} =>
{parent_tid=[3886]}, 88) = 3886

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc68629c000

mprotect(0x7fc68629d000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc686a9c910,
parent_tid=0x7fc686a9c910, exit_signal=0, stack=0x7fc68629c000, stack_size=0x7fff00, tls=0x7fc686a9c640} =>
{parent_tid=[3887]}, 88) = 3887

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc685a9b000

mprotect(0x7fc685a9c000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc68629b910,
parent_tid=0x7fc68629b910, exit_signal=0, stack=0x7fc685a9b000, stack_size=0x7fff00, tls=0x7fc68629b640} =>
{parent_tid=[3888]}, 88) = 3888

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc68529a000

mprotect(0x7fc68529b000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc685a9a910,
parent_tid=0x7fc685a9a910, exit_signal=0, stack=0x7fc68529a000, stack_size=0x7fff00, tls=0x7fc685a9a640} =>
{parent_tid=[0]}, 88) = 3889

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc684a99000

mprotect(0x7fc684a9a000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc685299910, parent_tid=0x7fc685299910, exit_signal=0, stack=0x7fc684a99000, stack_size=0x7fff00, tls=0x7fc685299640} => {parent_tid=[3890]}, 88) = 3890

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc684298000

mprotect(0x7fc684299000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc684a98910, parent_tid=0x7fc684a98910, exit_signal=0, stack=0x7fc684298000, stack_size=0x7fff00, tls=0x7fc684a98640} => {parent_tid=[3891]}, 88) = 3891

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc67f7ff000

mprotect(0x7fc67f800000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc67ffff910, parent_tid=0x7fc67ffff910, exit_signal=0, stack=0x7fc67f7ff000, stack_size=0x7fff00, tls=0x7fc67ffff640} => {parent_tid=[3892]}, 88) = 3892

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc67effe000

mprotect(0x7fc67efff000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc67f7fe910, parent_tid=0x7fc67f7fe910, exit_signal=0, stack=0x7fc67effe000, stack_size=0x7fff00, tls=0x7fc67f7fe640} => {parent_tid=[3893]}, 88) = 3893

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc67e7fd000

mprotect(0x7fc67e7fe000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc67effd910, parent_tid=0x7fc67effd910, exit_signal=0, stack=0x7fc67e7fd000, stack_size=0x7fff00, tls=0x7fc67effd640} => {parent_tid=[3894]}, 88) = 3894

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7fc67dffc000

mprotect(0x7fc67dffd000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fc67e7fc910, parent_tid=0x7fc67e7fc910, exit_signal=0, stack=0x7fc67dffc000, stack_size=0x7fff00, tls=0x7fc67e7fc640} => {parent_tid=[0]}, 88) = 3895
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
munmap(0x7fc686abd000, 8392704)         = 0
munmap(0x7fc68629c000, 8392704)         = 0
munmap(0x7fc685a9b000, 8392704)         = 0
munmap(0x7fc68529a000, 8392704)         = 0
munmap(0x7fc684a99000, 8392704)         = 0
munmap(0x7fc684298000, 8392704)         = 0
write(1, "Execution_time: 23.8079999999999"..., 43Execution_time: 23.80799999999999982947 ms
) = 43
write(1, " ", 1 )                       = 1
exit_group(0)                           = ?
+++ exited with 0 +++
```

Несмотря на огромные размеры и громоздкость этого текста, некоторая логика в нем все – таки присутствует. Для наглядности системных вызовов запустим strace с ключем -c.

% time	seconds	usecs/call	calls	errors	syscall
28.56	0.001829	182	10		clone3
28.12	0.001801	85	21		rt_sigprocmask
12.49	0.000800	44	18		mmap
12.46	0.000798	61	13		mprotect
9.56	0.000612	87	7		munmap
6.04	0.000387	129	3		write
1.51	0.000097	32	3		brk
1.25	0.000080	80	1		rt_sigaction
0.00	0.000000	0	1		read
0.00	0.000000	0	2		close
0.00	0.000000	0	4		pread64
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	2		openat
0.00	0.000000	0	3		newfstatat
0.00	0.000000	0	1		set_robust_list
0.00	0.000000	0	1		prlimit64
0.00	0.000000	0	1		getrandom
0.00	0.000000	0	1		rseq
100.00	0.006404	66	97	2	total

Такая таблица воспринимается гораздо проще. Вкратце разберем системные вызовы.

- `mmap`: Выделяет память в адресном пространстве процесса или устанавливает отображение файла в память.

- `rt_sigprocmask`: Устанавливает маску сигналов процесса.

- `execve`: Заменяет текущий процесс новым процессом, загружая и выполняя новую программу.

- `mprotect`: Изменяет защиту памяти, позволяя установить различные права доступа к областям памяти, таким как чтение, запись или выполнение.

- `munmap`: Удаляет отображение файла из памяти или освобождает выделенную ранее память, возвращая её в систему.

- `pread64`: Читает данные из файла в заданное место в буфере.

- `write`: Записывает данные в файл.

- `newfstatat`: Возвращает информацию о файле, подобно fstat, но использует относительный путь.

- `openat`: Открывает файл или директорию с использованием относительного пути.

- `brk`: Изменяет размер "кучи" процесса, используемой для динамического выделения памяти.

- `arch_prctl`: Управляет архитектурными регистрами процессора.

Вывод

После выполнения лабораторной работы по изучению утилиты strace, я сделал бы выводы о том, как использовать strace для отслеживания системных вызовов и сигналов, анализа взаимодействия программы с операционной системой, отладки, профилирования и анализа производительности программ. Я бы также описал, как strace может быть полезна для выявления проблем взаимодействия программы с операционной системой. Использование утилиты strace дает нам широкий обзор системных вызовов, выполняемых программой, и может помочь в понимании её работы и оптимизации. Это очень полезный инструмент для программистов и системных администраторов.