**Московский авиационный институт (национальный исследовательский университет)**

Факультет информационных технологий и прикладной математики Кафедра вычислительной математики и программирования

Лабораторная работа по предмету "операционные системы" №1

Студент: Мокеева С.А.

Преподаватель: Соколов А.А.

Группа: М8О-206Б-20

Дата: 12.04.2022

Оценка:

Подпись:

**Москва 2022г.**

**Постановка задачи**

Задача: подробно рассказать о каждом системном вызове, отображенном утилитой strace. Разбор системных вызовов осуществляется на примере четвертой лабораторной работы курса “Операционные системы”.

**Работа с утилитой strace**

Самый простой способ запуска утилиты - просто передать ей в параметрах имя команды или исполняемый файл программы, которую мы хотим исследовать.

sophie@sophie-VirtualBox:~/os/os2$ strace ./run2.sh

execve("./run2.sh", ["./run2.sh"], 0x7ffef6040900 /\* 61 vars \*/) = 0

int execve(const char \*filename, char \*const argv [], char \*const envp[]);

argv -- это массив строк, аргументов новой программы

envp -- это массив строк в формате key=value

brk(NULL) = 0x555a4f414000

int brk(void \*end\_data\_segment);

brk устанавливает конец сегмента данных в значение, указанное в аргументе end\_data\_segment, когда это значение является приемлимым, система симулирует нехватку памяти и процесс не достигает своего максимально возможного размера сегмента данных

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff5fbe36f0) = -1 EINVAL (Invalid argument)

arch\_prctl - установить состояние треда, специфичное для архитектуры

int arch\_prctl(int code, unsigned long addr)

Параметр code выбирает подфункцию и передаёт ей аргумент addr

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

int access(const char \*pathname, int mode);

access проверяет, имеет ли процесс права на чтение или запись, или же просто проверяет, существует ли файл (или другой объект файловой системы), с именем pathname

R\_OK – возможность чтения

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

int openat(int dirfd, const char \*pathname, int flags);

используется, чтобы преобразовать путь к файлу в описатель файла (небольшое неотрицательно целое число, которое используется с вызовами)

Если pathname является относительным, а dirfd является специальным значением AT\_FDCWD, то pathname интерпретируется относительно текущего рабочего каталога вызывающего процесса.

Возвращает новый файловый дескриптор

O\_RDONLY – открывается только для чтения

O\_CLOEXEC – разрешите флаг закрытия при выполнении для нового файлового дескриптора.

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=66821, ...}) = 0

int fstat(int fd, struct stat \*buf);

регистрирует файл, на который определяется файловым дескриптором fd, и заполняет buf.

S\_IFREG - regular file

st\_size field gives the size of the file in bytes

mmap(NULL, 66821, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f6e13ba8000

void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

mmap() создает новое отображение в виртуальном адресном пространстве вызывающего процесса. Начальный адрес для нового сопоставления указан в addr. Аргумент длины указывает длину отображения.

Если адрес равен NULL, то ядро выбирает адрес для создания отображения; это наиболее портативный метод создания нового сопоставления.

Содержимое файлового сопоставления инициализируется с использованием длины байтов, начиная со смещения смещения offset в файле (или другом объекте), на которое ссылается файловый дескриптор fd.

Аргумент prot описывает желаемую защиту памяти отображения

close(3) = 0

int close(int fd);

close() закрывает дескриптор файла, чтобы он больше не ссылался ни на какой файл и мог использоваться повторно.

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libtinfo.so.6", 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\0\240\346\0\0\0\0\0\0"..., 832) = 832

ssize\_t read(int fd, void \*buf, size\_t count);

read() пытается прочитать количество байтов count из файлового дескриптора fd в буфер, начиная с buf.

В случае успеха возвращает количество байт

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=192032, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f6e13ba6000

mmap(NULL, 194944, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f6e13b76000

mmap(0x7f6e13b84000, 61440, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f6e13b84000

mmap(0x7f6e13b93000, 57344, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f6e13b93000

mmap(0x7f6e13ba1000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2a000) = 0x7f6e13ba1000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libdl.so.2", 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\0 \22\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=18816, ...}) = 0

mmap(NULL, 20752, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f6e13b70000

mmap(0x7f6e13b71000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7f6e13b71000

mmap(0x7f6e13b73000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f6e13b73000

mmap(0x7f6e13b74000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f6e13b74000

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

читать из файлового дескриптора по заданному смещению

ssize\_t pread(int fd, void \*buf, size\_t count, off\_t offset);

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

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) = 0x7f6e1397e000

mprotect(0x7f6e139a3000, 1847296, PROT\_NONE) = 0

поставить защиту на область памяти

int mprotect(const void \*addr, size\_t len, int prot);

mprotect() изменяет защиту для страниц памяти вызывающего процесса, содержащих любую часть диапазона адресов в интервале [addr, addr+len-1].

