МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

Отчет по лабораторной работе № 9

по дисциплине: «Системное программирование»

на тему: «Создание и использование библиотек»

Вариант 9

Выполнили**:**  студенты группы 10701222 Махнач М. В.

Дяденькина Т.М.

Шкробот А. А.

Приняла**:** ст. пр. Давыденко Н.В.

Минск 2024

**Цель работы**: Изучить и закрепить на практике создание и использование статически и

динамически подключаемых пользовательских библиотек в операционных

системах семейства Linux.

**Задание 2**

Произвести рефакторинг проекта предыдущей лабораторной работы по автосборке проекта: вынесите код функций бизнес логики в отдельную динамическую библиотеку со статическим вызовом. Для автосборки проекта используйте утилиту make, обязательно.

При помощи программы ldd проверим зависимости, образовавшиеся между библиотеками. Сравните сумму размеров объектных файлов проекта и исполняемого файла. Результаты отобразите скриншотами. Копию исполняемого файла расположите на «Рабочем столе», запустите программу. Какой будет результат? Если возникла проблема, решите ее и поясните каким способом.

Сравните результаты первого и второго упражнения, сделайте вывод.

**Выполнение**

Код:

main.cpp

#include <iostream>

#include "game.h"

#include <dlfcn.h>

typedef bool (\*MyFunctionType)(int, int);

int main() {

int minRange, maxRange, targetNumber, guess, attempts;

char playAgain;

std::cout << "Welcome to the Guess the Number game!" << std::endl;

do {

attempts = 0;

std::cout << "Enter the minimum range: ";

std::cin >> minRange;

std::cout << "Enter the maximum range: ";

std::cin >> maxRange;

std::cout << "Think of a number between " << minRange << " and " << maxRange << "." << std::endl;

// Определение загаданного числа пользователем

std::cout << "Enter your target number: ";

std::cin >> targetNumber;

int left = minRange;

int right = maxRange;

while (left <= right) {

guess = (left + right) / 2; // Оптимальный алгоритм двоичного поиска

attempts++;

bool x;

void\* handle = dlopen("libGame.so", RTLD\_LAZY);

if (handle) {

MyFunction2

Type func = (MyFunctionType)dlsym(handle, "check\_guess");

if (func) {

x=func(guess, targetNumber); // Вызов функции

}

dlclose(handle);

}

if (x) {

std::cout << "The computer guessed your number " << targetNumber << " in " << attempts << " attempts." << std::endl;

break;

}

else if (guess < targetNumber) {

std::cout << "The computer guessed " << guess << ". It's lower!" << std::endl;

left = guess + 1;

}

else {

std::cout << "The computer guessed " << guess << ". It's higher!" << std::endl;

right = guess - 1;

}

}

if (left > right) {

std::cout << "The computer could not guess your number. You win!" << std::endl;

}

std::cout << "Do you want to play again? (Y/N): ";

std::cin >> playAgain;

} while (playAgain == 'Y' || playAgain == 'y');

std::cout << "Thank you for playing the Guess the Number game. Goodbye!" << std::endl;

return 0;

}

game.h

#pragma once

int generate\_random\_number(int min, int max);

bool check\_guess(int guess, int target);

game.cpp

#include "game.h"

#include <cstdlib>

int generate\_random\_number(int min, int max) {

return rand() % (max - min + 1) + min;

}

bool check\_guess(int guess, int target) {

return guess == target;

}

Makefile:

CXX = g++

CXXFLAGS = -fPIC -shared

LDFLAGS = -L. -lGame

TARGET\_LIB = libGame.so

TARGET\_PROG = myprog

# Исходные файлы

SRCS\_LIB = game.cpp

SRCS\_PROG = main.cpp

# Правила сборки

.PHONY: all clean

all: $(TARGET\_LIB) $(TARGET\_PROG)

$(TARGET\_LIB): $(SRCS\_LIB)

