# Звіт до лабораторної 4 (System Call API. Hooks & Event Handlers.)

Студента групи ТТП-32

Остренка Олександра

## 1. Умова лабораторної:

## Изображение выглядит как текст, Шрифт, документ, чек Автоматически созданное описание

## 2. Перша частина:

### 2.1. Теоретична частина.

Для початку реалізуємо взаємодію між різними процесами. Реалізуємо дві програми.

* Задача першої дуже проста – виділити шматок пам’яті під тип `int`, після чого зберігати там ціле число, яке кожні п’ять секунд збільшувати на одиницю. Відповідно даний процес буде тривати, допоки його явно не перервуть.
* Задача другої програми – підключитись до процесу першої програми, зчитати яке зараз число знаходиться на відповідній ділянці пам’яті, після чого модифікувати це число, а саме – збільшити на п’ять.

### 2.2. Реалізація.

* *Реалізація першої задачі.*  
  Для початку визначаємо розмір необхідної ділянки пам’яті та, за допомогою функції malloc, виділяємо відповідну ділянку. Явно приводимо вказівник на область пам’яті до вказівника типу int. Виводимо всю необхідну другій програмі інформацію в консоль. Далі запускаємо нескінченний цикл в якому, кожну секунду, будемо збільшувати значення вказівника на одиницю та виводити його в консоль.
* *Реалізація другої задачі.*Спочатку проводиться перевірка правильної кількості вхідних аргументів. Далі витягуємо відповідні аргументи та підключаємося до процесу за допомогою функції ptrace. За допомогою тієї ж функції витягуємо збережене у відповідній комірці пам’яті число int, потім і модифікуємо його – збільшуємо на одиницю, і, нарешті, відключаємося від процесу.

### 2.3. Тестування.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Для наочності запустимо дві програми в різних терміналах. Поки в лівому виконується циклічне збільшення деякого int, що зберігається у відповідній комірці пам’яті, в правому запускаємо кілька разів програму-перехоплювач. Видно, що перехоплення відбулося двічі – коли число було ‘5’ і ‘15’, після чого воно було збільшено на п’ять та перевизначено.

## 3. Друга частина:

### 3.1. Теоретична частина.

Ідея програми дуже проста – підключитись до файлу пристрою клавіатури та виводити в термінал які кнопки натискаються користувачем.

### 3.2. Реалізація.

Для початку треба зрозуміти, де знайти потрібний файл, що буде представляти клавіатуру. Усі файли, що представляють пристрої введення знаходяться в каталозі «dev/input»:  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Всі ці файли використовуються для зчитування введених даних з різних пристроїв. За клавіатуру відповідає якийсь з файлів eventX. Експериментальним шляхом з’ясовуємо, що у автора відповідний файл – event2.

Далі, за допомогою функції open() відкриваємо даний файл для читання. Тепер запускаємо нескінчений цикл для відслідковування будь-яких змін у цьому файлі. Намагаємось за допомогою функції read() прочитати дані з відповідного файлу та записати їх в структуру input\_event, яка містить інформацію про введену подію, таку як тип події, код клавіші та стан клавіші. Відповідно якщо при зчитуванні не було проблем і тип події – натискання клавіші, то дописуємо в файл виводу keylog.txt всю необхідну інформацію про цю клавішу – її код та стан (RELEASED, PRESSED, REPEATED).

### 3.3. Тестування.

Для тестування запускаємо через sudo відповідну програму, тицяємо будь-які клавіші клавіатури та дивимось, що записалось у відповідний файл.

  
Изображение выглядит как текст, снимок экрана, дисплей, Мультимедийное программное обеспечение

Автоматически созданное описание