PROT\_NONE - the memory cannot be accessed at all.

mmap(0x7f6e139a3000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f6e139a3000

mmap(0x7f6e13b1b000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f6e13b1b000

mmap(0x7f6e13b66000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f6e13b66000

mmap(0x7f6e13b6c000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f6e13b6c000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f6e1397b000

arch\_prctl(ARCH\_SET\_FS, 0x7f6e1397b740) = 0

mprotect(0x7f6e13b66000, 12288, PROT\_READ) = 0

mprotect(0x7f6e13b74000, 4096, PROT\_READ) = 0

mprotect(0x7f6e13ba1000, 16384, PROT\_READ) = 0

mprotect(0x555a4e938000, 16384, PROT\_READ) = 0

mprotect(0x7f6e13be6000, 4096, PROT\_READ) = 0

munmap(0x7f6e13ba8000, 66821) = 0

openat(AT\_FDCWD, "/dev/tty", O\_RDWR|O\_NONBLOCK) = 3

close(3) = 0

brk(NULL) = 0x555a4f414000

brk(0x555a4f435000) = 0x555a4f435000

openat(AT\_FDCWD, "/usr/lib/locale/locale-archive", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=14537584, ...}) = 0

mmap(NULL, 14537584, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f6e12b9d000

close(3) = 0

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/gconv/gconv-modules.cache", O\_RDONLY) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=27002, ...}) = 0

mmap(NULL, 27002, PROT\_READ, MAP\_SHARED, 3, 0) = 0x7f6e13bb2000

close(3) = 0

getuid() = 1000

getuid возвращает фактический идентификатор ID пользователя в текущем процессе.

getgid() = 1000

geteuid() = 1000

geteuid возвращает эффективный идентификатор ID пользователя в текущем процессе. Фактический ID соответствует ID пользователя, который вызвал процесс. Эффективный ID соответствует установленному setuid биту на исполняемом файле.

getegid() = 1000

getegid возвращает еффективный идентификатор группы текущего процесса.

rt\_sigprocmask(SIG\_BLOCK, NULL, [], 8) = 0

sigprocmask() используется для получения и/или изменения маски сигнала вызывающего потока.

SIG\_BLOCK

Набор заблокированных сигналов является объединением текущего набора и заданного аргумента.

Если set равен NULL, то маска сигнала не изменяется.

ioctl(-1, TIOCGPGRP, 0x7fff5fbe3544) = -1 EBADF (Bad file descriptor)

Функция ioctl манипулирует базовыми параметрами устройств, представленных в виде специальных файлов.

В качестве аргумента d должен быть указан открытый файловый дескриптор.

sysinfo({uptime=76630, loads=[189632, 141792, 109152], totalram=2083635200, freeram=85999616, sharedram=83279872, bufferram=14331904, totalswap=968105984, freeswap=231399424, procs=585, totalhigh=0, freehigh=0, mem\_unit=1}) = 0

struct sysinfo {

long uptime; /\* Количество секунд, прошедшее с загрузки системы \*/

unsigned long loads[3]; /\* средняя одно-, пяти-, и пятнадцатиминутная загруженность системы \*/

unsigned long totalram; /\* Общий объем доступной оперативной памяти \*/

unsigned long freeram; /\* Объем свободной памяти \*/

unsigned long sharedram; /\* Объем разделяемой памяти \*/

unsigned long bufferram; /\* Объем памяти, использованной под буферы \*/

unsigned long totalswap; /\* Общий объем области подкачки \*/

unsigned long freeswap; /\* Объем свободного пространства в области подкачки \*/

unsigned short procs; /\* Текущее количество процессов \*/

char \_f[22]; /\* Дополнение структуры до 64-х байтов \*/

};

rt\_sigaction(SIGCHLD, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

int sigaction(int signum, const struct sigaction \*act, struct sigaction \*oldact);

sigaction() используется для изменения выполняемого процессом действия при получении определённого сигнала