$(CXX) $(CXXFLAGS) -o $@ $^

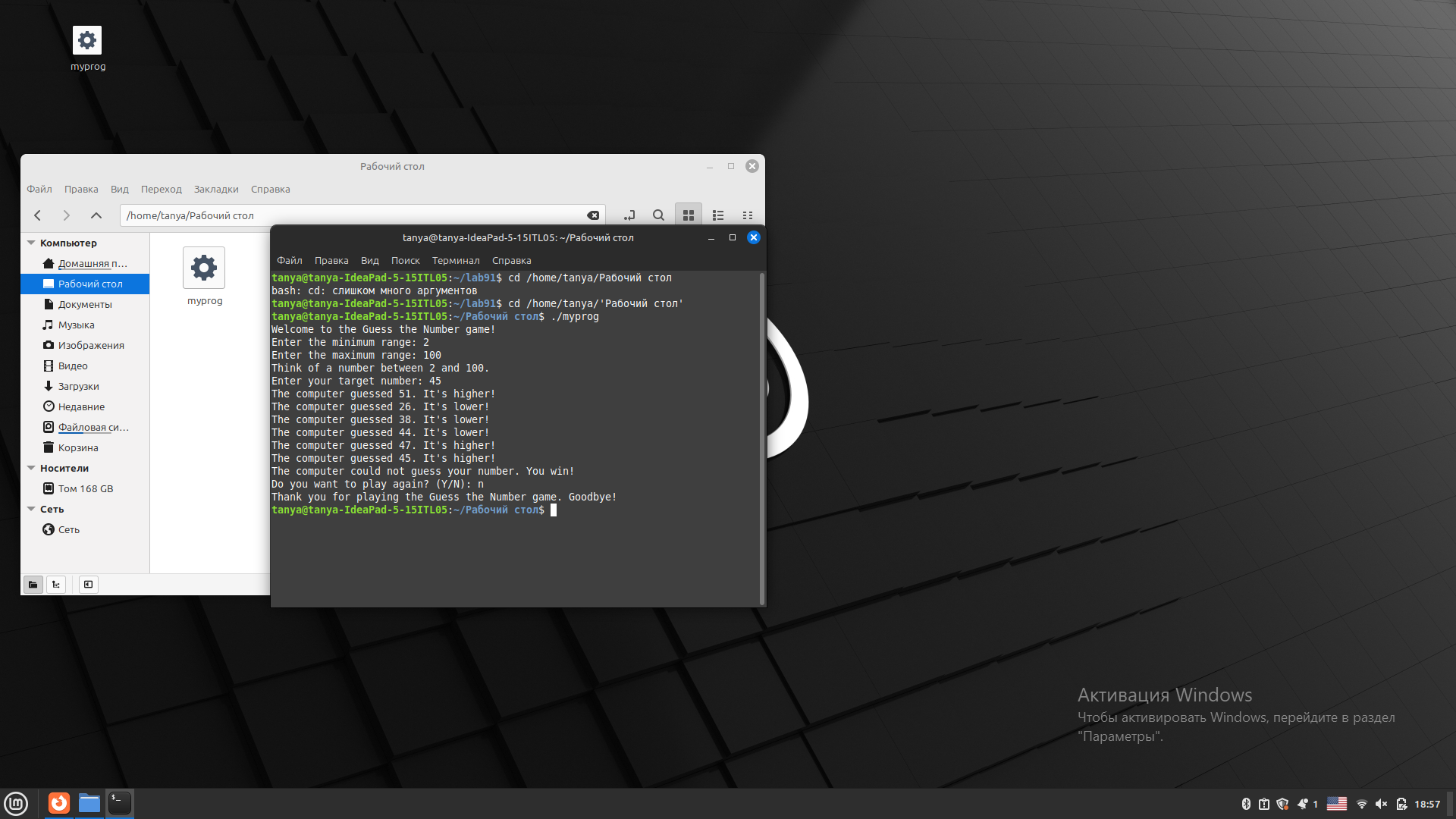
$(TARGET\_PROG): $(SRCS\_PROG)

$(CXX) -o $@ $^ $(LDFLAGS)

