

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

«NCURSES ОБОЛОЧКА ДЛЯ УТИЛИТЫ FIND»

БГУИР КП 1-40 02 01 116 ПЗ

Студент гр. 250501 Лукьянов Е.О.  
Руководитель: ассистент каф. ЭВМ  
Игнатович А.О.

МИНСК 2024

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1 ОБЗОР ЛИТЕРАТУРЫ .....	4
1.1 Постановка задачи .....	4
1.2 Обзор технологий и алгоритмов поставленной задачи .....	4
1.3 Анализ существующих аналогов .....	6
2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ .....	9
2.1 Модуль взаимодействия с пользователем .....	9
2.2 Модуль создания и редактирования данных файловой системы .....	9
2.3 Модуль обработки исключительных ситуаций .....	9
2.4 Модуль анализа данных файловой системы .....	9
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ .....	10
3.1 Разработка диаграммы классов .....	10
3.2 Описание классов и структур .....	10
3.3 Описание основных функций программы .....	12
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ .....	13
4.1 Разработка схем алгоритмов .....	13
4.2 Разработка алгоритмов .....	13
4.3 Код программы .....	15
5 РЕЗУЛЬТАТ РАБОТЫ .....	16
ЗАКЛЮЧЕНИЕ .....	36
СПИСОК ЛИТЕРАТУРЫ .....	37
ПРИЛОЖЕНИЕ А .....	38
ПРИЛОЖЕНИЕ Б .....	39
ПРИЛОЖЕНИЕ В .....	40
ПРИЛОЖЕНИЕ Г .....	41
ПРИЛОЖЕНИЕ Д .....	42

## ВВЕДЕНИЕ

В современном информационном обществе, где цифровые технологии проникают во все сферы деятельности, управление файловой системой становится ключевым аспектом эффективной работы с компьютером. Утилита «find» является важным инструментом в арсенале системного администратора, программиста или просто пользователя операционной системы UNIX или UNIX-подобной системы. С ее помощью можно осуществлять поиск файлов и каталогов по различным критериям, что делает ее неотъемлемым элементом командной строки для работы с файловой структурой операционной системы.

Несмотря на мощь и гибкость утилиты «find», ее использование через командную строку может быть вызывающим определенные сложности для тех пользователей, которые предпочитают визуальные интерфейсы. В этой связи возникает потребность в разработке оболочки, которая предоставила бы удобный, интуитивно понятный и визуально привлекательный интерфейс для работы с утилитой «find».

Целью данного курсового проекта является разработка такой оболочки на языке программирования C с использованием библиотеки ncurses. Ncurses предоставляет программистам возможность создавать текстовые пользовательские интерфейсы в терминальном окне, что делает его отличным выбором для разработки консольных приложений.

В рамках данного проекта мы сосредоточимся не только на создании удобного в использовании интерфейса для утилиты «find», но и на расширении ее функциональности за счет добавления дополнительных возможностей, которые могут сделать процесс поиска и управления файлами еще более эффективным и удобным для конечного пользователя.

Таким образом, курсовой проект представляет собой не только техническую задачу по разработке программного обеспечения, но и возможность глубже понять принципы работы файловых систем и командной строки в UNIX-подобных операционных системах, а также научиться применять библиотеку ncurses для создания текстовых пользовательских интерфейсов, обеспечивающих удобство и эффективность в работе с файловой структурой системы.

# 1 ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Постановка задачи

Программа ncurses оболочка для утилиты find» должна иметь удобный графический интерфейс и обширный функционал для работы с утилитой find. В программе должно быть реализовано и предусмотрено следующее:

1. Графическое представление утилиты find с использованием библиотеки ncurses. Интерфейс должен включать элементы управления, такие как поля ввода, кнопки, меню и окна для отображения результатов поиска.
2. Обработка ввода пользователя для задания критериев поиска файлов и каталогов. Пользователь должен иметь возможность указать путь к каталогу, критерии поиска, включая имя файла, тип, размер и другие параметры.
3. На основе введенных пользователем параметров формирование команды «find», которая будет передана на выполнение операционной системе Linux.
4. Запуск выполнения команды «find» с помощью системных вызовов. Обработка вывода команды для получения результатов поиска.
5. Вывод результатов поиска в пользовательском интерфейсе с помощью библиотеки ncurses. Результаты должны быть представлены в удобном и понятном формате для пользователя, возможно, с использованием таблиц или списков.

## 1.2 Обзор технологий и алгоритмов поставленной задачи

Для реализации графического интерфейса была выбрана библиотека ncurses. Это библиотека для создания текстовых пользовательских интерфейсов (TUI) в терминале в операционных системах UNIX. ncurses предоставляет разработчикам возможность создавать интерфейсы с использованием окон, панелей, меню и элементов управления, таких как кнопки и поля ввода. Библиотека обеспечивает переносимость и управление курсором, а также обработку событий клавиатуры.

Вот некоторые ключевые особенности и возможности библиотеки ncurses:

1. Переносимость: библиотека ncurses была разработана для обеспечения переносимости программ между различными UNIX-подобными операционными системами. Это означает, что приложения, написанные с

использованием `ncurses`, могут запускаться на различных платформах без необходимости изменения исходного кода.

2. Управление экраном: библиотека позволяет программистам создавать и управлять окнами, панелями, текстовыми полями, кнопками и другими элементами интерфейса на экране терминала.

3. Цвета и стили: библиотека `ncurses` поддерживает использование цветов и различных стилей форматирования текста для создания более привлекательного и информативного пользовательского интерфейса.

4. Управление курсором и клавиатурой: библиотека предоставляет возможность управления положением курсора на экране и обработки ввода с клавиатуры. Это позволяет создавать интерактивные приложения с возможностью взаимодействия пользователя с помощью клавиш.

5. Многопоточность: библиотека `ncurses` поддерживает многопоточность, что позволяет создавать многозадачные приложения с параллельным выполнением кода.

6. Богатая функциональность: библиотека содержит широкий набор функций для работы с экраном, окнами, цветами, клавиатурой и другими аспектами текстового пользовательского интерфейса.