Поле sa\_restorer не предназначено для использования в приложении (в POSIX sa\_restorer не определёно). Предназначение этого поля немного описано в sigreturn(2).

sa\_handler указывает действие, которое должно быть связано с signum; может принимать значение SIG\_DFL в качестве действия по умолчанию, SIG\_IGN, чтобы игнорировать этот сигнал, или принимать указатель на функцию управления сигналом. Данная функция получает номер сигнала в качестве своего единственного аргумента.

Если в sa\_flags указан SA\_SIGINFO, то sa\_sigaction (вместо sa\_handler) задаёт функцию обработки сигнала signum. В первом аргументе функция принимает номер сигнала, во втором — указатель на siginfo\_t и в третьем — указатель на ucontext\_t (приведённый к void \*) (вообще, функция-обработчик не используется третий аргумент. Подробней о ucontext\_t смотрите в getcontext(3)).

В sa\_mask задаётся маска сигналов, которые должны блокироваться (т.е. добавляется к маске сигналов нити, в которой вызывается обработчик сигнала) при выполнении обработчика сигнала. Также будет блокироваться и сигнал, вызвавший запуск обработчика, если только не был использован флаг SA\_NODEFER.

В flag указывается набор флагов, которые изменяют поведение сигнала.

rt\_sigaction(SIGCHLD, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigaction(SIGQUIT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

rt\_sigaction(SIGQUIT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigaction(SIGTSTP, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

rt\_sigaction(SIGTSTP, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigaction(SIGTTIN, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

rt\_sigaction(SIGTTIN, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigaction(SIGTTOU, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=0}, 8) = 0

rt\_sigaction(SIGTTOU, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, NULL, [], 8) = 0

rt\_sigaction(SIGQUIT, {sa\_handler=SIG\_IGN, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

uname({sysname="Linux", nodename="sophie-VirtualBox", ...}) = 0

uname возвращает информацию о системе в структуру с адресом buf

stat("/home/sophie/os/os2", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat(".", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat("/home", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}) = 0

stat("/home/sophie", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}) = 0

stat("/home/sophie/os", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat("/home/sophie/os/os2", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat("/home/sophie/os", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

getpid() = 36900

getppid() = 36897

getpid() = 36900

getpgrp() = 36897

ioctl(2, TIOCGPGRP, [36897]) = 0

rt\_sigaction(SIGCHLD, {sa\_handler=0x555a4e87eaa0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART, sa\_restorer=0x7f6e139c4210}, 8) = 0

prlimit64(0, RLIMIT\_NPROC, NULL, {rlim\_cur=7720, rlim\_max=7720}) = 0

int prlimit(pid\_t pid, int resource, const struct rlimit \*new\_limit,

struct rlimit \*old\_limit);

int prlimit(pid\_t pid, int resource, const struct rlimit \*new\_limit,

struct rlimit \*old\_limit);

rt\_sigprocmask(SIG\_BLOCK, NULL, [], 8) = 0

openat(AT\_FDCWD, "./run2.sh", O\_RDONLY) = 3

stat("./run2.sh", {st\_mode=S\_IFREG|0764, st\_size=155, ...}) = 0

ioctl(3, TCGETS, 0x7fff5fbe34d0) = -1 ENOTTY (Inappropriate ioctl for device)

lseek(3, 0, SEEK\_CUR) = 0

read(3, "#!/bin/bash\n\nset -e # exit on e"..., 80) = 80

lseek(3, 0, SEEK\_SET) = 0

lseek - установить смещение для позиционирования операций чтения/записи

off\_t lseek(int fildes, off\_t offset, int whence);

Функция lseek устанавливает смещение для файлового дескриптора fildes в значение аргумента offset в соответствии с директивой whence которая может принимать одно из следующих значений:

SEEK\_SET

Смещение устанавливается в offset байт (от начала файла -- прим. пер.).

prlimit64(0, RLIMIT\_NOFILE, NULL, {rlim\_cur=1024, rlim\_max=1024\*1024}) = 0

fcntl(255, F\_GETFD) = -1 EBADF (Bad file descriptor)

int fcntl(int fd, int cmd, long arg);

fcntl выполняет одну из различных дополнительных операций над файловым дескриптором fd. Эта операция определяется содержимым аргумента cmd.