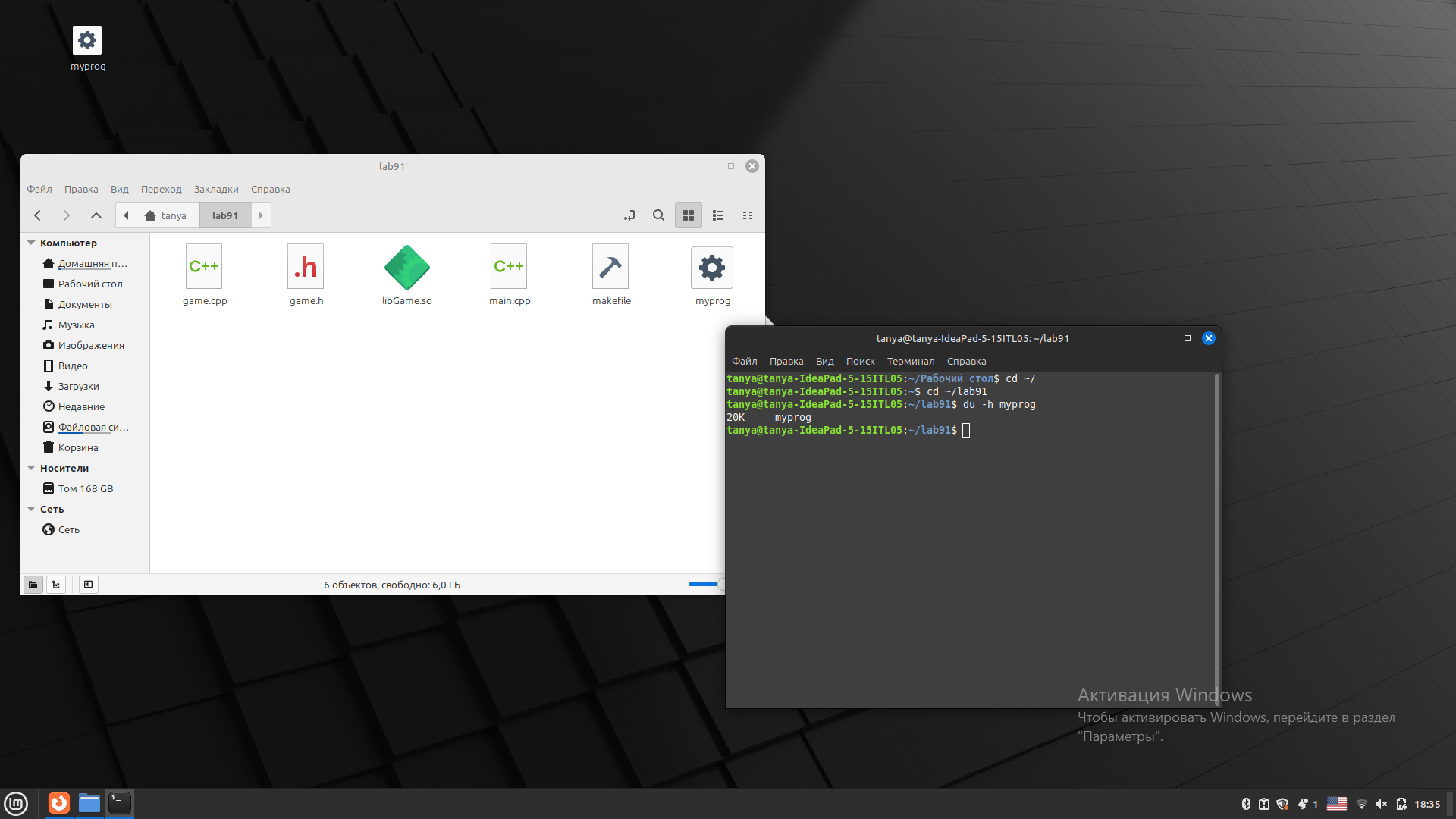
clean:

rm -f $(TARGET\_LIB) $(TARGET\_PROG)

