Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу

«Операционные системы»

Диагностика программного обеспечения

Студент: Семенов Илья Михайлович
Группа: М80 – 206Б-18
Вариант: 1
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Постановка задачи

Цель работы — приобретение практических навыков диагностики работы программного обеспечения.

Strace

Strace показывает все системные вызовы программы, которые она отправляет к системе во время выполнения, а также их параметры и результат выполнения. При необходимости можно подключиться к уже запущенному процессу.

Strace имеет следующие(и не только) ключи

- **-i** — выводить указатель на инструкцию во время выполнения системного вызова
- **-o** — выводить всю информацию о системных вызовах не в стандартный поток ошибок, а в файл
- **-r** — выводить временную метку для каждого системного вызова
- **-T** — выводить длительность выполнения каждого системного вызова
- **-b** — если указанный системный вызов обнаружен, трассировка прекращается
- **-l** — позволяет блокировать реакцию нажатия Ctrl+C и Ctrl+Z
- **-f** — отслеживать дочерние процессы и создаваемые потоки

Общий метод и алгоритм решения

Протестируем для примера лабораторную работу 4.

Код программы

main.c:

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <sys/mman.h>
#include <unistd.h>
#include <fcntl.h>
#include <semaphore.h>
#include <wait.h>
#include <sys/stat.h>
#include <stdbool.h>
#define BUFFER_SIZE 100
int parse_string(char buf[], int size, bool* prev) {
    int temp_size = 0;
    char copy[size];
    bool space = *prev;
```

```
    for (int i = 0; i < size; ++i) {
        if (buf[i] != ' ' || !space) {
            if (buf[i] == ' ') {
                space = true;
            } else {
                space = false;
            }
            copy[temp_size] = buf[i];
            temp_size++;
        }
    }
    for (int i = 0; i < temp_size; ++i) {
        buf[i] = copy[i];
    }
    *prev = space;
    return temp_size;
}
void throw_error(const char* error) {
    printf("%s\n", error);
    exit(1);
}
int main(int argc, char** argv) {
    if (argc < 2) {
        throw_error("No file");
    }
    if (strlen(argv[1]) > 100) {
        throw_error("Filename is too long");
    }
    //создание временного файла для маппинга
    char* tmp_name = strdup("/tmp/tmp_file.XXXXXX");
    int tmp_fd = mkstemp(tmp_name);
    if (tmp_fd == -1) {
        throw_error("Cannot create temp file to map");
    }
    free(tmp_name);
    int file_size = BUFFER_SIZE + 1;
    char file_filler[file_size];
    for (int i = 0; i < file_size; ++i) {
        file_filler[i] = '\0';
    }
    write(tmp_fd, file_filler, file_size);
    //маппинг файла
    unsigned char* map = (unsigned char*)mmap(NULL, file_size, PROT_WRITE |
PROT_READ, MAP_SHARED, tmp_fd, 0);
    if (map == NULL) {
        throw_error("Cant map file");
    }
    //создание семафоров для синхронизации работы
    const char* in_sem_name = "/input_semaphor";
    const char* out_sem_name = "/output_semaphor";
    sem_unlink(in_sem_name);
    sem_unlink(out_sem_name);
    sem_t* in_sem = sem_open(in_sem_name, O_CREAT, 777, 0);
    sem_t* out_sem = sem_open(out_sem_name, O_CREAT, 777, 0);
    if (in_sem == SEM_FAILED || out_sem == SEM_FAILED) {
        throw_error("Cannot create semaphor");
    }
```