F\_GETFD

Читает флаг close-on-exec. Если бит FD\_CLOEXEC установлен в 0, то файл будет оставлен открытым при вызове exec, в противном случае он будет закрыт.

dup2(3, 255) = 255

dup, dup2 - создать дубликат файлового дескриптора

close(3) = 0

fcntl(255, F\_SETFD, FD\_CLOEXEC) = 0

fcntl(255, F\_GETFL) = 0x8000 (flags O\_RDONLY|O\_LARGEFILE)

fstat(255, {st\_mode=S\_IFREG|0764, st\_size=155, ...}) = 0

lseek(255, 0, SEEK\_CUR) = 0

read(255, "#!/bin/bash\n\nset -e # exit on e"..., 155) = 155

fstat(2, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

write(2, "+ gcc main.c -o main\n", 21+ gcc main.c -o main

) = 21

stat(".", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat("/usr/local/sbin/gcc", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/local/bin/gcc", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/sbin/gcc", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/gcc", X\_OK) = 0

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/gcc", R\_OK) = 0

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/gcc", X\_OK) = 0

stat("/usr/bin/gcc", {st\_mode=S\_IFREG|0755, st\_size=1154192, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/gcc", R\_OK) = 0

rt\_sigprocmask(SIG\_BLOCK, [INT CHLD], [], 8) = 0

lseek(255, -65, SEEK\_CUR) = 90

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f6e1397ba10) = 36901

int clone(int (\*fn)(void \*), void \*child\_stack, int flags, void \*arg);

clone создаёт новый процесс

Аргумент fn является указателем на функцию, которая вызывается процессом-потомком в начале своего выполнения. Аргумент arg передаётся этой fn функции.

Аргумент child\_stack задаёт положение стека, используемого процессом-потомком.

Аргумент flags состоит из одного или более данных ниже битовых флагов, которые складываются по правилам битового сложения (OR). Флаги задают порядок процессов и то, что разделяется между вызывающим процессом и процессом-потомком:

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 36901

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

ioctl(2, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

stat("/home/sophie/.terminfo", 0x555a4f427e10) = -1 ENOENT (No such file or directory)

stat("/etc/terminfo", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}) = 0

stat("/lib/terminfo", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}) = 0

stat("/usr/share/terminfo", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}) = 0

access("/etc/terminfo/x/xterm-256color", R\_OK) = -1 ENOENT (No such file or directory)

access("/lib/terminfo/x/xterm-256color", R\_OK) = 0

openat(AT\_FDCWD, "/lib/terminfo/x/xterm-256color", O\_RDONLY) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=3503, ...}) = 0

read(3, "\36\2%\0&\0\17\0\235\1\356\5xterm-256color|xterm"..., 32768) = 3503

read(3, "", 28672) = 0

close(3) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

ioctl(1, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

ioctl(0, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

brk(0x555a4f456000) = 0x555a4f456000

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=36901, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, 0x7fff5fbe2c10, WNOHANG, NULL) = -1 ECHILD (No child processes)

rt\_sigreturn({mask=[]}) = 0

read(255, "gcc child.c -o child\nprintf 'res"..., 155) = 65

write(2, "+ gcc child.c -o child\n", 23+ gcc child.c -o child

) = 23

rt\_sigprocmask(SIG\_BLOCK, [INT CHLD], [], 8) = 0

lseek(255, -44, SEEK\_CUR) = 111

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f6e1397ba10) = 36909

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 36909

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

ioctl(2, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=36909, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, 0x7fff5fbe2c10, WNOHANG, NULL) = -1 ECHILD (No child processes)

rt\_sigreturn({mask=[]}) = 0

read(255, "printf 'result\\n2 3\\n' | ./main\n"..., 155) = 44

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

pipe([3, 4]) = 0

rt\_sigprocmask(SIG\_BLOCK, [INT CHLD], [CHLD], 8) = 0

lseek(255, -12, SEEK\_CUR) = 143

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f6e1397ba10) = 36915

rt\_sigprocmask(SIG\_SETMASK, [CHLD], NULL, 8) = 0

close(4) = 0

close(4) = -1 EBADF (Bad file descriptor)

rt\_sigprocmask(SIG\_BLOCK, [INT CHLD], [CHLD], 8) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f6e1397ba10) = 36916

rt\_sigprocmask(SIG\_SETMASK, [CHLD], NULL, 8) = 0

close(3) = 0

+ ./main

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [CHLD], 8) = 0

rt\_sigprocmask(SIG\_SETMASK, [CHLD], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [CHLD], 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

wait4(-1, + printf 'result\n2 3\n'