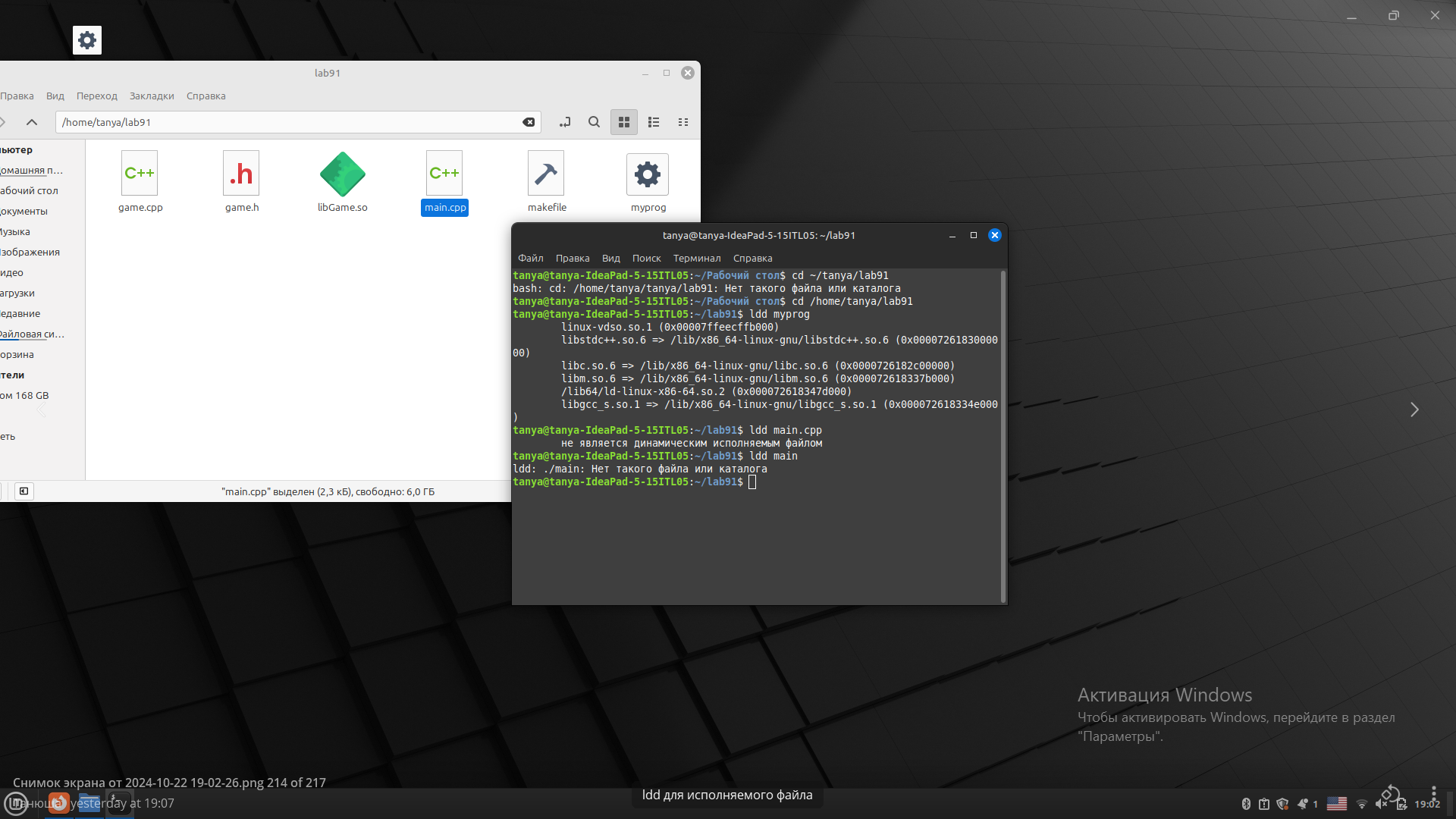
Работа программы:



