****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: Черненко Илья Денисович

Группа: М80 – 206Б-18

Преподаватель: Соколов А.А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения об утилите strace
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Примеры работы
6. Вывод

**Постановка задачи**

Цель работы: приобретение практических навыков диагностики работы программного обеспечения.

**Общие сведения об утилите strace**

*strace* – это утилита Linux, позволяющая отследить выполнение системных вызовов и сигналов к ядру системы. Она показывает все системные вызовы, которые отправляет программа во время выполнения, их параметры и результат выполнения.

Некоторые ключи strace:

* **-i** — вывод указателя на инструкцию во время выполнения системного вызова;
* **-x** — вывод всех не ASCI-строки в шестнадцатеричном виде;
* **-с** — подсчитывать количество ошибок, вызовов и время выполнения для каждого системного вызова;
* **-o** — вывод всей информации о системных вызовах не в стандартный поток ошибок, а в файл;
* **-T** — вывод длительности выполнения системных вызовов;
* **-l** — блокировка нажатия Ctrl+C и Ctrl+Z;
* **-f** — отслеживание дочерних процессов и потоков.

**Общий метод и алгоритм решения**

Для реализации поставленной задачи необходимо:

1. Провести диагностику лабораторной работы номер 4.

**Основные файлы программы**

**main.cpp:**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <semaphore.h>

#include <pthread.h>

#include <unistd.h>

#include "c\_queue.h"

typedef struct ans ans;

struct ans {

int cmd;

int val;

char path[32];

};

typedef struct node node,\*pnode;

struct node {

pnode s;

pnode b;

int val;

bool is\_root;

};

pnode node\_create(int val) {

pnode new\_node = (pnode)malloc(sizeof(node));

if (new\_node) {

new\_node -> val = val;

new\_node -> s = NULL;

new\_node -> b = NULL;

new\_node -> is\_root = false;

}

return new\_node;

}

pnode\* search(pnode\* t, queue \*path) {

if (!(\*t) && !q\_is\_empty(path)) {

return NULL;

}

if (!q\_is\_empty(path)) {

char c = q\_front(path);

pop(path);

if (c == 's') {

return search(&(\*t) -> s, path);

} else if (c == 'b') {

return search(&(\*t) -> b, path);

}

return NULL;

}

return t;

}

bool add(pnode\* t, int val, queue \*path) {

if (!(\*t) && q\_is\_empty(path)) {

(\*t) = node\_create(val);

return true;

}

pnode\* pr = search(t, path);

if (!pr) {

return false;

}

pnode new\_node = node\_create(val);

if (!new\_node) {

return false;

}

new\_node -> b = (\*pr);

(\*pr) = new\_node;

return true;

}

void rmv(pnode\* t) {

while((\*t) -> s != NULL){

rmv(&((\*t) -> s));

}

pnode tmp = \*t;

\*t = (\*t) -> b;

free(tmp);

}

bool valid\_numb(char\* numb) {

if (numb == NULL) {

return false;

}

bool flag = true;

int i = 0;

if (numb[i] != '-' && !(numb[i] >= '0' && numb[i] <= '9')) {

flag = false;

}

i++;

while (i < 11) {

if (numb[i] == '\0') {

break;

}

if (!(numb[i] >= '0' && numb[i] <= '9')) {

flag = false;

break;

}

i++;

}

return flag;

}

bool valid\_path(char\* path) {

if (path == NULL) {

return false;

}

if (path[0] == '@' && path[1] == '\0') {

return true;

}

for (int i = 0; i < 32; i++) {

if (path[i] == '\0') {

break;

} else if (path[i] != 's' && path[i] != 'b') {

return false;

}

}

return true;

}

ans\* parser(char\* cmd) {

ans\* parsed = (ans\*)malloc(sizeof(ans));

char\* pch = strtok(cmd," \n");

while (pch != NULL) {

if (strcmp(pch, "prt") == 0) {

parsed->cmd = 0;

break;

} else if (strcmp(pch, "rmv") == 0) {

pch = strtok(NULL, " \n");

if (valid\_path(pch)) {

parsed->cmd = 1;

strcpy(parsed->path, pch);

if (parsed->path[0] == 'b') {

parsed->cmd = -1;