В целом, `ncurses` является мощным инструментом для создания текстовых пользовательских интерфейсов в терминале, который широко используется для разработки консольных приложений в UNIX-подобных операционных системах. Его простота использования и гибкость делают его популярным выбором среди разработчиков, стремящихся создать удобные и интуитивно понятные интерфейсы для своих программ.

На рисунке 1.2.1 представлен пример приложения, основанного на `ncurses`.

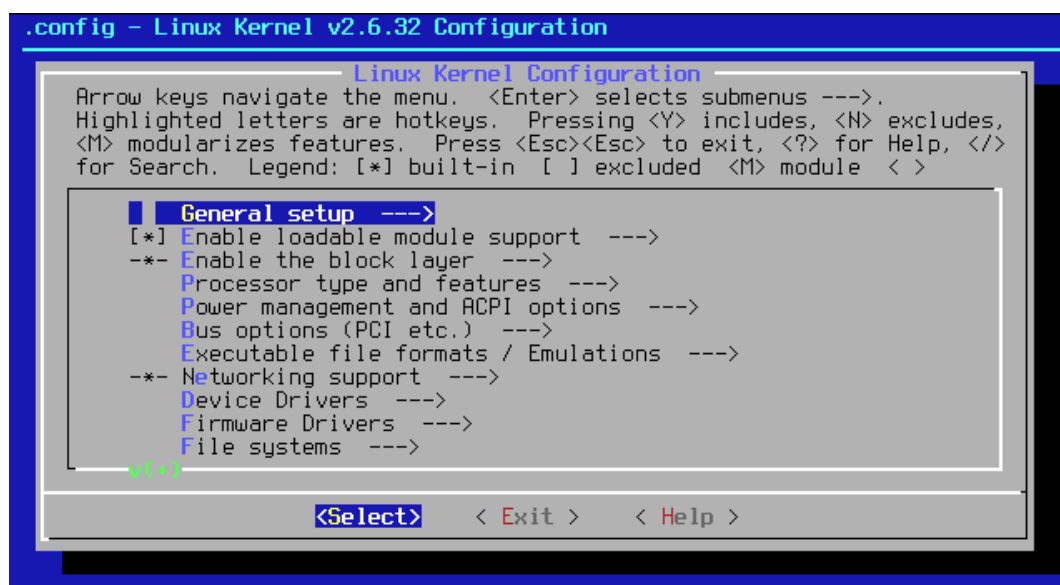


Рисунок 1.2.1 – программа для конфигурации ядра, написанного на ncurses

### 1.3 Анализ существующих аналогов

Когда дело доходит до аналогов приложения «Ncurses оболочка для утилиты find», важно учитывать не только альтернативные библиотеки для создания пользовательского интерфейса в терминале, но и другие программы, которые могут предоставлять аналогичные функциональные возможности. Вот несколько примеров:

1. «Midnight Commander» – это текстовый файловый менеджер для командной строки, который предоставляет множество инструментов для навигации по файловой системе, копирования, перемещения и удаления файлов, а также для выполнения поиска. Включает в себя интерфейс в стиле «Norton Commander» и поддерживает различные функции, включая возможность выполнения поиска файлов. На рисунке 1.3.1 представлен внешний вид данной программы.



Рисунок 1.3.1 – графический интерфейс программы «Midnight Commander»

2. «Ranger» – это еще один текстовый файловый менеджер с открытым исходным кодом для терминала Linux. «Ranger» предоставляет пользователю возможность навигации по файловой системе, предварительного просмотра файлов, выполнения различных действий с файлами и каталогами, включая поиск. На рисунке 1.3.2 представлен внешний вид данной программы.

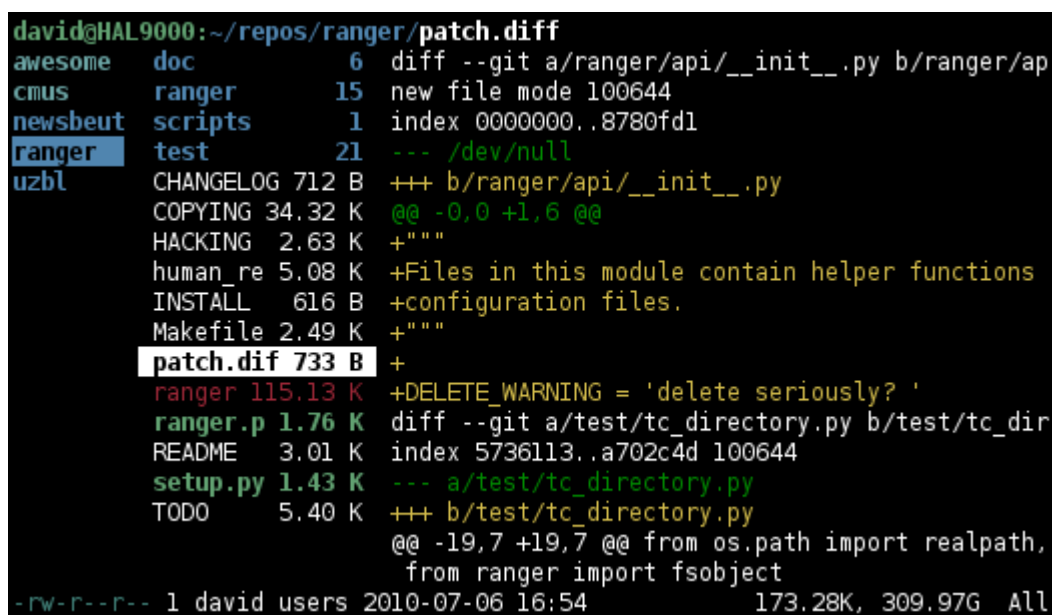
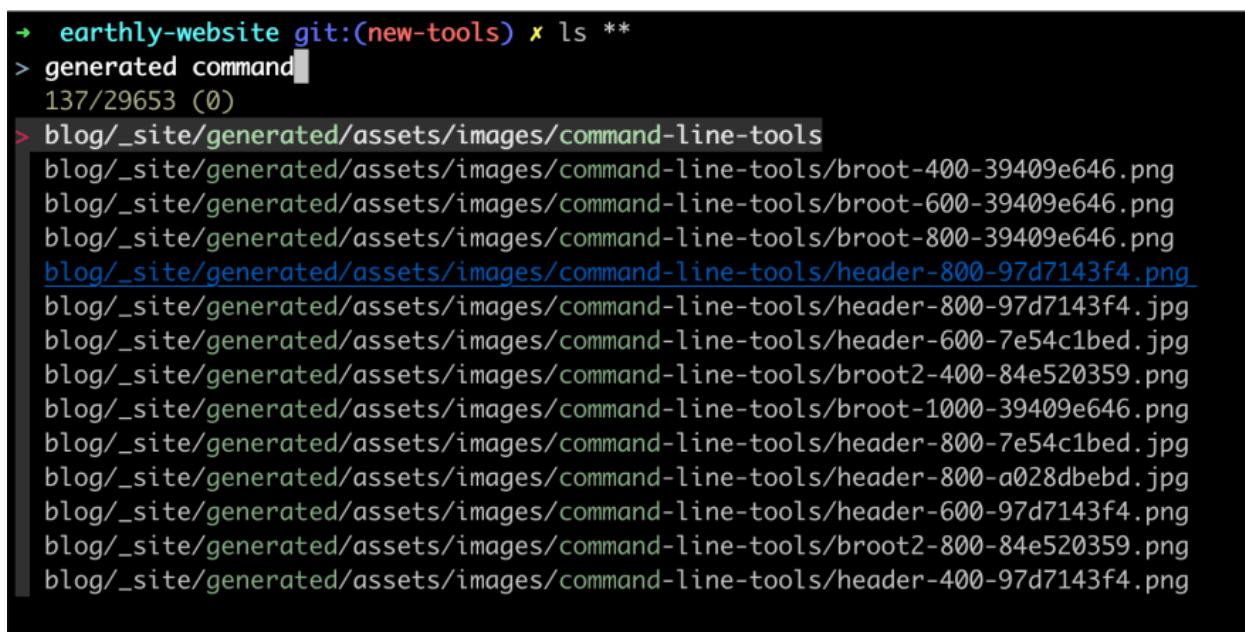


