ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ 8 ЛАБОРАТОРНОЙ РАБОТЫ**

**ПО ДИСЦИПЛИНЕ «ОПЕРАЦИОННЫЕ СИСТЕМЫ»**

Выполнил(а) студент

группы М8О-208Б-23

Федорова Екатерина Васильевна

Проверили и приняли:

Живалев Е.А.

Катаев Ю. И.

Москва, 2024

**Тема:** «Диагностика ПО»

**Цели работы:**

* Приобретение практических навыков диагностики работы программного обеспечения

**Задание:**

Провести анализ системных вызовов во всех выполненных лабораторных работах по курсу ОС с целью:

* Выявления ключевых системных вызовов
* Подтверждения соответствия использованных вызовов заданиям работ
* Составления итогового отчета с результатами исследования

**Инструменты для диагностики:**

В зависимости от используемой операционной системы, доступны следующие средства анализа:

Для Windows:

* Отладчик WinDbg
* Набор утилит Sysinternals Suite:
  + Handle.exe - анализ открытых дескрипторов
  + Procmon.exe - мониторинг активности процессов
  + Procexp.exe - расширенный диспетчер процессов

Для Unix-подобных систем:

* strace - основной инструмент трассировки системных вызовов

Strace является мощным инструментом диагностики в Linux-системах, который позволяет:

* Отслеживать взаимодействие программ с ядром через системные вызовы
* Анализировать поведение программы на низком уровне
* Выявлять проблемы производительности и ошибки в работе приложений

Основные опции strace для эффективной диагностики:

* «-o файл» - сохранение вывода в указанный файл
* «-e выражение» - фильтрация системных вызовов по заданному шаблону
* «-f» - отслеживание дочерних процессов
* «-t» - добавление временных меток к каждому вызову
* «-y» - отображение путей для файловых дескрипторов
* «-p pid» - присоединение к работающему процессу
* «-k» - трассировка стека вызовов

**Анализ log\_lab3.txt:**

14063 execve("./lab3/parent", ["./lab3/parent"], 0x7fff0b313088 /\* 91 vars \*/) = 0

Запуск родительского процесса с PID 14063. Успешное выполнение execve подтверждается нулевым кодом возврата.

14063 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x78504e1f3000

Выделение анонимной памяти размером 8192 байт для внутренних нужд процесса. Память имеет права на чтение и запись.

14063 clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x78504e1dea10) = 14481

14063 clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x78504e1dea10) = 14482

Создание двух дочерних процессов с PID 14481 и 14482. Флаги указывают на автоматическую очистку идентификатора потока и отправку SIGCHLD при завершении.

14481 dup2(3, 1) = 1

14482 dup2(4, 1) = 1

Перенаправление стандартного вывода (дескриптор 1) для каждого дочернего процесса на новые дескрипторы (3 и 4 соответственно).

14481 execve("./lab3/child", ["child", "/shm\_channel1"], 0x7fffb223f828)

14482 execve("./lab3/child", ["child", "/shm\_channel2"], 0x7fffb223f828)

Замена кода дочерних процессов на программу child с разными параметрами для каналов разделяемой памяти.

14063 mmap(NULL, 320, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x78504e22c000

14063 mmap(NULL, 320, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 6, 0) = 0x78504e1f2000

Создание двух областей разделяемой памяти по 320 байт каждая для межпроцессного взаимодействия.

14063 read(0, "Eins! Hier kommt die Sonne,\n", 1024) = 28

Чтение пользовательского ввода родительским процессом. Прочитано 28 байт.

14481 write(1, ",ennoS eid tmmok\n", 17) = 17

Запись обработанных данных первым дочерним процессом. Строка перевернута, что соответствует заданию.

14063 --- SIGINT {si\_signo=SIGINT, si\_code=SI\_KERNEL} ---

14481 --- SIGINT {si\_signo=SIGINT, si\_code=SI\_KERNEL} ---

14482 --- SIGINT {si\_signo=SIGINT, si\_code=SI\_KERNEL} ---

Получение сигнала SIGINT (Ctrl+C) всеми процессами, что приводит к их корректному завершению.

**Заключение:**

На примере лабораторной работы №3 было продемонстрировано, как с помощью strace можно отследить все ключевые системные вызовы:

* Создание процессов через clone()
* Организацию разделяемой памяти через mmap()
* Перенаправление ввода/вывода через dup2()
* Базовые операции чтения/записи через read() и write()

Анализ системных вызовов подтвердил корректность реализации межпроцессного взаимодействия через разделяемую память и правильную обработку сигналов завершения.

**Вывод:**

В процессе выполнения лабораторной работы были успешно освоены инструменты диагностики strace и ltrace. Эти утилиты показали себя как эффективные инструменты для отслеживания системных вызовов и анализа работы программ. Хотя изначально вывод утилит может показаться сложным для понимания из-за большого объема информации, использование различных ключей фильтрации (например, -e trace для отбора конкретных системных вызовов) позволяет получить именно те данные, которые необходимы для анализа.