}

break;

} else {

parsed->cmd = -1;

break;

}

} else if (strcmp(pch, "add") == 0) {

pch = strtok(NULL, " \n");

if (valid\_path(pch)) {

strcpy(parsed->path, pch);

pch = strtok(NULL, " \n");

if (parsed->path[0] == 'b') {

parsed->cmd = -1;

break;

}

if (valid\_numb(pch)) {

parsed->cmd = 2;

parsed->val = atoi(pch);

break;

} else {

parsed->cmd = -2;

break;

}

} else {

parsed->cmd = -1;

break;

}

} else if (strcmp(pch, "ext") == 0) {

parsed->cmd = 3;

break;

} else {

parsed->cmd = -777;

break;

}

}

return parsed;

}

void tree\_print(pnode t, int depth) {

if (t) {

for (int i = 0; i < depth; i++) {

write(1, "\t", 1);

}

char numb[11] = {'\0'};

sprintf(numb, "%d", t->val);

int i = 0;

while (numb[i] != '\0') {

i++;

}

write(1, numb, i);

write(1,"\n", 1);

tree\_print(t -> s, depth + 1);

tree\_print(t -> b, depth);

}

}

int create\_tmp() {

char \*fn = strdup("/tmp/tmpf.XXXXXX");

int fd = mkstemp(fn);

unlink(fn);

free(fn);

write(fd, " ", 100);

return fd;

}

int main(int argc, char\* argv[]) {

setvbuf(stdout, (char \*) NULL, \_IONBF, 0);

pnode test = NULL;

char cmd[100] = {'\0'};

ans \*parsed = (ans \*) malloc(sizeof(ans));

int fd = create\_tmp();

lseek(fd, 100, SEEK\_END);

write(fd, "", 1);

struct stat sb;

if (fstat(fd, &sb) == -1) {

perror("can't get file size\n");

}

int fsize = sb.st\_size;

char\* f\_in\_m = mmap(NULL, fsize, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

sem\_t\* sem\_calc = sem\_open("/calc", O\_CREAT, 777, 0);

if (sem\_calc == SEM\_FAILED) {

perror("Semaphores doesn't create");

exit(1);

}

sem\_unlink("/calc");

pid\_t pr = -1;

pr = fork();

if (pr < 0) {

write(1, "Can't create process\n", 22);

} else if (pr > 0) {

while (read(0, cmd, 100)) {

parsed = parser(cmd);

sprintf(f\_in\_m, "%d %d %s", parsed->cmd, parsed->val, parsed->path);

if (parsed->cmd == 3) {

return 0;

}

for (int i = 0; i < 100; i++) {

cmd[i] = '\0';

}

sem\_post(sem\_calc);

}

sem\_post(sem\_calc);

close(fd);

} else {

while (1) {

sem\_wait(sem\_calc);

queue \*q = q\_create();

sscanf(f\_in\_m, "%d %d %32s", &parsed->cmd, &parsed->val, parsed->path);

int k = 0;

while (parsed->path[k] != '\0') {

push(q, parsed->path[k]);

k++;

}

if (q\_size(q) == 0) {

push(q, '\0');

}

if (parsed->cmd == 3) {

return 0;

} else if (parsed->cmd == 2) {

if (test == NULL) {

while (q\_size(q) != 0) {

pop(q);

}

test = node\_create(parsed->val);

test->is\_root = true;

} else {

add(&test, parsed->val, q);

}

} else if (parsed->cmd == 1) {

pnode\* f = search(&test, q);

if (test == NULL) {

write(1, "empty tree\n", 11);

} else if ((\*f) == NULL) {

write(1, "its root\n", 9);

rmv(&test);

} else {

rmv(f);

}

} else if (parsed->cmd == 0) {

if (test == NULL) {

write(1, "empty tree\n", 11);

} else {

tree\_print(test, 0);

}

} else if (parsed->cmd == -2){

write(1, "invalid value\n", 14);

} else if (parsed->cmd == -1) {

write(1, "invalid path\n", 13);

} else if (parsed->cmd == -777) {

write(1, "invalid command\n", 16);