Рисунок 1.3.2 – графический интерфейс программы «Ranger»

3. «FZF (Fuzzy Finder)» – это инструмент командной строки для выполнения интерактивного поиска файлов и каталогов. «FZF» использует различные стратегии сопоставления для быстрого и точного поиска файлов и каталогов, а также поддерживает интеграцию с другими утилитами командной строки.



```
→ earthly-website git:(new-tools) ✕ ls **
> generated command
137/29653 (0)
> blog/_site/generated/assets/images/command-line-tools
blog/_site/generated/assets/images/command-line-tools/broot-400-39409e646.png
blog/_site/generated/assets/images/command-line-tools/broot-600-39409e646.png
blog/_site/generated/assets/images/command-line-tools/broot-800-39409e646.png
blog/_site/generated/assets/images/command-line-tools/header-800-97d7143f4.png
blog/_site/generated/assets/images/command-line-tools/header-800-97d7143f4.jpg
blog/_site/generated/assets/images/command-line-tools/header-600-7e54c1bed.jpg
blog/_site/generated/assets/images/command-line-tools/broot2-400-84e520359.png
blog/_site/generated/assets/images/command-line-tools/broot-1000-39409e646.png
blog/_site/generated/assets/images/command-line-tools/header-800-7e54c1bed.jpg
blog/_site/generated/assets/images/command-line-tools/header-800-a028dbabd.jpg
blog/_site/generated/assets/images/command-line-tools/header-600-97d7143f4.png
blog/_site/generated/assets/images/command-line-tools/broot2-800-84e520359.png
blog/_site/generated/assets/images/command-line-tools/header-400-97d7143f4.png
```

Рисунок 1.3.3 – графический интерфейс программы «Midnight Commander»

4. «Ack, Ag (The Silver Searcher)» – это инструменты командной строки для выполнения быстрого и эффективного поиска текста в файлах. Они предоставляют возможности для поиска по содержимому файлов с использованием регулярных выражений и поддерживают вывод результатов в удобном для анализа формате. На рисунке 1.3.5 представлен внешний вид данной программы.

Каждое из этих приложений имеет свои уникальные особенности и преимущества, которые могут быть полезны в различных сценариях использования. При выборе аналога для приложения важно учитывать требования и потребности конечных пользователей, а также функциональные возможности и интеграционные возможности предлагаемых вариантов.



## **2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

После определения функциональности приложения необходимо разбить его на функциональные блоки. Такой подход предотвращает проблемы с проектированием приложения, улучшает читаемость кода и делает его более гибким для будущих изменений.

### **2.1 Модуль взаимодействия с пользователем**

Этот модуль отвечает за создание пользовательского интерфейса для обеспечения взаимодействия пользователя с приложением. Для реализации данного модуля используется графическая библиотека ncurses, которая позволяет создавать простые графические интерфейсы.

### **2.2 Модуль обработки и составления запроса find**

Данный модуль обрабатывает как простые find запросы, так и сложные, учитывая логические операции: И, ИЛИ, НЕ. Модуль используется для формирования команды поиска файлов на основе пользовательского ввода. Он выполняет парсинг входной строки запроса и формирует соответствующую команду, которая затем передаётся утилите «find» для поиска по заданным пользователем параметрам.

### **2.3 Модуль чтения и записи пользовательских настроек и выполнения поиска**

Этот модуль отвечает за чтение и запись пользовательских настроек, формирование команд для выполнения операций в файловой системе и получение результатов выполнения этих операций в виде файлового указателя.

### **2.4 Модуль обработки пользовательского ввода**

Этот модуль, скорее всего, является частью библиотеки, которая предоставляет набор функций для работы с графическим интерфейсом консольного приложения. Он включает функции для обработки пользовательского ввода, отрисовки элементов интерфейса и обработки текстовых строк.

