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 № 8

по дисциплине:” Системное программирование”

на тему: ”Многозадачность процессов и этапы компиляции ”

Вариант 6

Выполнили**:**  студент группы 10701222 Дяденькина Т.М.

Приняла**:** пр. Давыденко Н.В.

Минск 2024

**Цель работы**: Изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов gcc/g++, отладчика gdb и других для проектирования, компиляции, отладки и запуска приложений на языке программирования С/C++. Изучить базовую концепцию операционной системы, «Процесс». Получить представление о многозадачности процессов

**Задание**

Задание 1

Используя материалы (исходные файлы) упражнения №1 разработайте программу, в которой дочерний процесс запускается системным вызовом fork в цикле, ограниченном счетчиком проходов по циклу. Через пять секунд родительский процесс должен закончить работу, а дочерний процесс выдать команду ps. Внимание. Список выполняющихся процессов можно получить с помощью команды $ps.

Теперь нужно осознать, как осуществляется взаимодействие процессов. Стандартным образом для этой цели используется канал (pipe). Но сейчас мы осуществим взаимодействие через текстовый файл. Создайте программу, в которой один процесс будет писать в файл какую-то строку, а второй будет считывать.

Проведите компиляцию однофайлового проекта c помощью скрипты bash с прохождением всех стадий компиляции.

Сборка проекта должна содержать файлы с результатами препроцессинга.

Исследуйте файлы препроцессора, найдите в них код своей программы. Определите размеры исходных, препроцессорных, ассемблерных, объектных и исполняемых файлов. С помощью соответствующей консольной команды или с помощь проводника определите форматы этих файлов. Результаты подтвердите скриншотами. Сделайте краткий вывод по выполненной работе.

Задание 2

Используя материалы (исходные файлы) упражнения №2 проведите компиляцию проекта листингов 22.5 и 22.6 с помощью утилиты make с прохождением всех стадий компиляции обеих программ.

Сборка проекта должна содержать файлы с результатами препроцессинга. Исследуйте фалы препроцессора, найдите в них код своей программы.

Определите размеры исходных, препроцессорных, ассемблерных, объектных и исполняемых файлов. С помощью соответствующей консольной команды или с помощь проводника определите форматы этих файлов. Результаты подтвердите скриншотами. Сделайте краткий вывод по выполненной работе.

