Звіт

до лабораторної роботи 4

з навчальної дисципліни «Операційні системи»

студента 3 курсу ФКНК групи МІ-32

Красікова Антона Ігоровича

“System Call API. Hooks & Event Handlers”

Умова завдання:

Скористатись можливостями ядра системи (System Call API) для наступних задач:

1) прочитати пам'ять іншого процесу (наприклад, в 1 процесі в змінну записуємо певне значення (наприклад, вводом з клавіатури), а 2й процес слідкує за цією ділянкою пам'ята 1го процесу та показує зміни, якщо вони відбуваються) або альтернативно – зробити сторінку wired / non-paged

2) перехопити (hook) клавіатуру або мишку із неактивного застосунку (тобто, наприклад, виводити у вікні те, що набирають на клавіатурі у іншому application, "перехопити клавіатурне введення") + продумати сценарій демонстрації.

Hints:

** див. /proc/[id]/mem - доступ до пам'яті*

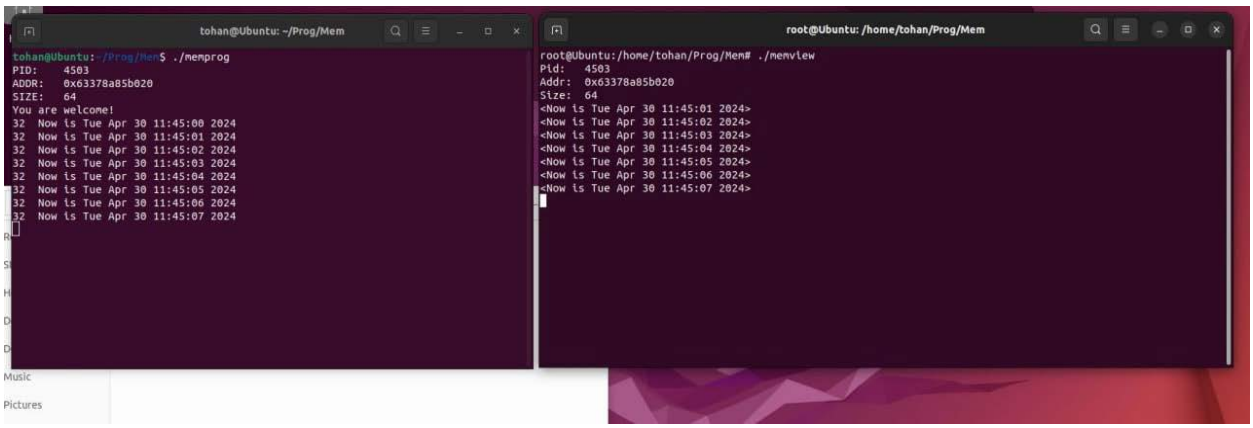
** /dev/input... - доступ до I/O-пристроїв*

Хід роботи:

1. Перше завдання виконується з використанням двох програм. Перша програма виводить необхідну для звернення до її пам’яті інформацію, а саме PID, адресу та довжину блока пам’яті, після чого через певний час починає автоматично змінювати значення в пам’яті, записуючи туди поточній час:

```
#include <unistd.h>
#include <iostream>
#include <ctime>
#include <thread>
#include <cstring>

using namespace std;

// String to change and read
char str[64] = "Now is ";

int main()
{
    // Output mem info
    cout <<
        "PID:    " << getpid()    << "\n" <<
        "ADDR:   " << (void*)str  << "\n" <<
        "SIZE:   " << sizeof(str) << "\n";
    cout << "You are welcome!\n";

    // Change string every 1s afres 20s delay
    for(this_thread::sleep_for(20s);;this_thread::sleep_for(1s))
    {
        time_t now = time(0);
        strcpy(str + 7,asctime(gmtime(&now)));
        cout << strlen(str) << "  " << str;
    }
}
```

Друга програма (яка має бути запущеною з правами root), запитує у користувача необхідну інформацію, після чого щосекунди зчитує пам’ять іншого процесу та виводить її вміст. Два вікна терміналу в операційній системі Ubuntu, що демонструють роботу цих програм, показані на копії екрану нижче:



Код програми, що читає та відображує пам’ять:

```
#include <iostream>
#include <string>
#include <thread>
#include <fcntl.h>
#include <unistd.h>

using namespace std;

int main(int argc, char **argv)
{
    string pid, s;
    unsigned long long addr, size;
    // Get info about memory
    cout << "Pid:   ";
    cin >> pid;
    cout << "Addr:  ";
    cin >> s; addr = stoull(s,nullptr,16);
    cout << "Size:  ";
    cin >> size;

    // device name
    string memname = "/proc/" + pid + "/mem";
    int fd = open(memname.c_str(), O_RDONLY);
    if(fd == -1)
    {
        cerr << "Sorry, error open memory. Try as root\n";
        return -2;
    }
    // Every 1s read mem and output (if it changes)
    for(pid = "", s = string(size,' ');;this_thread::sleep_for(1s))
    {
        // Seek to required position
        off_t res = lseek(fd, (off_t)addr, SEEK_SET);
        if(res == (off_t)-1)
        {
            cerr << "Sorry, error seek memory\n";
            return -3;
        }
        // read chunk
        int readed = read(fd, s.data(), s.size());
        // Limit string
        auto p = s.find('\n');
        if ( p != s.npos) s[p] = 0;

        // Output if different
        if (pid != s)
```

```
        {
            pid = s;
            cout << "<" << s << ">\n";
        }
    }
}
```

2. Щоб перехопити клавіатуру, спершу розглядаємо список пристроїв за допомогою команди `ls /dev/input/by-path/` та бачимо там клавіатуру — `platform-i8042-serio-0-event-kbd` (див. копію екрану нижче).

Програма-перехоплювач (що також має виконуватися з правами root) відкриває цей пристрій та читає його. При помилці читання (EOI) або перериванні програми (EINTR) програма завершує свою роботу; інакше виводить інформацію про натиснуту (або відпущену) клавішу:

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <linux/input.h>
#include <string.h>
#include <stdio.h>

// Device name:
const char *device = "/dev/input/by-path/platform-i8042-serio-0-event-kbd";

int main() {
    int fd = open(device, O_RDONLY);
    if (fd == -1)     {
        fprintf(stderr, "Cannot open %s: error %s.\n", device,
strerror(errno));
        return EXIT_FAILURE;
    }
    for(;;) {
        struct input_event kbd_event;  // Kbd event
        ssize_t n = read(fd, &kbd_event, sizeof(kbd_event));
        if (n == (ssize_t)-1) {
            if (errno == EINTR) continue;
            break;
        }
        else if (n != sizeof(kbd_event)) {
            errno = EIO;
            break;
        }
        if (kbd_event.type == EV_KEY &&
            kbd_event.value >= 0 &&
            kbd_event.value <= 2) {
            const char *acts[3] = {
                "RELEASE",
                "PRESS  ",
                "REPEAT "
```

```
            };
            printf("%s code 0x%04x (%d)\n",
                   acts[kbd_event.value], (int)kbd_event.code,
                   (int)kbd_event.code);
        }
    }
    fflush(stdout);
    fprintf(stderr, "%s\n", strerror(errno));
    return EXIT_FAILURE;
}
```

Ось як виглядає робота програми на екрані Ubuntu. Як можна помітити, було введене слово Hello та натиснута комбінація клавіш Ctrl-C.