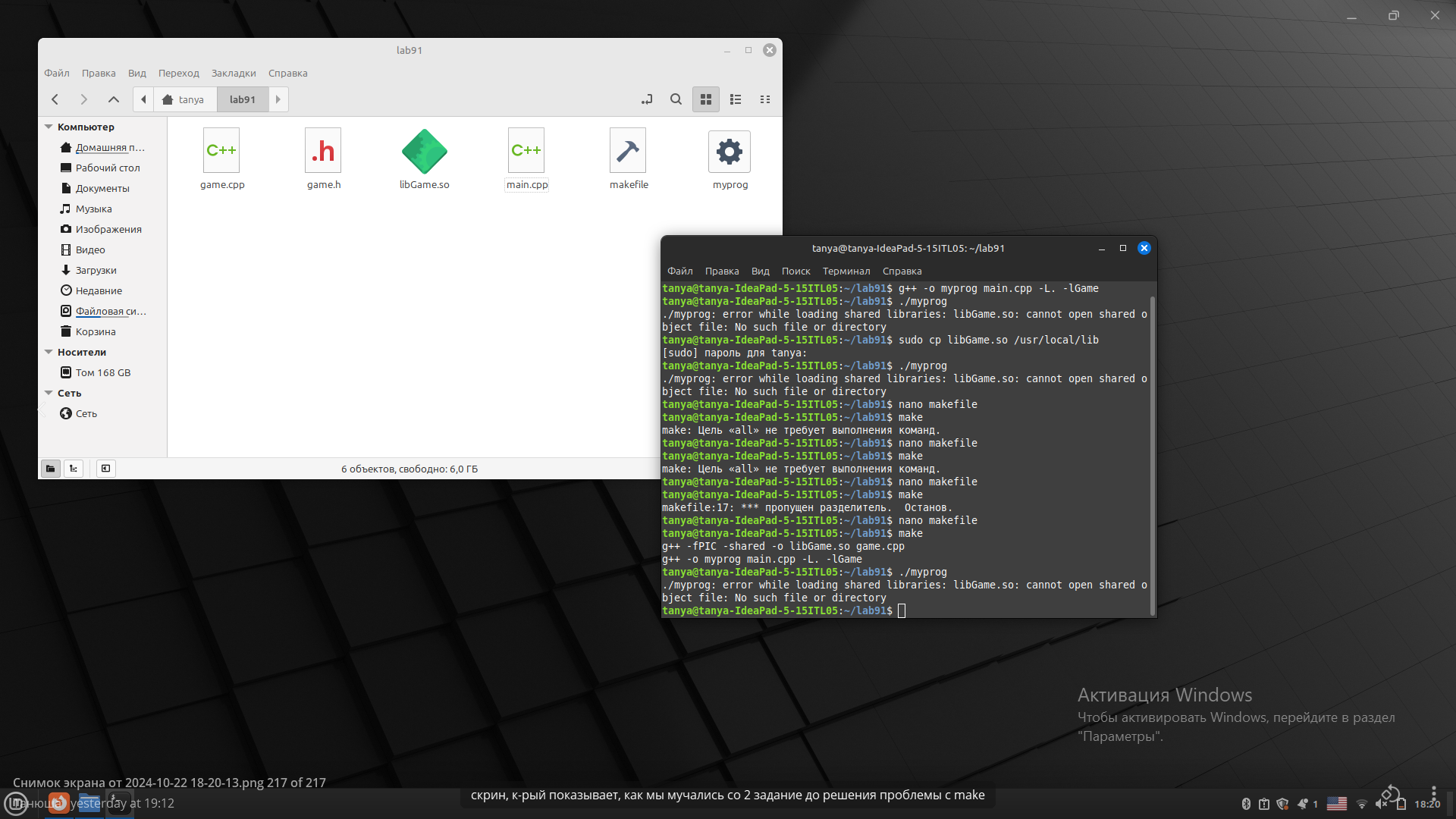
Размеры исполняемых и объектных файлов:



ldd:



Проблема подключения динамической библиотеки статическим методом:



Решение проблемы:

…

#include <dlfcn.h>

typedef bool (\*MyFunctionType)(int, int);

…

int main()

{

…

void\* handle = dlopen("libGame.so", RTLD\_LAZY);

if (handle) {

MyFunction2

Type func = (MyFunctionType)dlsym(handle, "check\_guess");

if (func) {

x = func(guess, targetNumber); // Вызов функции

}

dlclose(handle);

}

…

}

**Контрольные вопросы**

**Какие способы можно использовать для того, чтобы указать местоположение подключаемых динамических библиотек?**

В Linux существует несколько способов указать местоположение подключаемых динамических библиотек (shared libraries):

**1. Путь поиска по умолчанию:**

* **LD\_LIBRARY\_PATH:** Переменная окружения LD\_LIBRARY\_PATH содержит список каталогов, где динамический компоновщик (ld.so) будет искать библиотеки.
  + **Пример:** export LD\_LIBRARY\_PATH=/path/to/libs:/path/to/other/libs:$LD\_LIBRARY\_PATH
* **/etc/ld.so.conf:** Файл /etc/ld.so.conf содержит список каталогов, которые будут добавлены в путь поиска по умолчанию.
  + **Пример:**
  + /usr/local/lib
  + /opt/mylibs
  + После изменения этого файла нужно выполнить sudo ldconfig для обновления кэша библиотек.

**2. Встроенные пути поиска:**

* **Директории системы:** Динамический компоновщик по умолчанию ищет библиотеки в стандартных системных каталогах, таких как /lib, /usr/lib, /usr/local/lib.
* **RPATH:** Внутренняя директива RPATH в объектных файлах позволяет указать путь поиска библиотек, специфичный для конкретного приложения.
  + **Пример:** gcc -Wl,-rpath=/path/to/libs myprogram.c
* **DT\_RUNPATH:** Эта директива в объектных файлах подобна RPATH, но ее значения не наследуются дочерними процессами.

**3. С помощью переменных среды:**

* **LD\_PRELOAD:** Переменная LD\_PRELOAD позволяет загрузить библиотеки перед загрузкой других библиотек. Это может быть полезно для перехвата вызовов функций.
  + **Пример:** export LD\_PRELOAD=/path/to/mylibrary.so
* **LD\_NOW:** При установке LD\_NOW=1, динамический компоновщик будет загружать все зависимости сразу при запуске программы, а не откладывать это на момент первого вызова.
  + **Пример:** LD\_NOW=1 ./myprogram

**4. Использование ldconfig:**

* ldconfig - это утилита, которая сканирует каталоги, указанные в ld.so.conf, и обновляет кэш библиотек, хранящий информацию о доступных библиотеках.
  + **Пример:** sudo ldconfig

**Рекомендации:**

* **Используйте LD\_LIBRARY\_PATH с осторожностью:** Эта переменная может переопределять системные пути поиска и создавать проблемы с конфликтами.
* **Предпочитайте RPATH или DT\_RUNPATH:** Эти директивы позволяют указать путь поиска библиотек, специфичный для приложения, что делает его более переносимым и удобным.
* **Используйте ldconfig:** Не забудьте обновить кэш библиотек после изменения путей поиска.

**Пример:**

Предположим, что у вас есть динамическая библиотека mylib.so, расположенная в каталоге /home/user/mylibs/.

1. **Использование LD\_LIBRARY\_PATH:**
2. export LD\_LIBRARY\_PATH=/home/user/mylibs:$LD\_LIBRARY\_PATH
3. ./myprogram
4. **Использование RPATH:**
5. gcc -Wl,-rpath=/home/user/mylibs myprogram.c -L/home/user/mylibs -lmylib
6. ./myprogram
7. **Использование ldconfig:**
8. sudo echo "/home/user/mylibs" >> /etc/ld.so.conf
9. sudo ldconfig
10. ./myprogram

Выберите способ, который вам больше всего подходит, в зависимости от потребностей вашего приложения.

**Дополнительные сведения:**