Московский Авиационный Институт

Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа № 1
по курсу «Операционные системы»

Студент:	Валов В.В
Группа:	М8О-308Б-18
Вариант:	
Преподаватель:	Миронов Е.С.
Оценка:	
Дата:	

Москва, 2020

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

vadim@vadim:~/Desktop/os/lab4/src$ strace -T -i ./a.out output<input

[00007f759ffbae37] execve("./a.out", ["./a.out", "output"], 0x7ffcb4a62ad8 /* 66 vars */) = 0 <0.000158>

[00007f2980341ec9] brk(NULL) = 0x555df2c78000 <0.000009>

[00007f29803357de] access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000013>

[00007f2980342e27] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory) <0.000011>

[00007f2980342cdd] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3 <0.000014>

[00007f2980342c43] fstat(3, {st_mode=S_IFREG|0644, st_size=126711, ...}) = 0 <0.000008>

[00007f2980342f43] mmap(NULL, 126711, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f298052e000 <0.000011>

[00007f2980342ed7] close(3) = 0 <0.000008>

[00007f298033e139] access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000010>

[00007f2980342cdd] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3 <0.000013>

[00007f2980342da4] 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 <0.000010>

[00007f2980342c43] fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0 <0.000008>

[00007f2980342f43] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f298052c000 <0.000010>

[00007f2980342f43] mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f297ff35000 <0.000012>

[00007f2980342ff7] mprotect(0x7f298011c000, 2097152, PROT_NONE) = 0 <0.000015>

[00007f2980342f43] mmap(0x7f298031c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f298031c000 <0.000015>

[00007f2980342f43] mmap(0x7f2980322000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2980322000 <0.000012>

[00007f2980342ed7] close(3) = 0 <0.000008>

[00007f2980327024] arch_prctl(ARCH_SET_FS, 0x7f298052d500) = 0 <0.000009>

[00007f2980342ff7] mprotect(0x7f298031c000, 16384, PROT_READ) = 0 <0.000014>

[00007f2980342ff7] mprotect(0x555df0c9b000, 4096, PROT_READ) = 0 <0.000011>

[00007f2980342ff7] mprotect(0x7f298054d000, 4096, PROT_READ) = 0 <0.000013>

[00007f2980342fd7] munmap(0x7f298052e000, 126711) = 0 <0.000027>

[00007f29800447c3] fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0 <0.000010>

[00007f298004b4b9] brk(NULL) = 0x555df2c78000 <0.000010>

[00007f298004b4b9] brk(0x555df2c99000) = 0x555df2c99000 <0.000010>

[00007f2980045154] write(1, "Enter name of out file\n", 23Enter name of out file

) = 23 <0.000026>

[00007f29800447c3] fstat(0, {st_mode=S_IFREG|0664, st_size=12, ...}) = 0 <0.000010>

[00007f2980045081] read(0, "out\nmem \n5\n5", 4096) = 12 <0.000020>

[00007f2980045154] write(1, "Enter name of memory file\n", 26Enter name of memory file

) = 26 <0.000011>

[00007f2980045154] write(1, "Enter n\n", 8Enter n

) = 8 <0.000011>

[00007f2980045154] write(1, "Enter m\n", 8Enter m

) = 8 <0.000010>

[00007f2980045081] read(0, "", 4096) = 0 <0.000009>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_RDWR|O_CREAT|O_APPEND, 0600) = 3 <0.000037>

[00007f298004dc97] ftruncate(3, 100)   = 0 <0.000017>

[00007f2980050a13] mmap(NULL, 100, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f298054c000 <0.000012>

[00007f2980045a07] pipe([4, 5])   = 0 <0.000013>

[00007f2980019b1c] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f298052d7d0) = 3778 <0.000053>

[00007f2980045154] write(1, "Enter number\n", 13Enter number

) = 13 <0.000015>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_WRONLY) = 6 <0.000025>

Child

[00007f29800458d4] close(6)   = 0 <0.000013>

[00007f29800458d4] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3778, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[00007f2980045154] write(1, "End Enter\n", 10End Enter

) = 10 <0.000015>

[00007f2980045a07] pipe([6, 7])   = 0 <0.000012>

[00007f2980019b1c] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f298052d7d0) = 3780 <0.000056>

[00007f2980045154] write(1, "Enter number\n", 13Child

Enter number

) = 13 <0.000015>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_WRONLY) = 8 <0.000040>

[00007f29800458d4] close(8)   = 0 <0.000036>

[00007f29800458d4] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3780, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[00007f2980045154] write(1, "End Enter\n", 10End Enter

) = 10 <0.000012>

[00007f2980045a07] pipe([8, 9])   = 0 <0.000012>

[00007f2980019b1c] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f298052d7d0) = 3781 <0.000057>

Child

[00007f2980045154] write(1, "Enter number\n", 13Enter number

) = 13 <0.000019>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_WRONLY) = 10 <0.000023>

[00007f29800458d4] close(10)   = 0 <0.000099>

[00007f29800458d4] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3781, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[00007f2980045154] write(1, "End Enter\n", 10End Enter

) = 10 <0.000017>

[00007f2980045a07] pipe([10, 11])   = 0 <0.000014>

[00007f2980019b1c] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f298052d7d0) = 3782 <0.000060>

Child

[00007f2980045154] write(1, "Enter number\n", 13Enter number

) = 13 <0.000018>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_WRONLY) = 12 <0.000072>

[00007f29800458d4] close(12)   = 0 <0.000031>

[00007f29800458d4] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3782, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[00007f2980045154] write(1, "End Enter\n", 10End Enter

) = 10 <0.000016>

[00007f2980045a07] pipe([12, 13])   = 0 <0.000015>

[00007f2980019b1c] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f298052d7d0) = 3783 <0.000044>

[00007f2980045154] write(1, "Enter number\n", 13Child

Enter number

) = 13 <0.000014>

[00007f2980044c8e] openat(AT_FDCWD, "mem", O_WRONLY) = 14 <0.000015>

[00007f29800458d4] close(14) = 0 <0.000011>

[00007f2980045154] write(1, "End Enter\n", 10End Enter

) = 10 <0.000013>

[00007f2980045154] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3783, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[00007f2980019e06] 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.