[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 36915

pid\_t wait4(pid\_t pid, int \*status, int options, struct rusage \*rusage);

Функция wait4 приостанавливает исполнение текущего процесса до того, как свою работу завершит дочерний процесс с номером pid, или этот процесс не получит сигнал, прекращающий его работу, или не будет произведен вызов обработчика прерывания.

wait4(-1, child: started

number=2.000000

number=3.000000

number=inf

sum=5.000000

[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 36916

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

ioctl(2, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

rt\_sigprocmask(SIG\_SETMASK, [CHLD], NULL, 8) = 0

close(3) = -1 EBADF (Bad file descriptor)

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=36915, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, 0x7fff5fbe2b90, WNOHANG, NULL) = -1 ECHILD (No child processes)

rt\_sigreturn({mask=[]}) = 0

read(255, "cat result\n\n", 155) = 12

write(2, "+ cat result\n", 13+ cat result

) = 13

stat(".", {st\_mode=S\_IFDIR|0775, st\_size=4096, ...}) = 0

stat("/usr/local/sbin/cat", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/local/bin/cat", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/sbin/cat", 0x7fff5fbe3170) = -1 ENOENT (No such file or directory)

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/cat", X\_OK) = 0

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/cat", R\_OK) = 0

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/cat", X\_OK) = 0

stat("/usr/bin/cat", {st\_mode=S\_IFREG|0755, st\_size=43416, ...}) = 0

geteuid() = 1000

getegid() = 1000

getuid() = 1000

getgid() = 1000

access("/usr/bin/cat", R\_OK) = 0

rt\_sigprocmask(SIG\_BLOCK, [INT CHLD], [], 8) = 0

lseek(255, -1, SEEK\_CUR) = 154

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f6e1397ba10) = 36918

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigaction(SIGINT, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

wait4(-1, 5.000000

[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 36918

rt\_sigaction(SIGINT, {sa\_handler=SIG\_DFL, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, {sa\_handler=0x555a4e87b480, sa\_mask=[], sa\_flags=SA\_RESTORER, sa\_restorer=0x7f6e139c4210}, 8) = 0

ioctl(2, TIOCGWINSZ, {ws\_row=48, ws\_col=203, ws\_xpixel=0, ws\_ypixel=0}) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=36918, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, 0x7fff5fbe2c10, WNOHANG, NULL) = -1 ECHILD (No child processes)

rt\_sigreturn({mask=[]}) = 0

read(255, "\n", 155) = 1

read(255, "", 155) = 0

rt\_sigprocmask(SIG\_BLOCK, [CHLD], [], 8) = 0

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

exit\_group(0) = ?

+++ exited with 0 +++

Мы видим список системных вызовов, которые делает утилита.

Вывод происходит в виде:

имя\_системного\_вызова (параметры) = результат сообщение

Системные вызовы:

fork(...) - создание нового дочернего процесса;  
execve(...) - запускает программу  
access(...) - проверяет права доступа к файлу, возвращает -1 как код ошибки  
openat(...) - открывает файл, имеет в качестве возвращаемого значения файловый дескриптор  
fstat(...) - собирает информацию из файла  
mmap(...) - отображает файл на память  
mprotect(...) - контролирует доступ к памяти  
close(...) - закрывает файловый дескриптор  
read(...) - считывает из файлового дескриптора  
arch\_prctl(...) - устанавливает специфичное для архитектуры значение ядра  
munmap(...) - освобождает память, отведенную для отображения файла  
write(...) - пишет в консоль  
ftruncate(...) - увеличивает размер файла до n байтов  
clone(...) - создает новый процесс  
unlink(...) - удаляет файл  
lseek(...) - устанавливает смещение для позиционирования операций чтения/записи

**Выводы**В данной лабораторной работе я научилась пользоваться strace и с помощью его отслеживать системные вызовы. Также изучила сами системные вызовы и для чего они используются.