}

q\_destroy(q);

lseek(fd, 0, SEEK\_SET);

write(fd, "", 100);

}

sem\_close(sem\_calc);

munmap(f\_in\_m, 100);

lseek(fd, 0, SEEK\_SET);

write(fd, "", 100);

close(fd);

}

return 0;

}

**Примеры работы**

ilya@DESKTOP-TUSAG1B:/mnt/d/Study/Labs2course/os\_lab\_1/src/cmake-build-debug$ strace -T -i ./OS\_lab4 cat main.c | tail

[00007fd9350b4e37] execve("./OS\_lab4", ["./OS\_lab4", "cat", "main.c"], 0x7fffc2efbf40 /\* 19 vars \*/) = 0 <0.205473>

[00007f11e701bec9] brk(NULL) = 0x7fffe7a57000 <0.000046>

[00007f11e700f7de] access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory) <0.000159>

[00007f11e701ce27] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory) <0.000070>

[00007f11e701ccdd] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3 <0.000099>

[00007f11e701cc43] fstat(3, {st\_mode=S\_IFREG|0644, st\_size=67794, ...}) = 0 <0.000046>

[00007f11e701cf43] mmap(NULL, 67794, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f11e7238000 <0.000110>

[00007f11e701ced7] close(3) = 0 <0.000037>

[00007f11e7018139] access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory) <0.000113>

[00007f11e701ccdd] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3 <0.000184>

[00007f11e701cda4] 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 <0.000035>

[00007f11e701cc43] fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0 <0.000025>

[00007f11e701cf43] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f11e7230000 <0.000035>

[00007f11e701cf43] mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f11e6de0000 <0.000107>

[00007f11e701cff7] mprotect(0x7f11e6dfa000, 2093056, PROT\_NONE) = 0 <0.000032>

[00007f11e701cf43] mmap(0x7f11e6ff9000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f11e6ff9000 <0.000091>

[00007f11e701cf43] mmap(0x7f11e6ffb000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f11e6ffb000 <0.000040>

[00007f11e701ced7] close(3) = 0 <0.000126>

[00007f11e7018139] access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory) <0.000180>

[00007f11e701ccdd] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3 <0.000344>

[00007f11e701cda4] 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 <0.000085>

[00007f11e701cc43] fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0 <0.000043>

[00007f11e701cf43] mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f11e69e0000 <0.000222>

[00007f11e701cff7] mprotect(0x7f11e6bc7000, 2097152, PROT\_NONE) = 0 <0.000051>

[00007f11e701cf43] mmap(0x7f11e6dc7000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f11e6dc7000 <0.000156>

[00007f11e701cf43] mmap(0x7f11e6dcd000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f11e6dcd000 <0.000057>

[00007f11e701ced7] close(3) = 0 <0.000106>

[00007f11e701cf43] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f11e7220000 <0.000066>

[00007f11e7001024] arch\_prctl(ARCH\_SET\_FS, 0x7f11e7220740) = 0 <0.000024>

[00007f11e701cff7] mprotect(0x7f11e6dc7000, 16384, PROT\_READ) = 0 <0.000042>

[00007f11e701cff7] mprotect(0x7f11e6ff9000, 4096, PROT\_READ) = 0 <0.000039>

[00007f11e701cff7] mprotect(0x7f11e7602000, 4096, PROT\_READ) = 0 <0.000038>

[00007f11e701cff7] mprotect(0x7f11e7227000, 4096, PROT\_READ) = 0 <0.000035>

[00007f11e701cfd7] munmap(0x7f11e7238000, 67794) = 0 <0.000134>

[00007f11e6de5eb5] set\_tid\_address(0x7f11e7220a10) = 190 <0.000026>

[00007f11e6de5f17] set\_robust\_list(0x7f11e7220a20, 24) = 0 <0.000023>

[00007f11e6df295d] rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f11e6de5cb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f11e6df2890}, NULL, 8) = 0 <0.000022>

[00007f11e6df295d] rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f11e6de5d50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f11e6df2890}, NULL, 8) = 0 <0.000023>

[00007f11e6de5ff3] rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0 <0.000022>

[00007f11e6af5fa0] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=8192\*1024}) = 0 <0.000014>

