Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №1 по курсу**

**«Операционные системы»**

**Использование strace**

Студент: Ефимов Александр Владимирович

Группа: М80 – 201Б-18

Вариант: 6

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2019

1. **Постановка задачи**

Проанализировать поведение программы используя утилиту **strace**.

1. **Описание программы**

**Strace** – утилита для отладки и диагностики программ. Утилита выводит все системные вызовы, сделанные программой, в последовательности их вызовов (т.е трассировка системных вызовов).

1. **Листинг программы**

За основу взята программа из 4 лабораторной.

|  |
| --- |
| #include <unistd.h>  #include <stdlib.h>  #include <stdio.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <sys/mman.h>  #include <string.h>  #include <fcntl.h>  #ifndef PAGE\_SIZE  #define PAGE\_SIZE 4096  #endif  const size\_t BUFF\_SIZE = 100;  const size\_t CharSize = sizeof(char);  const char FILE\_NAME[] = "c943a334af5f2cc790749b07e3a85733";  int main()  {  int fd;  if ( (fd = open(FILE\_NAME, O\_CREAT | O\_RDWR)) < 0)  {  printf("Failed to create temporary file(\"%s\")\n", FILE\_NAME);  exit(EXIT\_FAILURE);  }  if (posix\_fallocate(fd, 0, PAGE\_SIZE) < 0)  {  printf("File allocation failure(\"%s\")\n", FILE\_NAME);  exit(EXIT\_FAILURE);  }  char \*mappedFile;  if ((mappedFile = mmap(NULL, PAGE\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0)) == MAP\_FAILED)  {  perror("Failed to map file into virtual memory\n");  remove(FILE\_NAME);  exit(EXIT\_FAILURE);  }  //Improvise. Adapt. Overcome.  int \*crappyMutex;  if ((crappyMutex = mmap(NULL, sizeof(int), PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, fd, 0)) == MAP\_FAILED)  {  perror("Failed to create mutex\n");  remove(FILE\_NAME);  exit(EXIT\_FAILURE);  }  \*crappyMutex = 1;  pid\_t pid = fork();  if (pid == -1)  {  printf("Failed to execute\n");  remove(FILE\_NAME);  exit(EXIT\_FAILURE);  }  if (pid == 0)  {  char com[BUFF\_SIZE];  int i = 0;  //Get the command from parrent  while (\*crappyMutex) { ;}  while (i != BUFF\_SIZE)  {  com[i] = mappedFile[i];  mappedFile[i] = '\0';  if (com[i] == '\0')  {  ++i;  break;  }  ++i;  }  dup2(fd, STDOUT\_FILENO);  dup2(fd, STDERR\_FILENO);  //Use the command and sent result to parent  char \*result[] = {"sh", "-c", com, NULL};  execv("/bin/sh", result);  }  else  {  char ComIn[BUFF\_SIZE];  int i = 0;  do {  scanf("%c", &ComIn[i]);  if (ComIn[i] == '\n')  {  ++i;  break;  }  ++i;  } while (i != BUFF\_SIZE);  //Write the command to child  for (int j = 0; j < i; ++j)  { mappedFile[j] = ComIn[j]; }  mappedFile[i + 1] = '\0';  \*crappyMutex = 0;  //Wait for child  waitpid(pid, NULL, 0);  //Parsing process  for (i = 0; mappedFile[i] != '\0'; ++i)  {  if (mappedFile[i] == ' ') printf("\_");  else if (mappedFile[i] == '\t') printf("\_\_\_");  else printf("%c", mappedFile[i]);  }  }  remove(FILE\_NAME);  munmap(mappedFile, PAGE\_SIZE);  munmap(crappyMutex, sizeof(int));  return EXIT\_SUCCESS;  } |

Программа запускает терминал, куда вводится команда, после чего в выходных данных этого терминала пробелы заменяются на “\_”, а TAB – на “\_\_\_”. В этом терминале через cat будет прочитал тестовый файл:

|  |
| --- |
| This is space delimited.  This is tab delimited. |

Программа запущена через команду: strace -o out.txt ./ReformerMapped.out, где *out.txt* – выходной файл трассировки, а *ReformerMapped.out* – бинарный файл программы.'

