В ходе выполнения задания был написан BASH-скрипт для автоматизации запуска ftrace из консоли.

Для выполнения задания потребуется запускать исследуемое ПО во втором терминале и передавать в первый терминал, через аргументы командной строки, PID исследуемого процесса.

Были исследованы две программы по методике, предложенной в материалах к занятию:

1. ftrace_demo
2. Своя версия демонстрационной программы с названием main

На скриншоте ниже, в левом окне, можно ознакомится с порядком работы.

Сначала, в правом окне, запускается программа ftrace_demo, она выводит в консоль PID процесса и ожидает нажатия на клавишу ENTER.

После того, как мы выяснили PID изучаемого процесса, в левом окне запускается скрипт ftrace_start.sh, ему передается PID 4950. После его запуска происходит инициализация ftrace с заданными параметрами и запускается трассировка. После чего скрипт ожидает нажатия клавиши ENTER.

После запуска скрипта в левом окне, в правом окне производится нажатие клавиши ENTER, программа совершает различные системные вызовы и завершает работу.

Далее, в левом окне необходимо нажать ENTER, после чего завершается работа ftrace, сбрасывается инициализация, а вывод ftrace копируется в /tmp/ftrace_results.txt

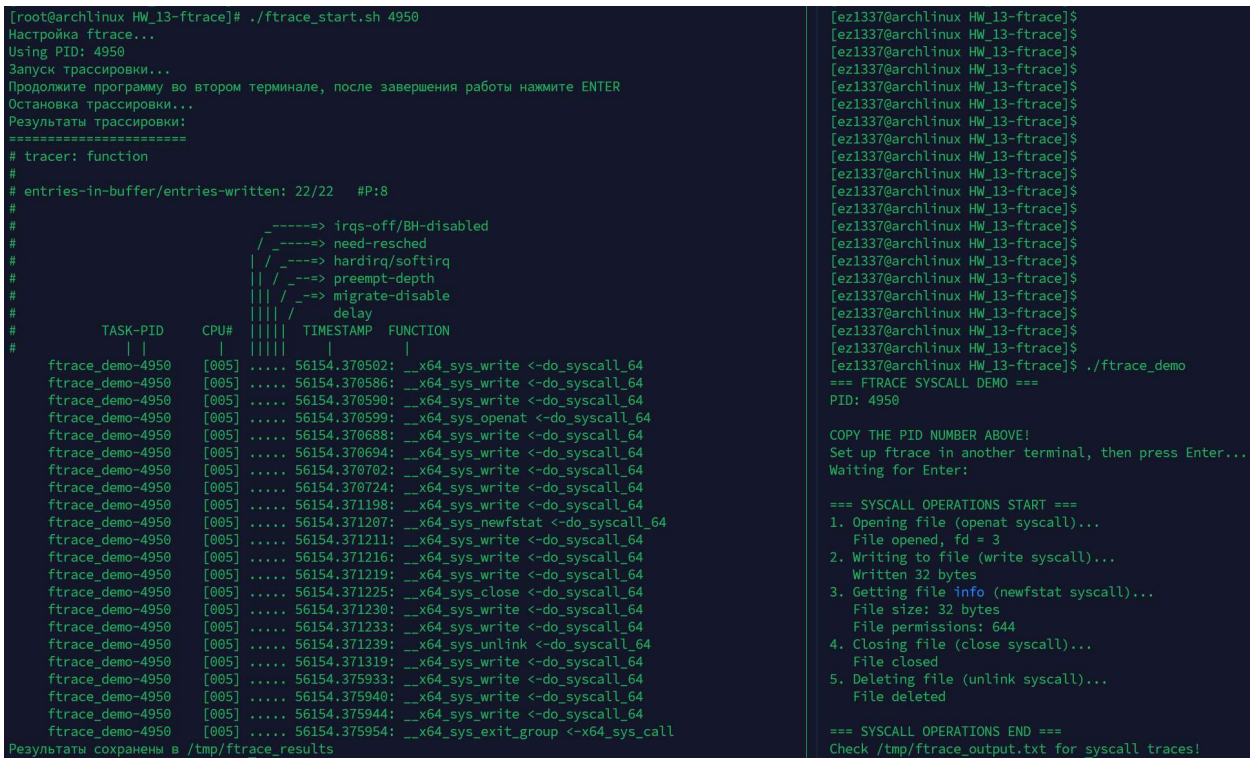

Рисунок 1 – Вывод ftrace для программы ftrace_demo

Аналогичные действия выполним для программы main. В код этой программы добавлено еще несколько примеров системных вызовов.