## 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

### 3.1 Разработка диаграммы классов

Диаграмма классов для курсового проекта приведена в **Приложении А**.

### 3.2 Описание классов и структур

Перечисление **PARAMETR** – параметры поиска.

Описание полей перечисления:

NO\_PARAM = 0 – параметры отсутствуют.

NAME = 1 – поиск по имени.

GROUP = 2 – поиск по группе.

PATH = 3 – путь.

PERM = 4 – поиск по режиму доступа.

REGEX = 5 – регулярное выражение.

SIZE = 6 – поиск по размеру.

USER = 7 – поиск по пользователю.

UID = 8 – поиск по идентификатору пользователя.

QUERY\_FORMAT = 9 – формат запроса.

PARAMETERS\_END = 10 – конец перечисления.

Перечисление **CHECKBOXES** – типы файлов.

Описание полей перечисления:

TYPE\_D = 0 – директории.

TYPE\_F = 1 – файлы.

TYPE\_L = 2 – символические ссылки.

CHECKBOXES\_END – конец перечисления.

Структура **BUFFER\_SETTINGS** – буфер параметров поиска.

Описание полей структуры:

const PARAMETR flag – перечисление параметров поиска.

const char \*ui\_name – указатель на отображаемую строку.

char \*field\_buffer – строка для хранения пользовательского ввода параметра поиска.

Структура **BUFFER\_CHECKBOXES** – буфер типов файлов.

Описание полей структуры:

const CHECKBOXES flag – перечисление типов файлов.

const char \*ui\_name – указатель на отображаемую строку.  
bool \*checked – выбран ли определённый тип файлов.

Структура **SCREEN\_SIZE** – размер экрана.

Описание полей структуры:

int max\_x – максимальное значение x.

int max\_y – максимальное значение y.

Структура **TOOLBAR** – панель инструментов.

Описание полей структуры:

const char\* name – указатель на строку, описывающую что выполняет клавиша.

const char\* key\_name – указатель на строку, описывающую название клавиши.

Структура **KEY\_HANDLER** – обработчик клавиш.

Описание полей структуры:

control\_key key - клавиша.

event\_handler handler – функция-обработчик нажатия клавиши.

Массив **FLAG\_STR\_NAME[]** – флаги для команды find.

Описание элементов массива:

«-name» – флаг для поиска по имени.

«-group» – флаг для поиска по группе.

«-path» – флаг указания по пути.

«-perm» – флаг для поиска по правам доступа.

«-regex» – флаг для поиска с использованием регулярных выражений.

«-size» – флаг для поиска по размеру.

«-user» – флаг для поиска по пользователю.

«-type» – флаг для поиска по типу файла.

Массив **CHECKBOXES\_TOKENS[]** – параметры для флага -type.

Описание элементов массива:

«d» – параметр для поиска директорий.

«f» – параметр для поиска файлов.

«l» – параметр для поиска символических ссылок.

Массив **TOKENS[]** – названия флагов для составления запросов.

Описание элементов массива:

«{name}» – для поиска по имени.

«{group}» – для поиска по группе.

«{path}» – для пути.

«{perm}» – для поиска по правам доступа.  
«{regex}» – для поиска с использованием регулярных выражений.  
«{size}» – для поиска по размеру.  
«{user}» – для поиска по пользователю.  
«{uid}» – для поиска по пользовательскому идентификатору.

### 3.3 Описание основных функций программы

`void write_settings()` – записать параметры в файл.  
`void read_settings()` – считать параметры из файла.  
`void create_exec_str(char* buf, char* path, char* query)` – создать строку вызова `find`.  
`FILE *get_query_result_file(char* path)` – получить результирующий файл.  
`static void render_about_window()` – обработчик отрисовки окна «О программе».  
`static void on_about_resize_handler()` – обработчик события изменения размера окна.  
`static void on_about_exit_handler()` – обработчик события выхода в основное окно.  
`static void about_refresher_handler()` – порядок отрисовки окна «О программе».  
`void about()` – точка входа в обработчик событий окна «О программе».  
`static void init_form()` – инициализация форм.  
`static void refresher_handler()` – порядок отрисовки.  
`static void render_main_window()` – отрисовать основное окно.  
`void render_main_window_gui()` – основная точка входа в цикл событий программы.  
`static void on_settings_handler()` – обработчик события перехода к окну параметров.  
`static void on_about_handler()` – обработчик события перехода к окну «О программе».  
`static void settings_refresher_handler()` – порядок отрисовки окна параметров.  
`static void render_settings()` – отрисовать окно параметров.  
`void settings()` – точка входа в цикл событий окна параметров.  
`void process_input(const char input[256])` – парсер, составляющий запрос из пользовательского формата и ввода.  
`void default_key_handler(const key_handler* control_key_handlers, size_t size)` – основной обработчик событий клавиатуры.

## 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

### 4.1 Разработка схем алгоритмов

Функция **void render\_main\_window\_gui()** – основная точка входа в цикл событий программы. Схема данного алгоритма показана в **Приложении Б**

Функция **void read\_settings()** – считывание параметров настроек из файла. Схема данного алгоритма показана в **Приложении В**

### 4.2 Разработка алгоритмов

Алгоритм по шагам функции составления команды «find» **void create\_exec\_str(char\* buf, char\* path, char\* query)**

1. Начало алгоритма.

2. Входные данные:

char\* buf – буфер для хранения и возвращения запроса.

char\* path – путь для поиска.

char\* query – пользовательский формат запроса.

Промежуточные данные:

char parser\_buf[PATH\_MAX] = "" – буфер для получения результата составления запроса.

bool checkbox\_flag = false – выбран ли определённый тип файлов.

static char parser\_query[PATH\_MAX] – массив символов для запуска парсера.

FILE \*fpipe – указатель на файловый поток.

3. Скопировать «find» в буфер, strcpy(buf, "find"), и «./parser» в parser\_query, strcpy(parser\_query, "./parser").