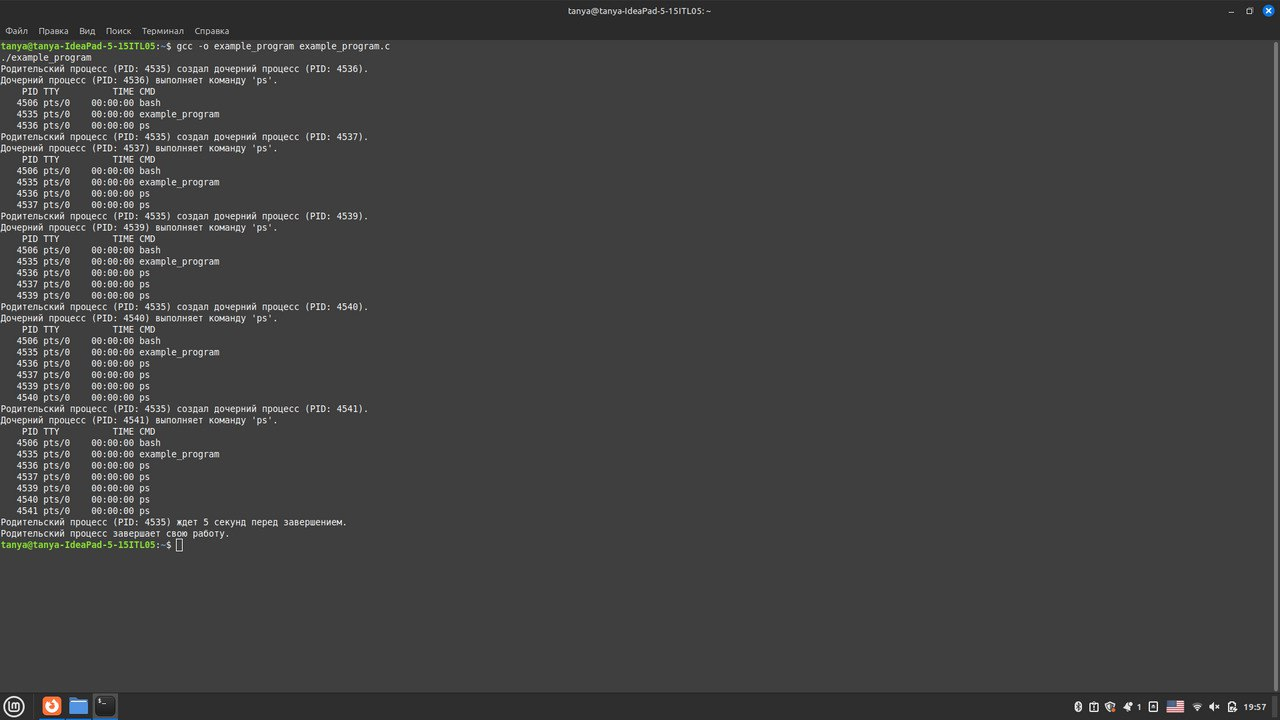
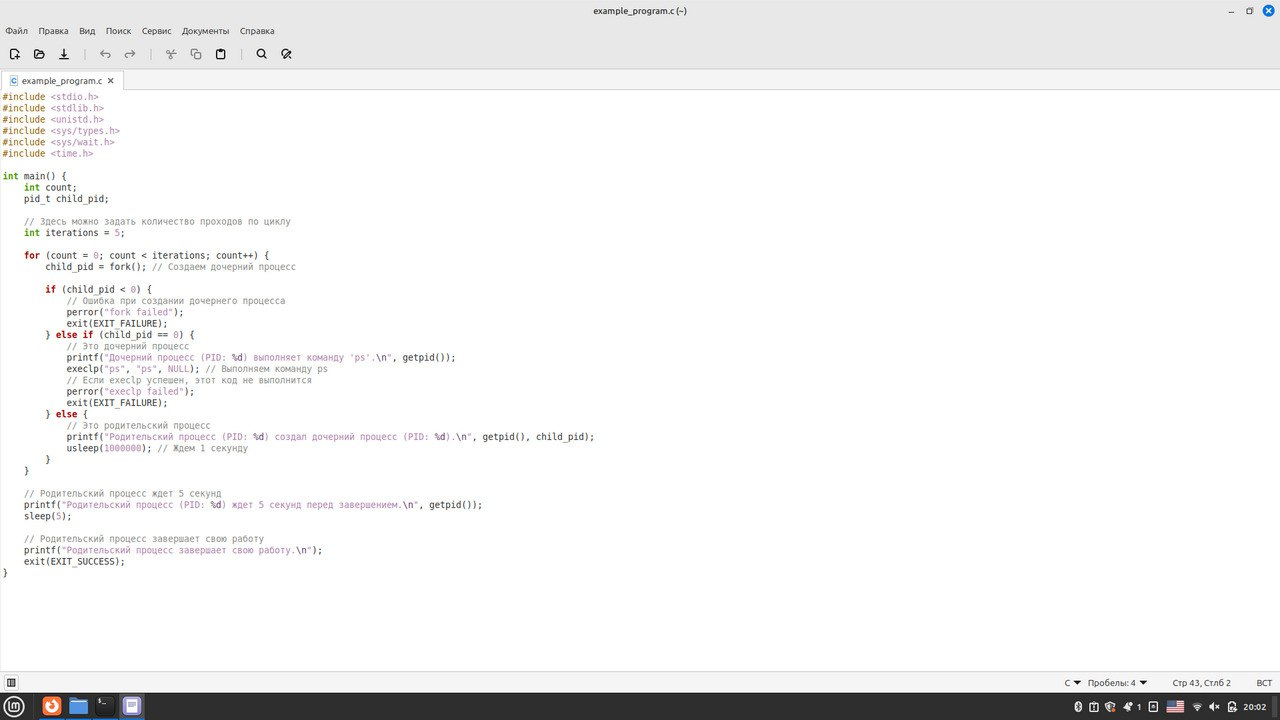
Программа листинга 22.8 должна запускать браузер, установленный в вашей системе, или текстовый редактор, или проводник, или другую, написанную вами программу, а также можно чтобы ваша программа запускала и то и другое и третье.