```
    strcpy(map, argv[1]);
    map[BUFFER_SIZE] = strlen(argv[1]);
    int pid = fork();
    if (pid == -1) {
        throw_error("Fork failure");
    } else if (pid == 0) { //child
        int output_file = open(argv[1], O_RDWR | O_TRUNC | O_CREAT, S_IREAD |
S_IWRITE);
        if (output_file == -1) {
            map[BUFFER_SIZE] = 101;
            sem_post(out_sem);
            throw_error("Cannot create output file");
        }
        bool space = false;
        sem_post(out_sem);
        while (true) {
            sem_wait(in_sem);
            int new_size = parse_string(map, map[BUFFER_SIZE], &space);
            write(output_file, map, new_size);
            if (map[BUFFER_SIZE] < BUFFER_SIZE){
                sem_post(out_sem);
                break;
            }
            sem_post(out_sem);
        }
        close(output_file);
        exit(0);
    } else { //parent
        sem_wait(out_sem);
        if (map[BUFFER_SIZE] != 101) {
            int read_count = read(STDIN_FILENO, map, BUFFER_SIZE);
            map[BUFFER_SIZE] = read_count;
            sem_post(in_sem);
            while (read_count == BUFFER_SIZE) {
                sem_wait(out_sem);
                read_count = read(STDIN_FILENO, map, BUFFER_SIZE);
                map[BUFFER_SIZE] = read_count;
                sem_post(in_sem);
            }
            int stat_lock;
            wait(&stat_lock);
            if (stat_lock != 0) {
                printf("%s\n", "Child failure");
            }
        } else {
            int stat_lock;
            wait(&stat_lock);
            if (stat_lock != 0) {
                printf("%s\n", "Child failure");
            }
        }
        sem_close(in_sem);
        sem_close(out_sem);
    }
}
```

Демонстрация работы программы

ilya@ilya-lenovo:~/CLionProjects/os_lab_04/src/cmake-build-debug$ strace -T -i ./os_lab_04 output < input_file

[00007f52cc34fe37] execve("./os_lab_04", ["./os_lab_04", "output"], 0x7ffe867f9808 /* 52 vars */) = 0 <0.000740>

[00007f9e4601eec9] brk(NULL) = 0x555cc6d36000 <0.000023>

[00007f9e460127de] access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000037>

[00007f9e4601fe27] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory) <0.000041>

[00007f9e4601fcdd] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3 <0.000039>

[00007f9e4601fc43] fstat(3, {st_mode=S_IFREG|0644, st_size=152516, ...}) = 0 <0.000025>

[00007f9e4601ff43] mmap(NULL, 152516, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9e46204000 <0.000034>

[00007f9e4601fed7] close(3) = 0 <0.000021>

[00007f9e4601b139] access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000030>

[00007f9e4601fcdd] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3 <0.000038>

[00007f9e4601fda4] 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 <0.000027>

[00007f9e4601fc43] fstat(3, {st_mode=S_IFREG|0755, st_size=144976, ...}) = 0 <0.000046>

[00007f9e4601ff43] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e46202000 <0.000060>

[00007f9e4601ff43] mmap(NULL, 2221184, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e45de4000 <0.000070>

[00007f9e4601fff7] mprotect(0x7f9e45dfe000, 2093056, PROT_NONE) = 0 <0.000058>

[00007f9e4601ff43] mmap(0x7f9e45ffd000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19000) = 0x7f9e45ffd000 <0.000073>

[00007f9e4601ff43] mmap(0x7f9e45fff000, 13440, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9e45fff000 <0.000054>

[00007f9e4601fed7] close(3) = 0 <0.000108>

[00007f9e4601b139] access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000139>

[00007f9e4601fcdd] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3 <0.000090>

[00007f9e4601fda4] 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 <0.000078>

[00007f9e4601fc43] fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0 <0.000101>

[00007f9e4601ff43] mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e459f3000 <0.000074>

[00007f9e4601fff7] mprotect(0x7f9e45bda000, 2097152, PROT_NONE) = 0 <0.000082>

[00007f9e4601ff43] mmap(0x7f9e45dda000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f9e45dda000 <0.000069>

[00007f9e4601ff43] mmap(0x7f9e45de0000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9e45de0000 <0.000140>

[00007f9e4601fed7] close(3) = 0 <0.000048>

[00007f9e4601ff43] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e461ff000 <0.000064>

[00007f9e46004024] arch_prctl(ARCH_SET_FS, 0x7f9e461ff740) = 0 <0.000047>

[00007f9e4601fff7] mprotect(0x7f9e45dda000, 16384, PROT_READ) = 0 <0.000057>

[00007f9e4601fff7] mprotect(0x7f9e45ffd000, 4096, PROT_READ) = 0 <0.000057>

[00007f9e4601fff7] mprotect(0x555cc67ca000, 4096, PROT_READ) = 0 <0.000039>

[00007f9e4601fff7] mprotect(0x7f9e4622a000, 4096, PROT_READ) = 0 <0.000037>

[00007f9e4601ffd7] munmap(0x7f9e46204000, 152516) = 0 <0.000061>

[00007f9e45de9eb5] set_tid_address(0x7f9e461ffa10) = 5138 <0.000029>