4. Если путь указан (path != NULL), то присоединить его к buf.

5. Цикл от i = 0 до CHECKBOXES\_QUANTITY – для считывания флагов типов файлов.

6. Если флаг указан, checkbox[i] == true и !checkbox\_flag, то установить флаг checkbox\_flag = true и присоединить «-type» и «,» к buf, strcat(buf, "-type") и strcat(buf, ","). Иначе – шаг 7.

7. Добавить флаг типа файлов, strcat(buf, checkboxes\_tokens[i]).

8. Конец цикла i.

9. Если checkbox\_flag == true, то удалить запятую, buf[strlen(buf) - 1] = '\0'.

10. Если query указан (query != NULL), то перейти к шагу 11, иначе перейти к шагу 15.
11. Присоединить к буферу пробел, а к parser\_query – query.
12. Если файловый поток создаётся неуспешно (0 == (fpipe = (FILE \*)popen(parser\_query, "r"))), то сообщить об ошибке, иначе считать строку из потока в parser\_buf.
13. Присоединить parser\_buf к buf.
14. Закрыть файловый поток.
15. Присоединить « 2>/dev/null» к buf для перенаправления вывода ошибок в системный файл «/dev/null».
16. Конец алгоритма.

Алгоритм по шагам функции составления и вызова команды «find» и получения результата **FILE\* get\_query\_result\_file(char\* path)**

1. Начало алгоритма.
2. Входные данные:  
char\* path – путь для поиска.  
Промежуточные данные:  
char command\_buffer[PATH\_MAX] – буфер для хранения запроса.  
char query[PATH\_MAX] – пользовательский формат запроса.  
char buffer[256] – вспомогательный буфер.  
size\_t current\_pos – текущая позиция.  
size\_t prev\_pos – предыдущая позиция.  
Выходные данные:  
FILE\* fpipe – указатель на файловый поток.
3. Скопировать «'» в query, strcpy(query, «'»).
4. Бесконечный цикл while(1).
5. Если текущий символ пользовательского формата запроса, preferences[get\_index\_by\_param(QUERY\_FORMAT)][current\_pos], равняется «&» или «|» или «\0», то перейти к шагу 6, иначе – шаг 13.
6. Скопировать часть пользовательского формата запроса от prev\_pos до current\_pos.
7. Присоединить buffer к query и « » к query.
8. Цикл от i = 0 до ARRAY\_SIZE(tokens) – для считывания значений параметров поиска.
9. Если подстрока tokens[i] есть в строке buffer, strstr(buffer, tokens[i]), то перейти к шагу 10, иначе – шаг 8.

10. Присоединить preferences[i] к query, strcat(query, preferences[i]), и присоединить « » к query, выйти из цикла, break.
11. Конец цикла i.
12. Если текущий символ равен «\0», preferences[get\_index\_by\_param(QUERY\_FORMAT)][current\_pos] == '\0', то закончить бесконечный цикл while, break, иначе – текущую позицию сделать предыдущей, prev\_pos = current\_pos. Перейти к шагу 8.
13. Перейти на следующий символ, current\_pos++.
14. Присоединить «'» к query
15. Вызов функции create\_exeс\_str(char\* buf, char\* path, char\* query) для составления и получения команды «find».
16. Если файловый поток создаётся неуспешно (0 == (fpipe = (FILE \*)popen(command\_buffer, "r"))), то сообщить об ошибке, иначе вернуть fpipe.
17. Конец алгоритма.

### **4.3 Код программы**

Полный код программы представлен в **Приложении Г**.

## 5 РЕЗУЛЬТАТ РАБОТЫ

При запуске приложения пользователя встречает основное графическое окно. На панели расположено название файлов и их расширение. А также присутствует поле для ввода пути для поиска файлов. На рисунке 5.1 показан графический интерфейс основного окна.