Задание 3

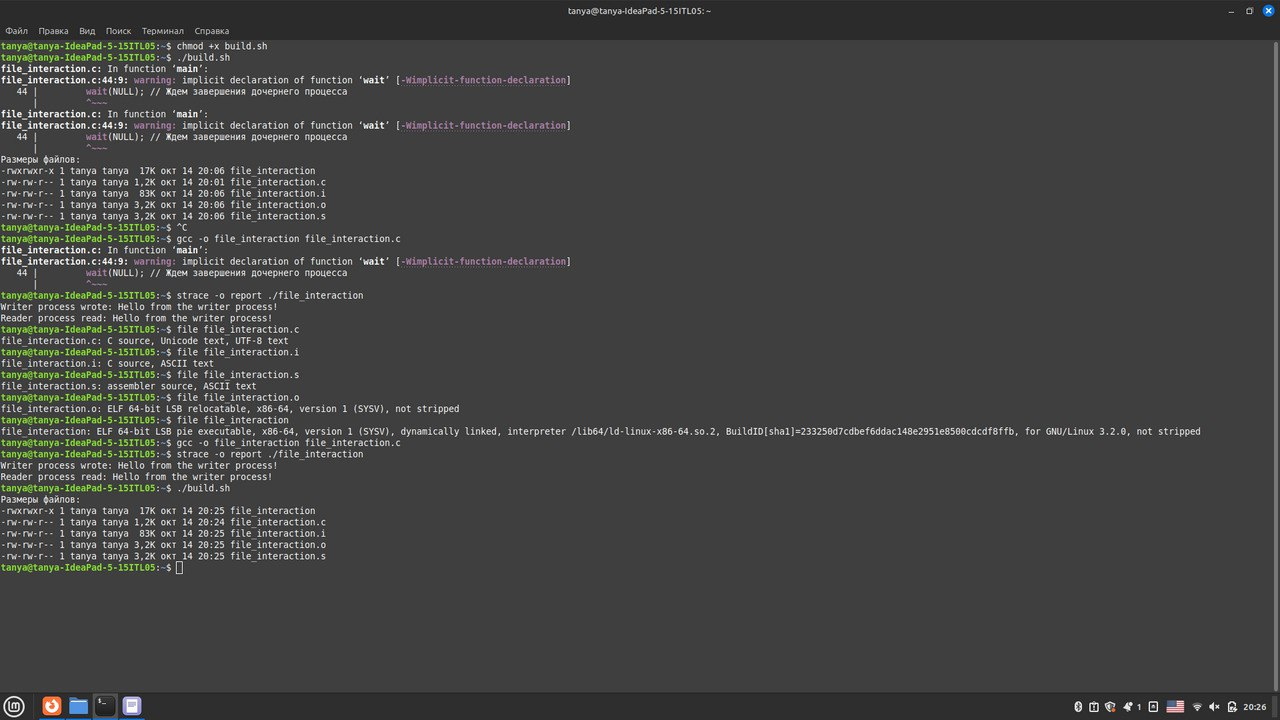
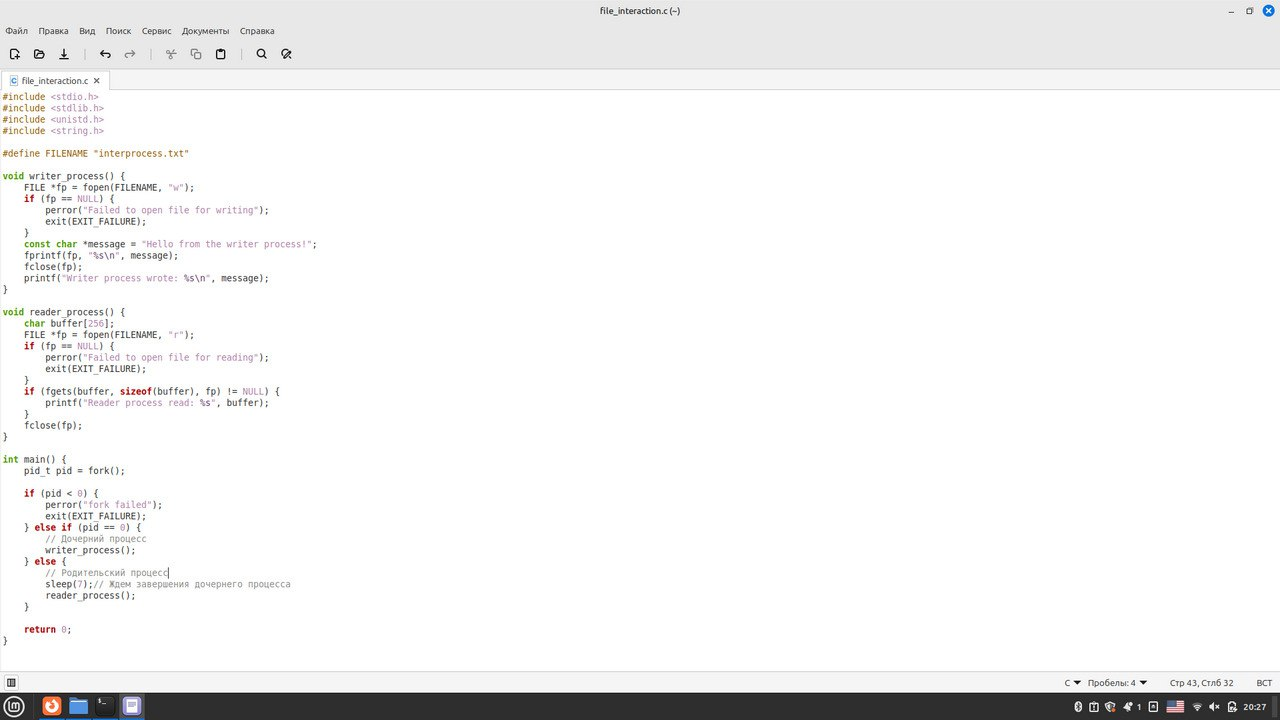
Используя материалы, исходные файлы упражнения №3 (листинги 22.9 – 22.14) создайте программу запускающую «микросервисы»: программу-календарь, браузер, проводник, текстовый редактор, установленные в вашей системе.