[00007f11e6af64b9] brk(NULL) = 0x7fffe7a57000 <0.000014>

[00007f11e6af64b9] brk(0x7fffe7a78000) = 0x7fffe7a78000 <0.000055>

[00007ffff0554519] gettimeofday({tv\_sec=1582616029, tv\_usec=669971}, NULL) = 0 <0.000032>

[00007f11e6ac58e7] getpid() = 190 <0.000016>

[00007f11e6aefc8e] openat(AT\_FDCWD, "/tmp/tmpf.Cdk524", O\_RDWR|O\_CREAT|O\_EXCL, 0600) = 3 <0.000514>

[00007f11e6af1d47] unlink("/tmp/tmpf.Cdk524") = 0 <0.000980>

[00007f11e6df1281] write(3, " "..., 100) = 100 <0.000133>

[00007f11e6df1b57] lseek(3, 100, SEEK\_END) = 200 <0.000026>

[00007f11e6df1281] write(3, "\0", 1) = 1 <0.000072>

[00007f11e6aef7c3] fstat(3, {st\_mode=S\_IFREG|0600, st\_size=201, ...}) = 0 <0.000025>

[00007f11e6afba13] mmap(NULL, 201, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f11e7248000 <0.000099>

[00007f11e6aef977] statfs("/dev/shm/", {f\_type=TMPFS\_MAGIC, f\_bsize=4096, f\_blocks=38399999, f\_bfree=9346316, f\_bavail=9346316, f\_files=999, f\_ffree=1000000, f\_fsid={val=[1, 0]}, f\_namelen=255, f\_frsize=4096, f\_flags=ST\_VALID|ST\_NOSUID|ST\_NODEV|ST\_NOATIME}) = 0 <0.000153>

[00007f11e6def84e] futex(0x7f11e6ffe370, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0 <0.000014>

[00007f11e6df1d2b] openat(AT\_FDCWD, "/dev/shm/sem.calc", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory) <0.000216>

[00007f11e6ac58e7] getpid() = 190 <0.000014>

[00007f11e6aef815] lstat("/dev/shm/doGDcd", 0x7ffff0363940) = -1 ENOENT (No such file or directory) <0.000154>

[00007f11e6df1d2b] openat(AT\_FDCWD, "/dev/shm/doGDcd", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 4 <0.000494>

[00007f11e6df1281] write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\21\177\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32 <0.000121>

[00007f11e6afba13] mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f11e7247000 <0.000137>

[00007f11e6af1c27] link("/dev/shm/doGDcd", "/dev/shm/sem.calc") = 0 <0.004898>

[00007f11e6aef7c3] fstat(4, {st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...}) = 0 <0.000057>

[00007f11e6af1d47] unlink("/dev/shm/doGDcd") = 0 <0.001076>

[00007f11e6df1421] close(4) = 0 <0.000018>

[00007f11e6af1d47] unlink("/dev/shm/sem.calc") = 0 <0.000851>

[00007f11e6ac4b1c] clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f11e7220a10) = 191 <0.009969>

[00007f11e6df134e] read(0, add s 3434

"add s 3434\n", 100) = 11 <400.557512>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000019>

[00007f11e6df134e] read(0, prt

"prt\n", 100) = 4 <7.517719>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000029>

[00007f11e6df134e] read(0,

"\n", 100) = 1 <0.886412>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000022>

[00007f11e6df134e] read(0, add s 22

"add s 22\n", 100) = 9 <6.478158>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000019>

[00007f11e6df134e] read(0, add sb 9

"add sb 9\n", 100) = 9 <3.390076>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000022>

[00007f11e6df134e] read(0, prt

"prt\n", 100) = 4 <1.190316>

[00007f11e6df0ab4] futex(0x7f11e7247000, FUTEX\_WAKE, 1) = 1 <0.000038>

[00007f11e6df134e] read(0, ext

"ext\n", 100) = 4 <11.542125>

[00007f11e6ac4e06] exit\_group(0) = ?

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

**Вывод**

Обрел навыки работы с утилитой strace и провел диагностику ранее выполненной лабораторной работы (рассмотрел системные вызовы, используемые программой).