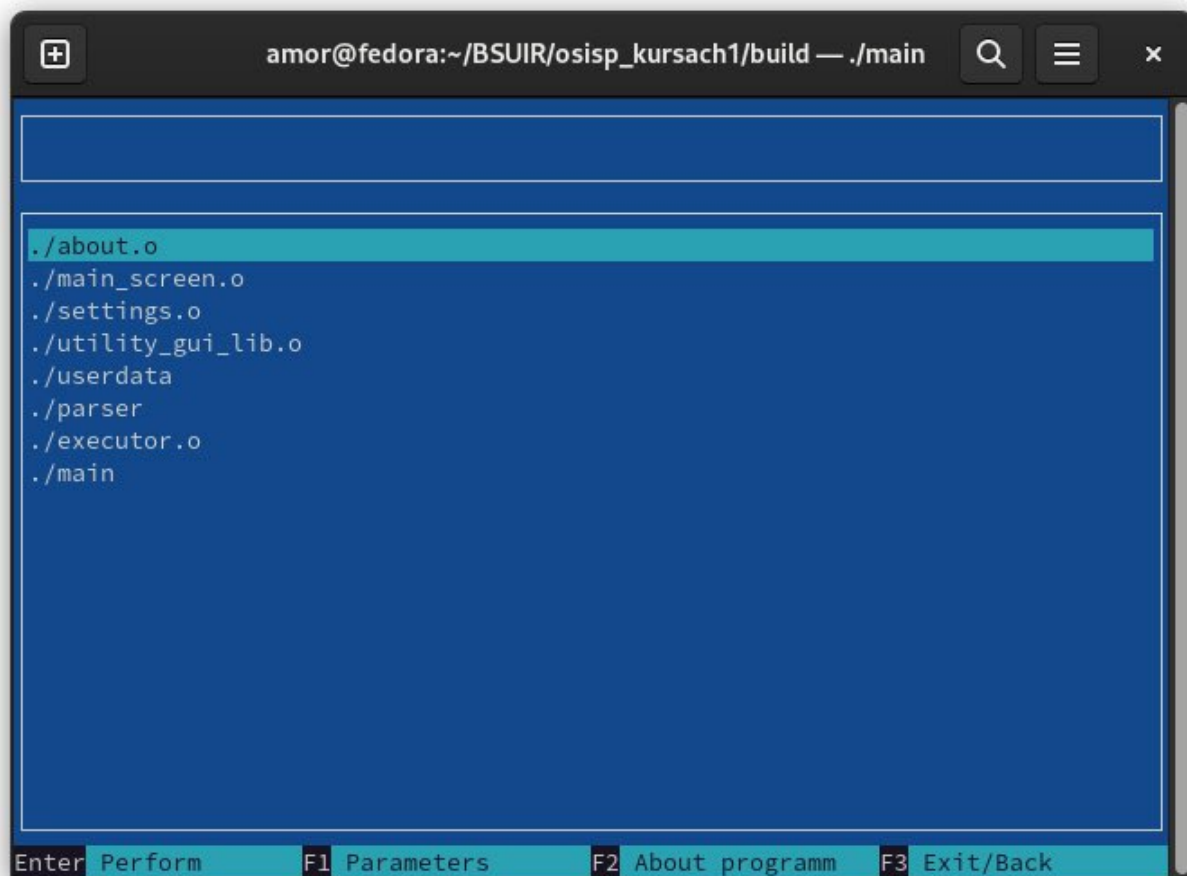


Рисунок 5.1 – графический интерфейс основного окна приложения

При вводе необходимого пути для поиска и нажатии на клавишу «Enter» составляется запрос, начинается поиск файлов по определённому пути, обновление панели и вывод найденных файлов. На рисунке 5.2 показан графический интерфейс основного окна при поиске по полному пути.



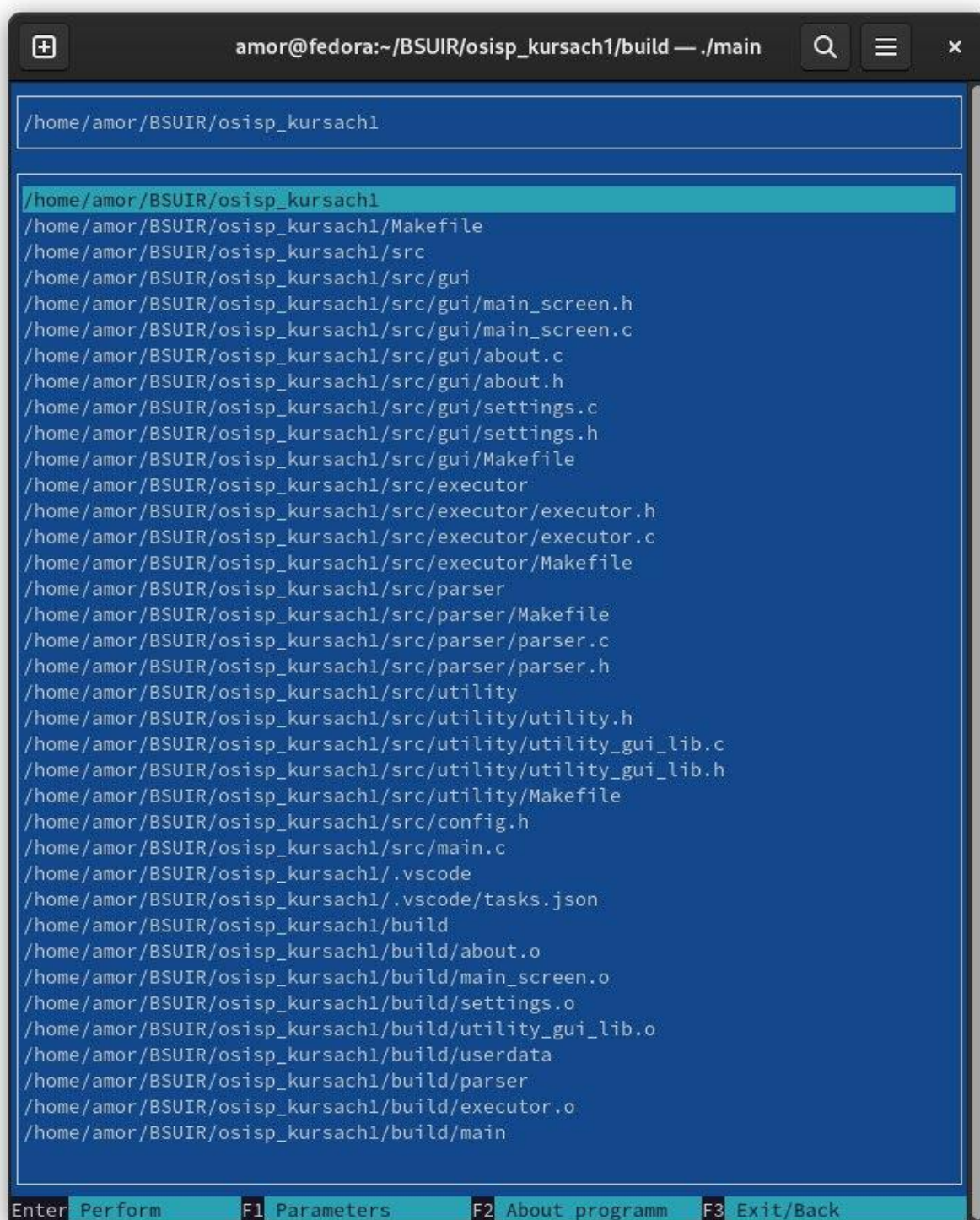


Рисунок 5.2 – графический интерфейс основного окна приложения при поиске по полному пути

Также доступна возможность поиска по пути относительно начальной директории. На рисунке 5.3 представлен графический интерфейс основного окна при поиске по относительному пути.