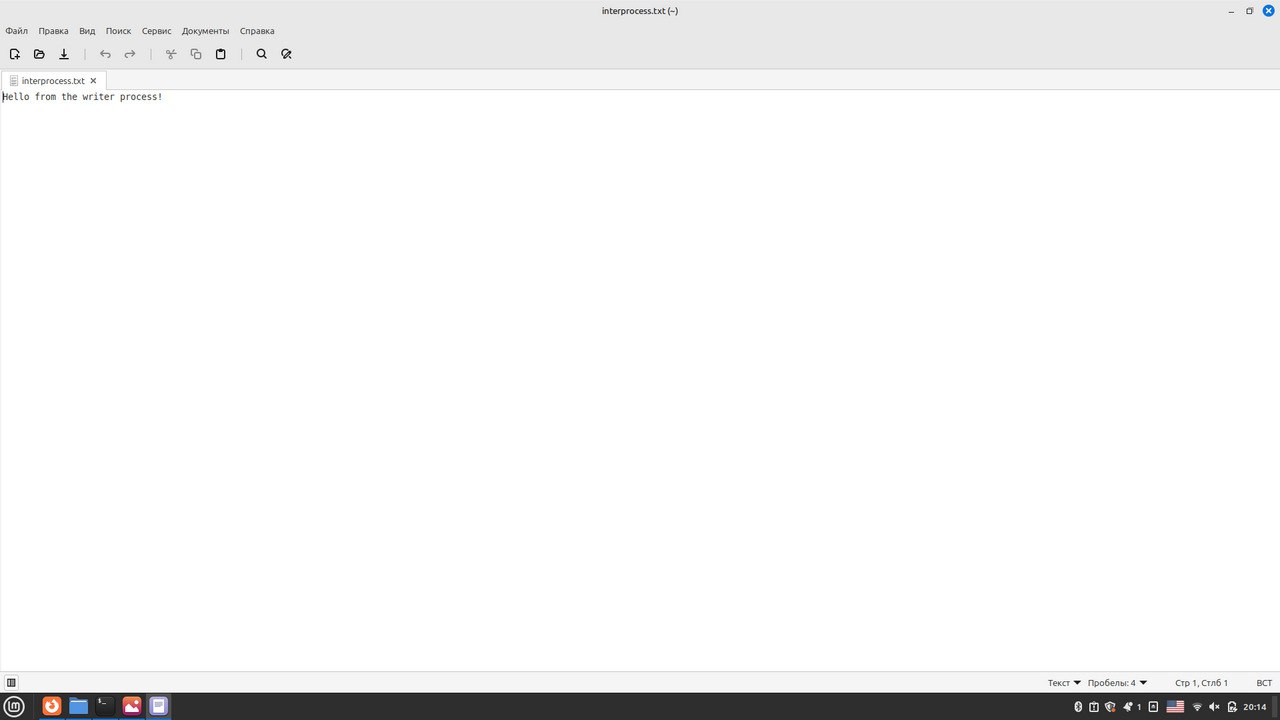
С помощью очень полезной утилиты strace, запустите переданную ей вами разработанную программу и выведите в стандартный поток ошибок отчет об использованных системных вызовах. Полученный результат представьте в отчете и дайте объяснения