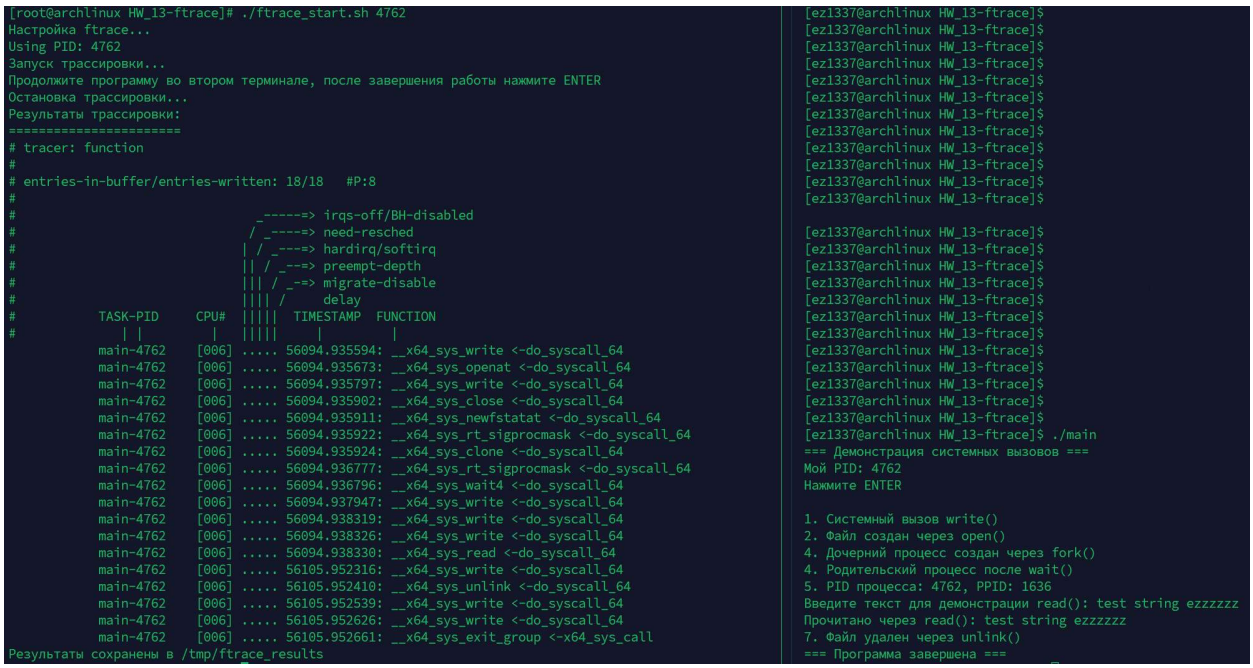

Рисунок 2 – Вывод ftrace для программы main

Проанализируем результаты трассировки для программы main.

Используется трейсер function. Всего записано 18 событий, а система имеет 8 ядер.

__x64_sys_write соответствует вызову **write(STDOUT_FILENO, "1. Системный вызов write()\n", 27)**

__x64_sys_openat – открытие файла **open("test_file.txt", O_CREAT | O_WRONLY | O_TRUNC, 0644)**

__x64_sys_newfstatat – получение информации о файле **stat("syscall_demo.c", &file_stat)**

__x64_sys_rt_sigprocmask управление маской сигналов, вызывается перед fork **fork()**

__x64_sys_clone – создание процесса **fork()**

__x64_sys_rt_sigprocmask - восстановление маски сигналов после создания процесса

__x64_sys_wait4 – ожидание дочернего процесса **wait(NULL)**

Как видно, вызов fork реализован через clone, wait через wait4, opennat используется вместо open, newfstatat вместо fstat, exit_group вместо exit.

__x64_sys_read – чтение ввода **read(STDIN_FILENO, buffer, ...)**

__x64_sys_unlink – удаление файла **unlink("test_file.txt")**

__x64_sys_exit_group – завершение процесса **return 0 из main()**

Заключение

В ходе выполнения работы была написана простая демонстрационная программа на Си, настроен ftrace и собраны данные о системных вызовах, получен опыт работы с трассировщиком, проведен первичный анализ системных вызовов.

Исходные коды, скрипт, скриншоты, логи и Make файл выложены на github по ссылке HW_13-ftrace.