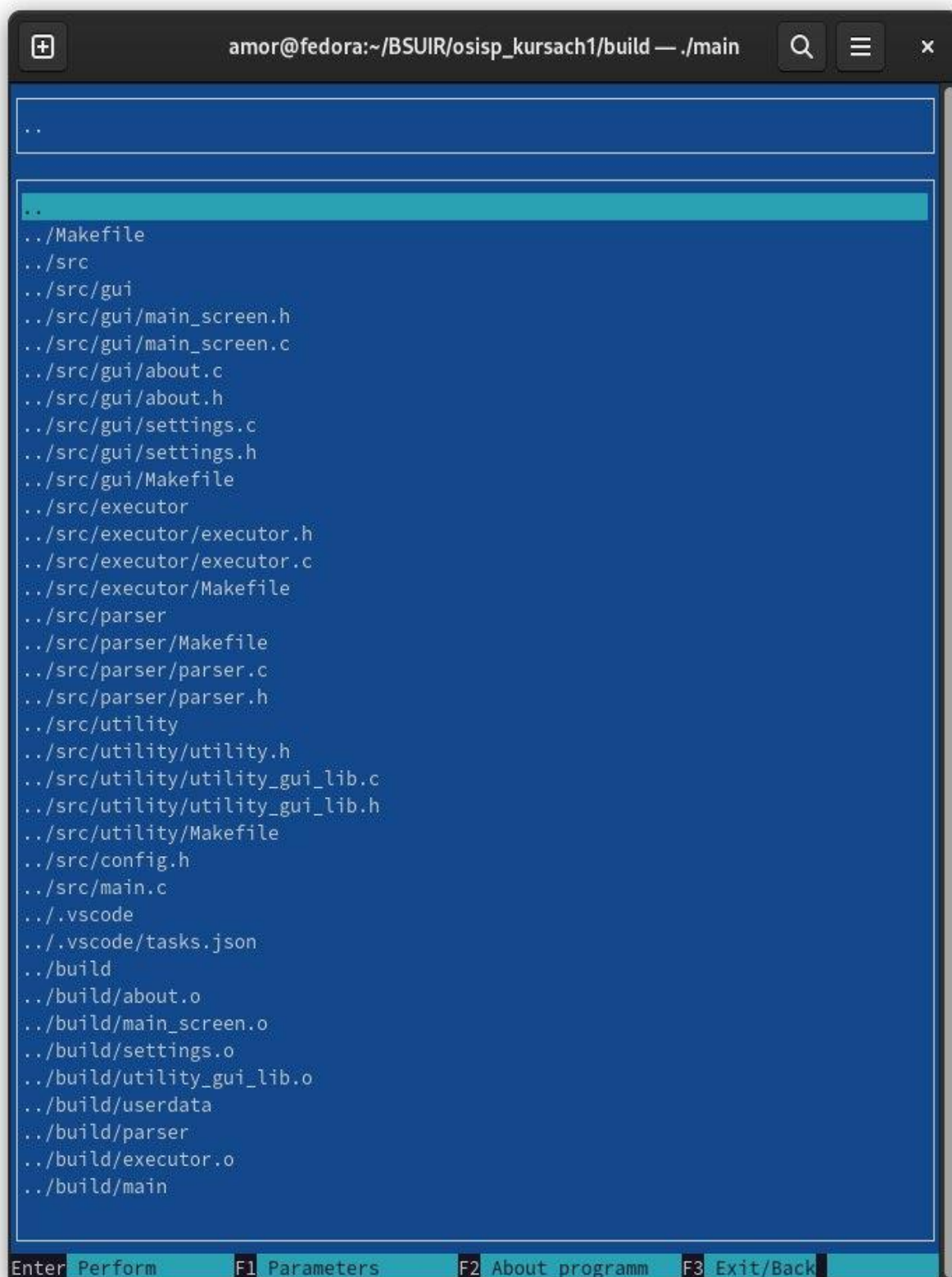


Рисунок 5.3 – графический интерфейс основного окна при поиске по относительному пути

При нажатии клавиши «F1» открывается окно настроек поиска. Доступен поиск по следующим параметрам: имени, группе, пользователю,

регулярному выражению, правам доступа, размеру файла. Реализовано отдельное поле для составления сложных запросов для поиска по нескольким параметрам с использованием логических операций И, ИЛИ. НЕ. Также доступен выбор поиска по типу: файлы, каталоги, символические ссылки. По умолчанию выполняется поиск по всем указанным типам. На рисунке 5.4 представлен графический интерфейс окна настроек поиска при незадаанных параметрах.

