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: Старцев Иван Романович
Группа: М8О-201Б-21
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/IvanTvardovsky/OS-labs

Постановка задачи

Подробно рассказать о каждом системном вызове из утилиты strace на примере лабораторной работы №4.

Описание работы strace

execve — открывает файл на исполнение.

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

execve() выполняет программу, задаваемую аргументом *filename*.

argv — это массив строковых параметров, передаваемых новой программе. По соглашению, в первой строке должно содержаться имя файла, относящееся к запускаемой программе.

envp — это массив строк в формате ключ=значение, которые передаются новой программе в качестве окружения (environment). Оба массива argv и envp завершаются указателем null.

brk — изменяет расположение маркера окончания неинициализированных данных, который определяет конец сегмента данных процесса.

int brk(void *addr);

brk() устанавливает конец сегмента данных в значение, указанное в аргументе *addr*, если это значение является приемлемым, система имеет достаточно памяти и процесс не достиг максимально возможного размера своего сегмента данных

arch_prtcl — устанавливает состояние процесса или потока, зависящее от архитектуры.

int arch_prctl(int code, unsigned long addr);
int arch_prctl(int code, unsigned long *addr);

Функция arch_prctl() задаёт состояние процесса или нити, зависящие от архитектуры. В аргументе *code* выбирается подфункция и ей передаётся значение *addr*; параметр *addr* рассматривается либо как unsigned long при операциях «установки», либо как unsigned long * при операциях «получения» значения.

access - для проверки существования файла

openat — открывает файл в определенной директории.

newfstatat — возвращает информацию о файле в буфер.

close — закрывает файловый дескриптор.

mmap — создает новое отображение памяти в адресном пространстве процесса.

ftruncate — устанавливает файлу необходимый размер.

munmap — удаляет отображение.

mprotect − контролирует доступ к области памяти.

set_robust_list - запрашивает ядро записать начало списка надёжных фьютексов, принадлежащего вызывающей нити

rt_sigaction - получает и изменяет обработчик сигнала.

rt_sigprocmask - используется для проверки или настройки сигнальной маски текущего процесса.

set_tid_address - устанавливает у вызывающей нити
значение clear_child_tid равным tidptr (В ядре для каждой нити хранится два атрибута (адреса): set_child_tid и clear_child_tid. Их значение по умолчанию равно NULL)

futex - предоставляет программам метод для ожидания пока определённое условие не станет истинным

fstatat — требует права выполнения (поиска) на все каталоги, указанные в полном имени файла pathname. (опрашиваемый файл задаётся в виде файлового дескриптора fd.)

statfs - возвращает информацию о смонтированной файловой системе

clone - создаёт новый процесс подобно fork

clock_nanosleep - позволяет вызывающей нити приостановить работу на некоторое время с наносекундной точностью

lseek - позволяет задавать смещение, которое будет находиться за существующим концом файла (но это не изменяет размер файла)

exit_group - завершает исполнение всех потоков процесса.