[00007f9e45de9f17] set_robust_list(0x7f9e461ffa20, 24) = 0 <0.000027>

[00007f9e45df695d] rt_sigaction(SIGRTMIN, {sa_handler=0x7f9e45de9cb0, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f9e45df6890}, NULL, 8) = 0 <0.000036>

[00007f9e45df695d] rt_sigaction(SIGRT_1, {sa_handler=0x7f9e45de9d50, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f9e45df6890}, NULL, 8) = 0 <0.000029>

[00007f9e45de9ff3] rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0 <0.000035>

[00007f9e45b08fa0] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0 <0.000033>

[00007f9e45b094b9] brk(NULL)   = 0x555cc6d36000 <0.000045>

[00007f9e45b094b9] brk(0x555cc6d57000) = 0x555cc6d57000 <0.000040>

[00007f9e45ad88e7] getpid()   = 5138 <0.000035>

[00007f9e45b02c8e] openat(AT_FDCWD, "/tmp/tmp_file.mNETrL", O_RDWR|O_CREAT|O_EXCL, 0600) = 3 <0.000198>

[00007f9e45df5281] write(3, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 101) = 101 <0.000109>

[00007f9e45b0ea13] mmap(NULL, 101, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f9e46229000 <0.000056>

[00007f9e45b02977] statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=992842, f_bfree=772475, f_bavail=772475, f_files=992842, f_ffree=992388, f_fsid={val=[0, 0]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV}) = 0 <0.000049>

[00007f9e45df384e] futex(0x7f9e46002370, FUTEX_WAKE_PRIVATE, 2147483647) = 0 <0.000156>

[00007f9e45b04d47] unlink("/dev/shm/sem.input_semaphor") = 0 <0.000059>

[00007f9e45b04d47] unlink("/dev/shm/sem.output_semaphor") = 0 <0.002950>

[00007f9e45df5d2b] openat(AT_FDCWD, "/dev/shm/sem.input_semaphor", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory) <0.000063>

[00007f9e45ad88e7] getpid() = 5138 <0.000029>

[00007f9e45b02815] lstat("/dev/shm/GSKPfj", 0x7ffd1f0de3f0) = -1 ENOENT (No such file or directory) <0.000043>

[00007f9e45df5d2b] openat(AT_FDCWD, "/dev/shm/GSKPfj", O_RDWR|O_CREAT|O_EXCL, 01411) = 4 <0.000050>

[00007f9e45df5281] write(4, "\0\0\0\0\0\0\0\200\0\0\0\236\177\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32 <0.000048>

[00007f9e45b0ea13] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f9e46228000 <0.000038>

[00007f9e45b04c27] link("/dev/shm/GSKPfj", "/dev/shm/sem.input_semaphor") = 0 <0.000058>

[00007f9e45b027c3] fstat(4, {st_mode=S_IFREG|S_ISVTX|0411, st_size=32, ...}) = 0 <0.000027>

[00007f9e45b04d47] unlink("/dev/shm/GSKPfj") = 0 <0.000036>

[00007f9e45df5421] close(4) = 0 <0.000026>

[00007f9e45df5d2b] openat(AT_FDCWD, "/dev/shm/sem.output_semaphor", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory) <0.000044>

[00007f9e45ad88e7] getpid() = 5138 <0.000024>

[00007f9e45b02815] lstat("/dev/shm/YWrX3Q", 0x7ffd1f0de3f0) = -1 ENOENT (No such file or directory) <0.000035>

[00007f9e45df5d2b] openat(AT_FDCWD, "/dev/shm/YWrX3Q", O_RDWR|O_CREAT|O_EXCL, 01411) = 4 <0.000068>

[00007f9e45df5281] write(4, "\0\0\0\0\0\0\0\200\0\0\0\236\177\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32 <0.000060>

[00007f9e45b0ea13] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f9e46227000 <0.000038>

[00007f9e45b04c27] link("/dev/shm/YWrX3Q",
"/dev/shm/sem.output_semaphor") = 0 <0.000045>

[00007f9e45b027c3] fstat(4, {st_mode=S_IFREG|S_ISVTX|0411, st_size=32, ...})
= 0 <0.000026>

[00007f9e45b04d47] unlink("/dev/shm/YWrX3Q") = 0 <0.000034>

[00007f9e45df5421] close(4) = 0 <0.000026>

[00007f9e45ad7b1c] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f9e461ffa10) = 5139 <0.001859>

[00007f9e45df534e] read(0, "Lorem ipsum "..., 100) = 100
<0.000062>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000051>

[00007f9e45df534e] read(0, ", sed do eiusmod tempor incididu"..., 100) = 100
<0.000040>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000096>

[00007f9e45df534e] read(0, " minim veniam, quis nostrud\nexer"..., 100) = 100
<0.000021>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000044>

[00007f9e45df534e] read(0, " "..., 100) = 100 <0.000020>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000029>

[00007f9e45df534e] read(0, " ut aliquip ex ea commod"..., 100) = 100
<0.000017>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000038>

[00007f9e45df534e] read(0, "tate velit esse cillum dolore\nue"..., 100) = 75
<0.000018>

[00007f9e45df4ab4] futex(0x7f9e46228000, FUTEX_WAKE, 1) = 1 <0.000300>

[00007f9e45df4ab4] --- SIGCHLD {si_signo=SIGCHLD,
si_code=CLD_EXITED, si_pid=5139, si_uid=1000, si_status=0, si_utime=0,
si_stime=0} ---

[00007f9e45df616e] wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}],
0, NULL) = 5139 <0.000054>

[00007f9e45b0eab7] munmap(0x7f9e46228000, 32) = 0 <0.000073>

[00007f9e45b0eab7] munmap(0x7f9e46227000, 32) = 0 <0.000060>

[00007f9e45ad7e06] exit_group(0)   = ?

[????????????????] +++ exited with 0 +++

Вывод

В результате данной лабораторной работы я узнал о возможностях утилиты strace, а так же о том, как много информации может дать диагностика программы для разработчика.