**Выполнение 1 задания**

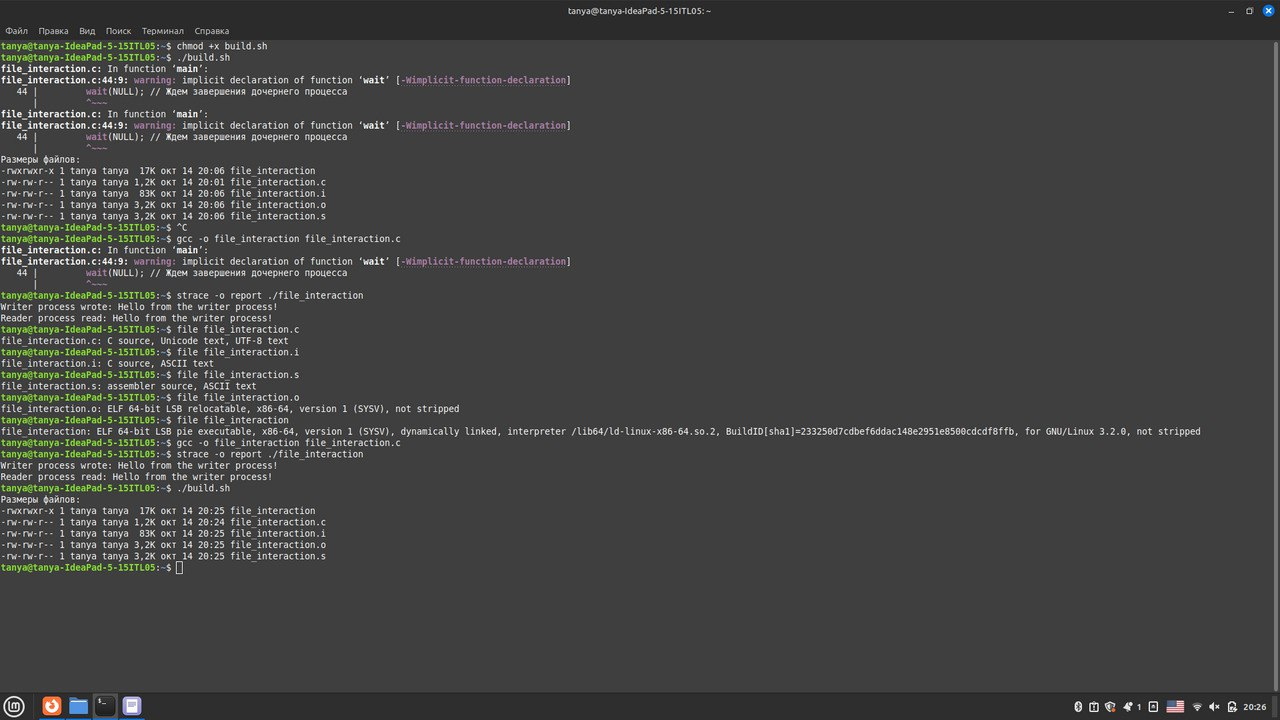
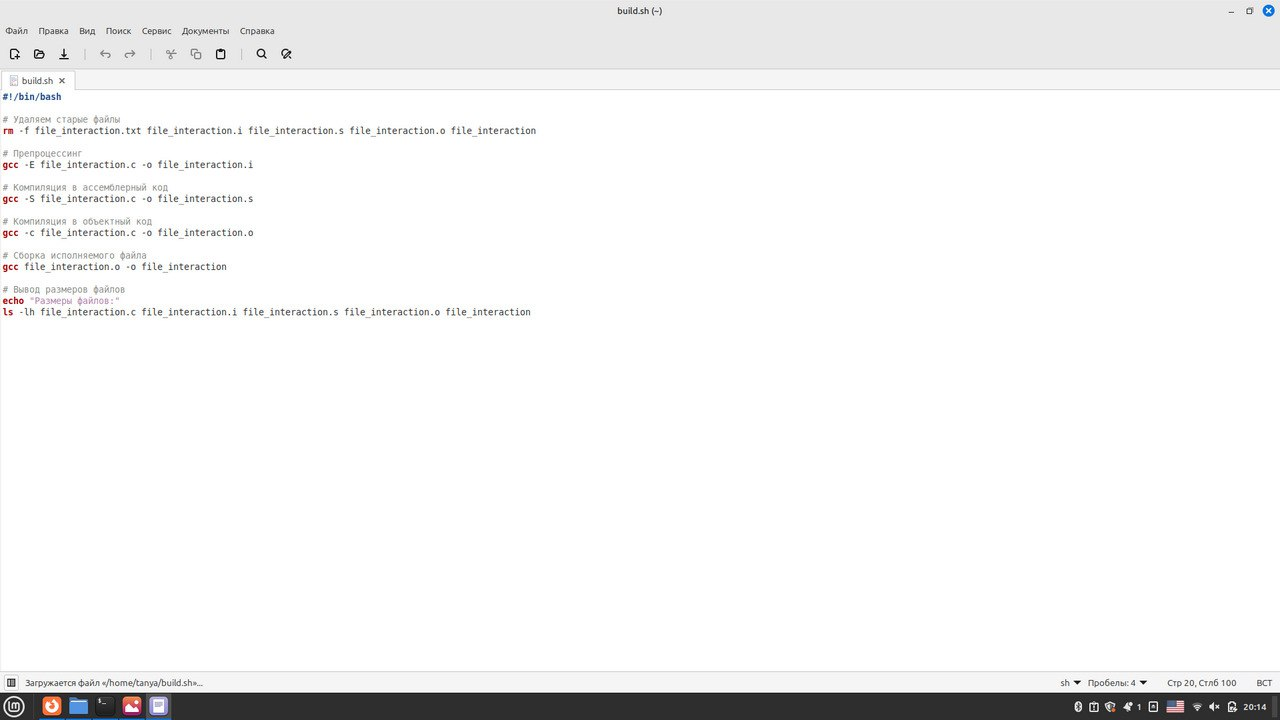
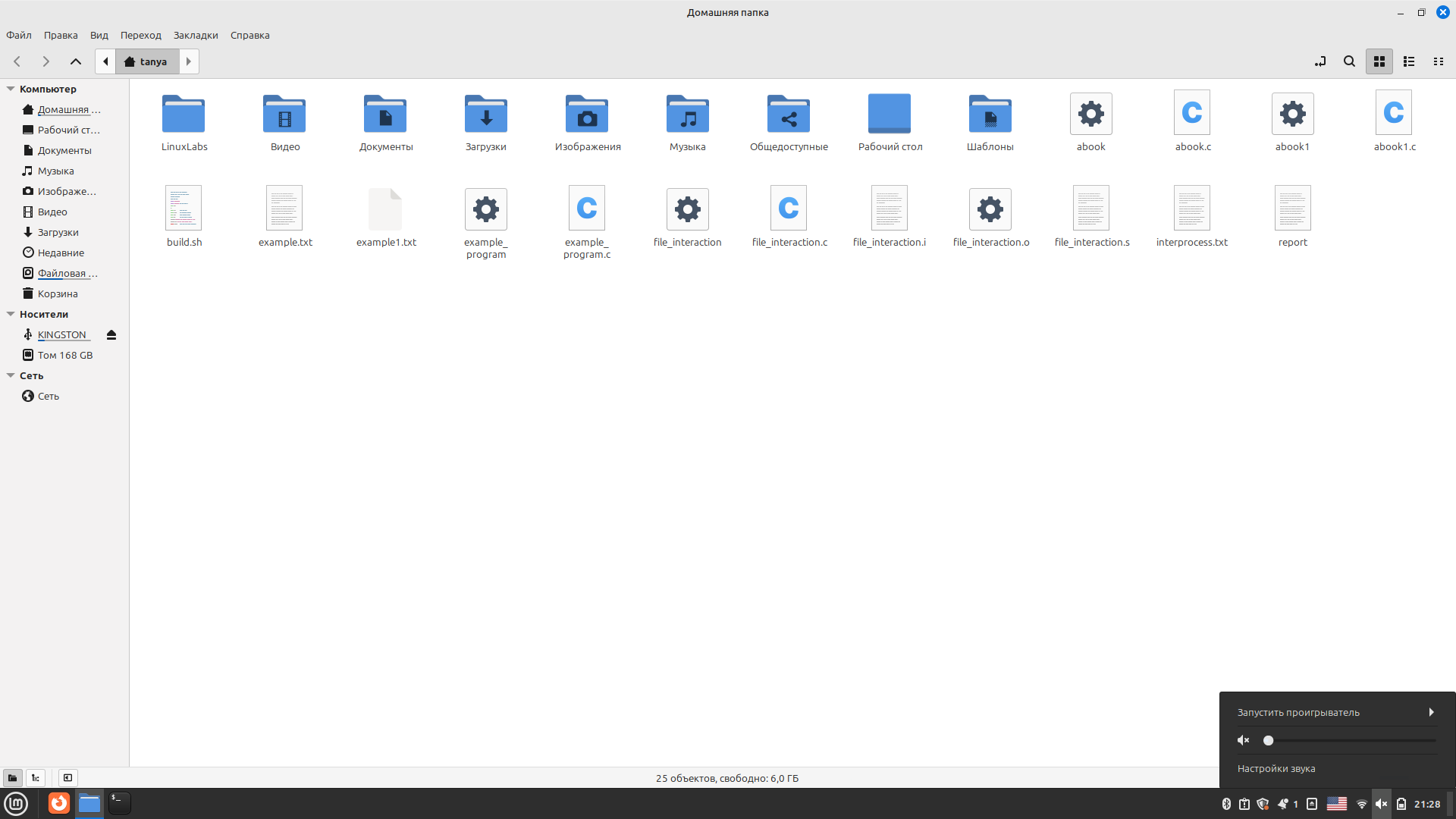
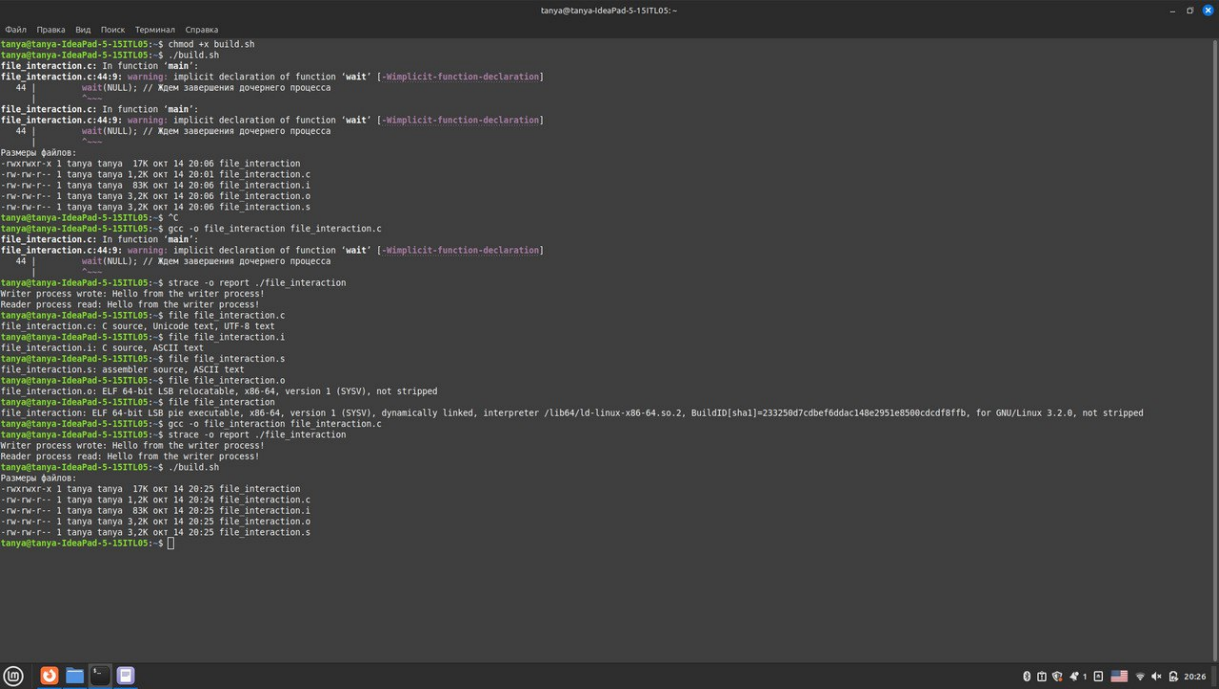
Fork

Programm, which work with file (w and r)





Bash, size of files and type of files

- file\_interaction.c — оригинальный исходный файл.  
- file\_interaction.i — файл после препроцессинга.  
- file\_interaction.s — ассемблерный код программы.  
- file\_interaction.o — объектный файл.  
- file\_interaction — исполняемый файл.

**Контрольные вопросы**

В чем различие компиляции и интерпретации программы?

Ответ:

Основное отличие между ними в том, как они обрабатывают исходный код программы. Компилятор преобразует весь код в машинный язык, а интерпретатор выполняет код построчно. Интерпретатор выполняет одну инструкцию за раз, транслируя и выполняя ее, а затем переходя к следующей.