Демонстрация работы strace

```
tvard@tvard-HVY-WXX9:~/os/OS-labs/lab4$ strace -f ./lab4
execve("./lab4", ["./lab4"], 0x7ffdd6445d68 /* 72 vars */) = 0
brk(NULL)                               = 0x558c38da4000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdabefae70) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fc001cf8000
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=59823, ...}, AT_EMPTY_PATH) =
0
mmap(NULL, 59823, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc001ce9000
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\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) =
0x7fc001ac1000
mmap(0x7fc001ae9000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fc001ae9000
mmap(0x7fc001c7e000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1bd000) = 0x7fc001c7e000
mmap(0x7fc001cd6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fc001cd6000
mmap(0x7fc001cdc000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fc001cdc000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fc001abe000
arch_prctl(ARCH_SET_FS, 0x7fc001abe740) = 0
set_tid_address(0x7fc001abea10)         = 6804
set_robust_list(0x7fc001abea20, 24)     = 0
rseq(0x7fc001abf0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fc001cd6000, 16384, PROT_READ) = 0
mprotect(0x558c383ef000, 4096, PROT_READ) = 0
mprotect(0x7fc001d32000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fc001ce9000, 59823)           = 0
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
getrandom("\x46\xc5\xc9\x64\x03\x90\x3c\x51", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x558c38da4000
```

```
brk(0x558c38dc5000)                     = 0x558c38dc5000
read(0, test.txt
"test.txt\n", 1024)             = 9
read(0, output.txt
"output.txt\n", 1024)           = 11
openat(AT_FDCWD, "test.txt", O_RDONLY)  = 3
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLDstrace: Process 6839 attached
, child_tidptr=0x7fc001abea10) = 6839
[pid  6804] wait4(6839,  <unfinished ...>
[pid  6839] set_robust_list(0x7fc001abea20, 24) = 0
[pid  6839] dup2(3, 0)                  = 0
[pid  6839] execve("/home/tvard/os/OS-labs/lab4/child",
["/home/tvard/os/OS-labs/lab4/chil"..., "test.txt"], 0x7ffdabefb048 /* 72 vars */) = 0
[pid  6839] brk(NULL)                   = 0x56263f5a6000
[pid  6839] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffeb4f54830) = -1 EINVAL (Invalid
argument)
[pid  6839] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f45a51b8000
[pid  6839] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or
directory)
[pid  6839] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 4
[pid  6839] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=59823, ...},
AT_EMPTY_PATH) = 0
[pid  6839] mmap(NULL, 59823, PROT_READ, MAP_PRIVATE, 4, 0) =
0x7f45a51a9000
[pid  6839] close(4)                    = 0
[pid  6839] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|
O_CLOEXEC) = 4
[pid  6839] read(4, "\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\0"..., 832) = 832
[pid  6839] 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  6839] pread64(4, "\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
[pid  6839] pread64(4, "\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
[pid  6839] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid  6839] 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  6839] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4,
0) = 0x7f45a4f81000
[pid  6839] mmap(0x7f45a4fa9000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000) = 0x7f45a4fa9000
[pid  6839] mmap(0x7f45a513e000, 360448, PROT_READ, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 4, 0x1bd000) = 0x7f45a513e000
[pid  6839] mmap(0x7f45a5196000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x214000) = 0x7f45a5196000
[pid  6839] mmap(0x7f45a519c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f45a519c000
[pid  6839] close(4)                    = 0
[pid  6839] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f45a4f7e000
```

```
[pid  6839] arch_prctl(ARCH_SET_FS, 0x7f45a4f7e740) = 0
[pid  6839] set_tid_address(0x7f45a4f7ea10) = 6839
[pid  6839] set_robust_list(0x7f45a4f7ea20, 24) = 0
[pid  6839] rseq(0x7f45a4f7f0e0, 0x20, 0, 0x53053053) = 0
[pid  6839] mprotect(0x7f45a5196000, 16384, PROT_READ) = 0
[pid  6839] mprotect(0x56263eb8f000, 4096, PROT_READ) = 0
[pid  6839] mprotect(0x7f45a51f2000, 8192, PROT_READ) = 0
[pid  6839] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid  6839] munmap(0x7f45a51a9000, 59823) = 0
[pid  6839] getrandom("\x44\xc6\x9a\x70\xc6\x70\x16\x92", 8, GRND_NONBLOCK)
= 8
[pid  6839] brk(NULL)                   = 0x56263f5a6000
[pid  6839] brk(0x56263f5c7000)         = 0x56263f5c7000
[pid  6839] openat(AT_FDCWD, "test.txt", O_RDONLY) = 4
[pid  6839] openat(AT_FDCWD, "/dev/shm/shm", O_RDWR|O_CREAT|O_NOFOLLOW|
O_CLOEXEC, 0777) = 5
[pid  6839] ftruncate(5, 4096)          = 0
[pid  6839] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x7f45a51f1000
[pid  6839] newfstatat(0, "", {st_mode=S_IFREG|0664, st_size=75, ...},
AT_EMPTY_PATH) = 0
[pid  6839] read(0, "8.0 2.0 -4.0 -1.0\n0.0 3.2 2.09\n-"..., 4096) = 75
[pid  6839] close(5)                    = 0
[pid  6839] close(5)                    = -1 EBADF (Bad file descriptor)
[pid  6839] close(5)                    = -1 EBADF (Bad file descriptor)
[pid  6839] close(5)                    = -1 EBADF (Bad file descriptor)
[pid  6839] read(0, "", 4096)           = 0
[pid  6839] close(5)                    = -1 EBADF (Bad file descriptor)
[pid  6839] newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88,
0), ...}, AT_EMPTY_PATH) = 0
[pid  6839] write(1, "1 0 -0.1 9.75 1 \n", 171 0 -0.1 9.75 1
) = 17
[pid  6839] close(4)                    = 0
[pid  6839] exit_group(0)               = ?
[pid  6839] +++ exited with 0 +++
<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 6839
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=6839, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
openat(AT_FDCWD, "/dev/shm/shm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0777) = 5
mmap(NULL, 4096, PROT_READ, MAP_SHARED, 5, 0) = 0x7fc001d31000
newfstatat(4, "", {st_mode=S_IFREG|0664, st_size=0, ...}, AT_EMPTY_PATH) = 0
munmap(0x7fc001d31000, 4096)            = 0
unlink("/dev/shm/shm")                  = 0
close(5)                                = 0
write(4, "1 0 -0.1 9.75 1 ", 16)        = 16
close(4)                                = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Проделав лабораторную работу, я приобрёл навыки, необходимые для работы с strace, а также изучил системные вызовы.