Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Операционные системы»

Студент: Дубровин Дмитрий Константинович
Группа: М8О-207Б-21
Вариант: -
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/1droozd1/os_labs

Постановка задачи

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

Задание

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

Системные вызовы

1. **int execve(const char** *filename***, char *const** *argv* **[], char *const** *envp***[]); execve()** выполняет программу, заданную параметром *filename*. Программа должна быть или двоичным исполняемым файлом, или скриптом, начинающимся со строки вида "**#!** *интерпретатор* [аргументы]". В последнем случае интерпретатор -- это правильный путь к исполняемому файлу, который не является скриптом; этот файл будет выполнен как **интерпретатор** [arg] *filename*. *argv* -- это массив строк, аргументов новой программы. *envp* -- это массив строк в формате **key=value**, которые передаются новой программе в качестве окружения (environment). Как *argv*, так и *envp* завершаются нулевым указателем. К массиву аргументов и к окружению можно обратиться из функции **main**(), которая объявлена как **int main(int argc, char *argv[], char *envp[])**.
2. **void *mmap(void** *addr, **size_t** length, **int** prot, **int** flags, **int** fd, **off_t** offset); **mmap()** создает новое отображение в виртуальном адресном пространстве вызывающего процесса. Начальный адрес для нового сопоставления указан в addr. Аргумент length задает длину сопоставления (которая должна быть больше 0). Если addr равен NULL, то ядро выбирает адрес (выровненный по странице), по которому будет создано сопоставление; это наиболее переносимый метод создания нового сопоставления. Если addr не равен NULL, то ядро воспринимает это как подсказку о том, где разместить отображение; в Linux ядро выберет ближайшую границу страницы (но всегда выше или равна значению, указанному /proc/sys/vm/mmap_min_addr) и попытайтесь создать там сопоставление. Если там уже существует другое сопоставление, ядро выбирает новый адрес, который может зависеть от подсказки, а может и не зависеть. Адрес нового сопоставления возвращается в результате вызова.
3. **int mprotect(void *addr, size_t len, int prot);** mprotect() - это системный вызов в Unix-подобных операционных системах, который используется для изменения атрибутов защиты страниц памяти. Аргументы функции mprotect() включают указатель на начало области памяти, размер этой области и флаги, которые управляют атрибутами защиты страниц. Функция может изменять атрибуты для целых страниц памяти, поэтому ее аргументы должны быть выровнены по границе страницы. Аргументы функции: addr - указатель на начало области

памяти, для которой нужно изменить атрибуты защиты. Этот указатель должен быть выровнен по границе страницы. len - размер области памяти, для которой нужно изменить атрибуты защиты. Размер должен быть кратным размеру страницы системы. prot - флаги, которые управляют атрибутами защиты для страниц памяти. mprotect() может использоваться для установки различных атрибутов защиты для страниц памяти, таких как: PROT_NONE: страницы памяти не могут быть доступны ни для чтения, ни для записи. PROT_READ: страницы памяти доступны только для чтения. PROT_WRITE: страницы памяти доступны для чтения и записи. PROT_EXEC: страницы памяти могут быть исполнены как код. Также можно использовать комбинации этих атрибутов, например, PROT_READ | PROT_WRITE для разрешения чтения и записи страниц памяти. Применение mprotect() может быть полезно для защиты критически важных данных или предотвращения ошибок доступа к памяти, например, чтения или записи в области памяти, которая не должна быть изменена. Он также может быть использован для изменения атрибутов доступа к страницам памяти во время выполнения программы, в зависимости от ее потребностей.

4. **ssize_t write(int fd, const void *buf, size_t count);** write() - это системный вызов в Unix-подобных операционных системах, который используется для записи данных из буфера в файловый дескриптор. Он является одним из основных способов записи данных в файлы в Linux. Аргументы функции: fd - файловый дескриптор, куда нужно записать данные. Это может быть, например, дескриптор файла, сокета или консоли. Дескриптор должен быть открыт для записи. buf - указатель на буфер, содержащий данные, которые нужно записать. count - количество байт, которые нужно записать в файловый дескриптор. Функция возвращает количество записанных байт в случае успеха или -1 в случае ошибки, устанавливая переменную errno.

Демонстрация работы

Все в файле, прикрепленном к отчету.

Выводы

Strace – это утилита Linux, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессором и операционной системой. Использование данной утилиты позволяет понять, что процесс пытается сделать в данное время. Strace может быть очень полезен при отладке программ.