1. **Результат трассировки.**

|  |
| --- |
| execve("./ReformerMapped", ["./ReformerMapped"], 0x7fffc1f877c0 /\* 63 vars \*/) = 0  brk(NULL) = 0x556eb8928000  access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)  access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "tls/haswell/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "tls/haswell/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "tls/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "tls/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "haswell/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "haswell/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/tls/haswell/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/tls/haswell/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/tls/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/tls/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/haswell/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/haswell/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/x86\_64/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "./lib/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3  fstat(3, {st\_mode=S\_IFREG|0644, st\_size=100387, ...}) = 0  mmap(NULL, 100387, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fc7cc3fd000  close(3) = 0  access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)  openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3 |

До чтения программы выполняются такие действия, как подключение динамических библиотек (*access* проверяет права доступа пользователя к файлу, *openat* открывает файловый дескриптор, *fstat* описывает характеристики файла).

|  |
| --- |
| 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  fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0  mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc7cc3fb000  mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fc7cbdfe000  mprotect(0x7fc7cbfe5000, 2097152, PROT\_NONE) = 0  mmap(0x7fc7cc1e5000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7fc7cc1e5000  mmap(0x7fc7cc1eb000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fc7cc1eb000  close(3) = 0  arch\_prctl(ARCH\_SET\_FS, 0x7fc7cc3fc500) = 0  mprotect(0x7fc7cc1e5000, 16384, PROT\_READ) = 0  mprotect(0x556eb6b87000, 4096, PROT\_READ) = 0  mprotect(0x7fc7cc416000, 4096, PROT\_READ) = 0  munmap(0x7fc7cc3fd000, 100387) = 0 |

Здесь уже начинается чтение содержимого программы (как видно по заголовку **ELF**) и его отображение на память (*read* читает содержимое в указанном файловом дескрипторе, *mmap* отображает дескриптор на виртуальную память (если стоит флаг MAP\_ANONYMOUS, то просто совершает аллокацию в ней), *mprotect* меняет права отображения, *close* закрывает файловый дескриптор, *munmap* освобождает виртуальную память).

|  |
| --- |
| openat(AT\_FDCWD, "c943a334af5f2cc790749b07e3a85733", O\_RDWR|O\_CREAT, 0450) = 3  fallocate(3, 0, 0, 4096) = 0  mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fc7cc415000  mmap(NULL, 4, PROT\_READ|PROT\_WRITE, MAP\_SHARED|MAP\_ANONYMOUS, 3, 0) = 0x7fc7cc414000  clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fc7cc3fc7d0) = 12818  fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0  brk(NULL) = 0x556eb8928000  brk(0x556eb8949000) = 0x556eb8949000  read(0, "cat test.txt\n", 1024) = 13  wait4(12818, NULL, 0, NULL) = 12818  --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=12818, si\_uid=1000, si\_status=0, si\_utime=585, si\_stime=0} ---  fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0  write(1, "This\_is\_space\_delimited.\n", 25) = 25  write(1, "This\_\_\_is\_\_\_tab\_\_\_delimited.\n", 29) = 29  unlink("c943a334af5f2cc790749b07e3a85733") = 0  munmap(0x7fc7cc415000, 4096) = 0  munmap(0x7fc7cc414000, 4) = 0  exit\_group(0) = ?  +++ exited with 0 +++ |

Данный фрагмент соответствует ходу выполнения программы (*openat* вызывается функцией open(FILE\_NAME, O\_CREAT | O\_RDWR), *fallocate* – posix\_fallocate(fd, 0, PAGE\_SIZE), *mmap* – mmap, *clone* – fork(), *read* – scanf, *wait4* – waitpid, *write* – printf, *unlink* – remove, *munmap* – munmap).

Отслеживание выполнения данного фрагмента позволяет определить действия программы на уровне ядра, а также, так как виден результат работы каждой команды, возможно обнаружить системный вызов, приводящий к ошибке.

1. **Вывод**

Strace может быть использован, если отсутствует исходный код (поток программы можно определить по вызовам), а также для определения взаимодействия программы с файловой системой.