

Міністерство освіти і науки України  
Львівський національний університет імені Івана Франка

*Факультет електроніки*  
*Кафедра радіоелектроніки та комп'ютерних систем*

Звіт  
про виконання лабораторної роботи №9  
на тему: «Застосування динамічних бібліотек в ОС  
Windows і Linux»

Виконав студент  
групи ФЕІ - 23  
факультету електроніки  
Попович Василь  
Викладач:  
Сінькевич О. О.

**Мета:** навчитися застосовувати динамічні бібліотеки у ОС Linux та Windows.

**Завдання:** Реалізуйте застосування для Linux і Windows, що може бути розширене під час виконання. Інтерфейс модуля розширення функцій типу void без параметрів. Після запуску застосування видає на екран підказку й очікує введення команди з клавіатури. Можливі такі команди: load ім'я\_модуля (завантаження модуля в пам'ять), unload ім'я\_модуля (вилучення модуля з пам'яті) call ім'я\_функції (виклик функції з модуля). Кожен модуль розширення повинен містити код, який виконується під час його завантаження в пам'ять і вилучення з пам'яті.

### **Теоретичні відомості:**

DLL (англ. Dynamic-link library — динамічно приєднувана бібліотека) — реалізовані компанією Microsoft загальні бібліотеки в ОС Windows та OS/2. Як правило бібліотеки мають розширення файлу \*.dll, \*.ocx (для бібліотек, що містять елементи керування ActiveX) або \*.drv (драйвери старих версій ОС). Структура DLL така сама, як і в PE-файлів (Portable Executable) для 32-, 64-розрядних Windows, та New-Executable (NE) для 16-бітових Windows. DLL може містити код, дані та ресурси в будь-якій комбінації.

Різні програми багатовіконного середовища часто виконують однакові дії, наприклад, хрестик в правому верхньому куті вікна, який закриває його, малюється більшістю програм однаково. Марнотратно було б, щоб кожна із цих програм мала відповідну функцію — це роздувало б їхні розміри. Тому, розумно, щоб такі функції поступали в спільне користування. Для цього служать бібліотеки з динамічним зв'язуванням. Відповідні функції завантажуються в пам'ять комп'ютера не з файлу програми, а із спеціального файлу, вже при виконанні. Насправді, операційна система не завантажує їх повторно. Якщо програма при запуску вимагає завантаження динамічної бібліотеки, то операційна система перевіряє, чи така бібліотека вже є в пам'яті. Якщо вона є, то операційна система збільшує лічильник клієнтів для динамічної бібліотеки на одиницю. При завершенні роботи, програма повідомляє операційну систему про необхідність вивантажити динамічну бібліотеку. При цьому операційна система зменшує лічильник клієнтів на одиницю. Якщо після такого зменшення кількість клієнтів стає рівною нулю, то тоді динамічна бібліотека справді вивантажується із пам'яті комп'ютера.

Під Windows динамічні бібліотеки зберігаються в файлах із розширенням \*.dll. Крім підпрограм там можуть також зберігатися інші ресурси, наприклад, іконки чи бітмапи. В коді програми, що використовує функції з динамічної бібліотеки, завантаження та вивантаження бібліотеки повинні бути прописані безпосередньо.

## **Виконання:**

Було створено дві бібліотеки lib\_1 та lib\_2.

### **Вміст lib1.c**

```
#include <stdio.h>
```

```
void func() {  
    printf("This function from lib1.");  
}
```

### **Вміст lib2.c**

```
#include <stdio.h>
```

```
void func() {  
    printf("This function from lib2.");  
}
```

### **Вміст main.c**

```
#include <dlfcn.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
double (*fn)();
```

```
void* lib = NULL;
```

```
void load(char* libName) {
```

```
    if (lib != NULL) {
```

```

        printf("Some library already loaded!\n");

        return;
    }

    lib = dlopen(libName, RTLD_NOW);

    if (!lib)

        fprintf(stderr, "%s\n", dlerror());

    else

        printf("Library %s loaded.\n", libName);
}

void call(char* funcName) {
    if (lib == NULL) {
        printf("Any library doesn't currently loaded!\n");
        return;
    }

    fn = dlsym(lib, funcName);

    char* error;

    if ((error = dlerror()) != NULL) {
        fprintf(stderr, "%s\n", error);
        return;
    }

    (*fn)();

    printf("\n");
}

```

```

void unload(char* libName) {

```

```

if (lib == NULL) {
    printf("Any library doesn't currently loaded!\n");
    return;
}

dlclose(lib);

printf("Unload %s library.\n", libName);
}

```

```

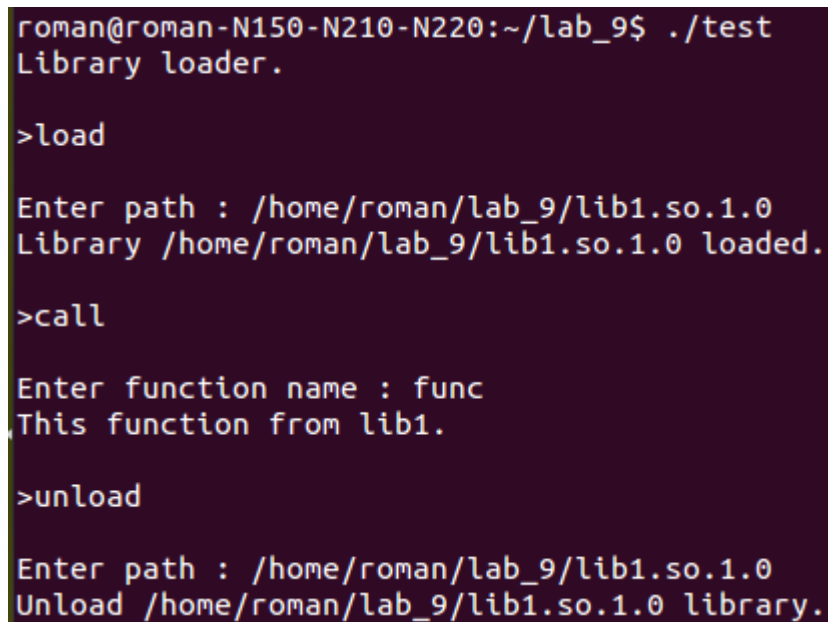
int main() {
    char command[50];
    printf("Library loader.\n");
    while(1) {
        printf("\n>");
        scanf("%s", command);
        if (strcmp(command, "load") == 0) {
            printf("\nEnter path : ");
            scanf("%s", command);
            load(command);
        } else if (strcmp(command, "unload") == 0) {
            printf("\nEnter path : ");
            scanf("%s", command);
            unload(command);
        } else if (strcmp(command, "call") == 0) {
            printf("\nEnter function name : ");
            scanf("%s", command);
            call(command);
        } else if (strcmp(command, "exit") == 0) {

```

```
        return 0;
    } else {
        printf("\nIncorrect input!\n");
    }
}

return 0;
}
```

Результат роботи програми:



```
roman@roman-N150-N210-N220:~/lab_9$ ./test
Library loader.

>load

Enter path : /home/roman/lab_9/lib1.so.1.0
Library /home/roman/lab_9/lib1.so.1.0 loaded.

>call

Enter function name : func
This function from lib1.

>unload

Enter path : /home/roman/lab_9/lib1.so.1.0
Unload /home/roman/lab_9/lib1.so.1.0 library.
```

**Висновок:** на цій лабораторній роботі я навчився застосовувати динамічні бібліотеки.