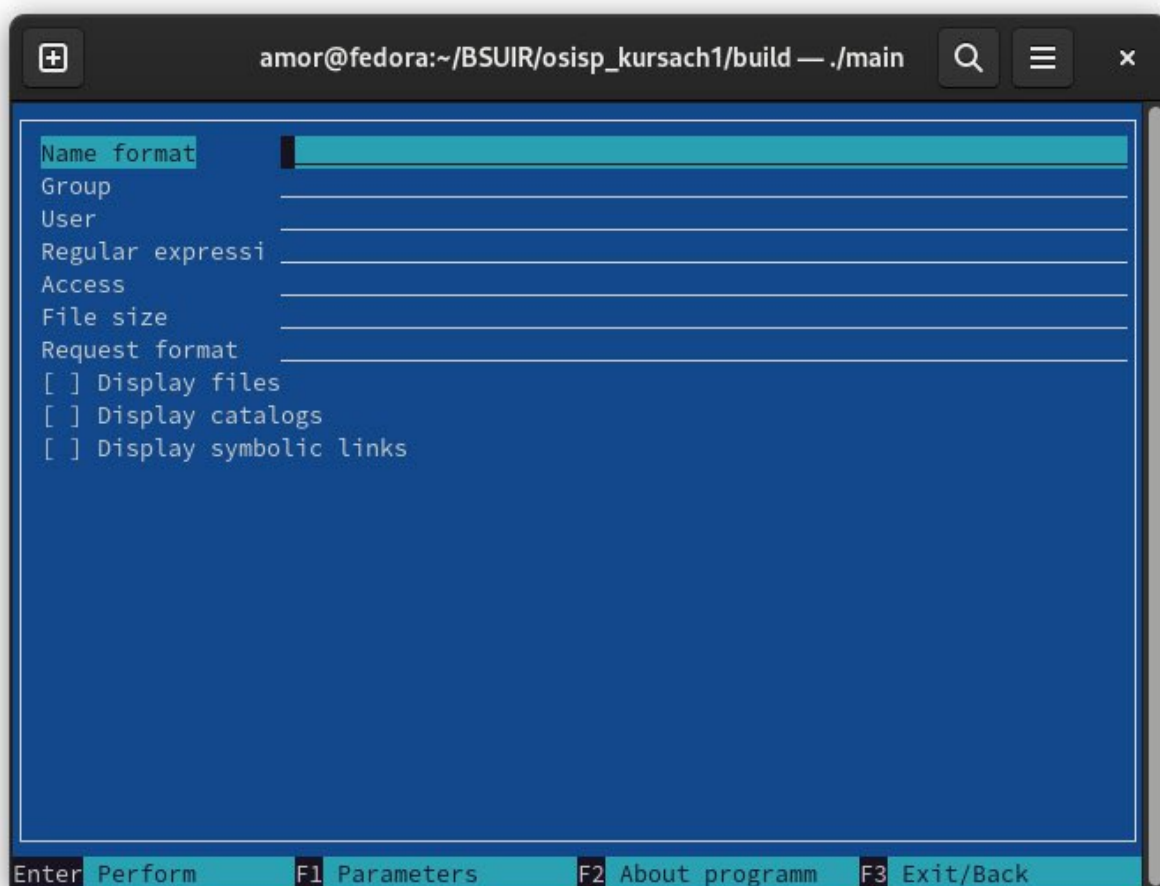


Рисунок 5.4 – графический интерфейс окна настроек поиска при незадаанных параметрах

Результатом поиска без параметров будут все типы: файлы, каталоги и символические ссылки. На рисунке 5.5 представлен графический интерфейс основного окна при незадаанных параметрах поиска и поиске по относительному пути.

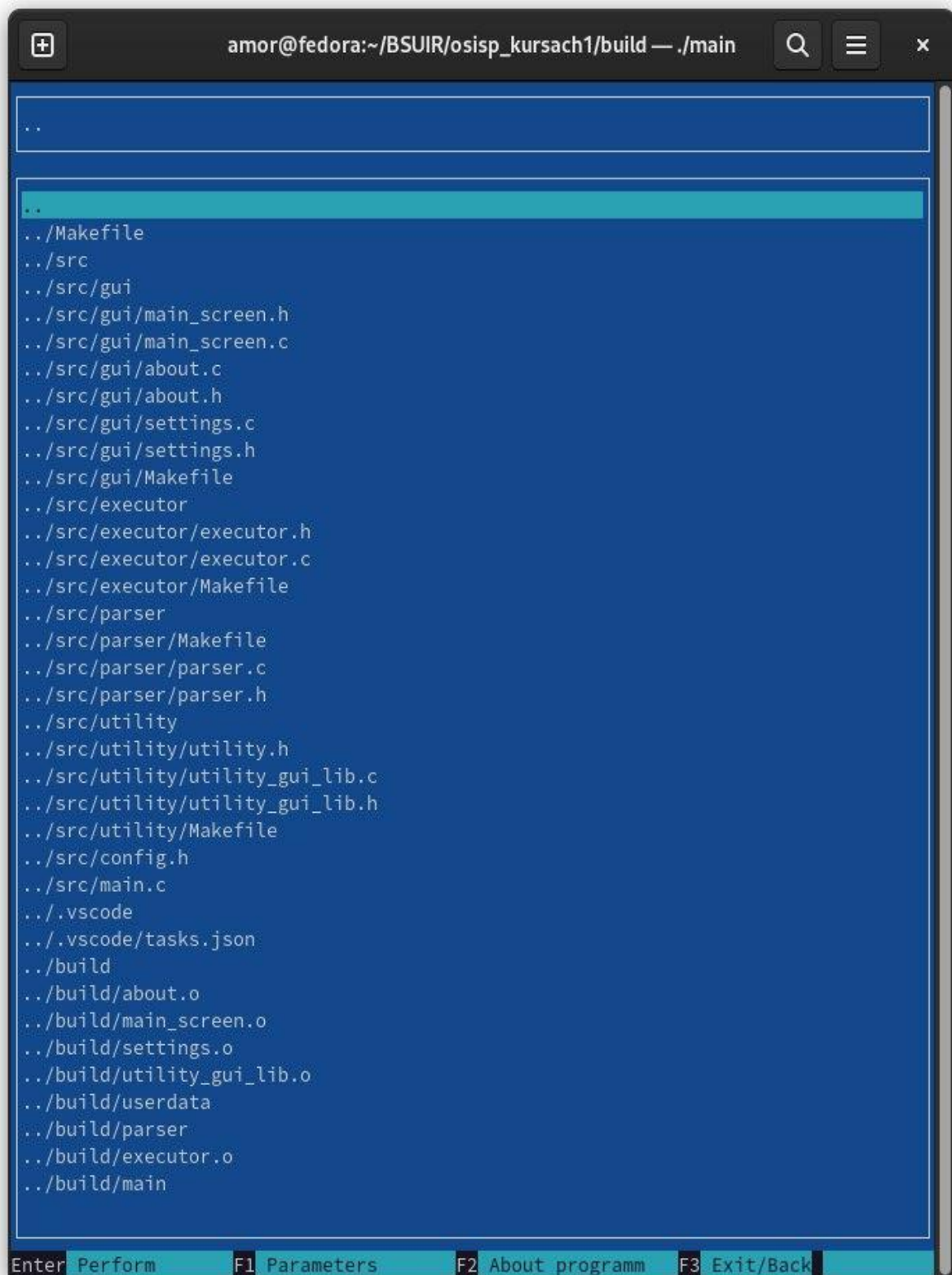


Рисунок 5.5 – графический интерфейс основного окна при незадаанных параметрах поиска и поиске по относительному пути

При поиске файлов по имени, обратному введённому, окно настроек будет выглядеть следующим образом. На рисунке 5.6 представлен

графический интерфейс окна настроек при поиске файлов по имени, обратному указанному.

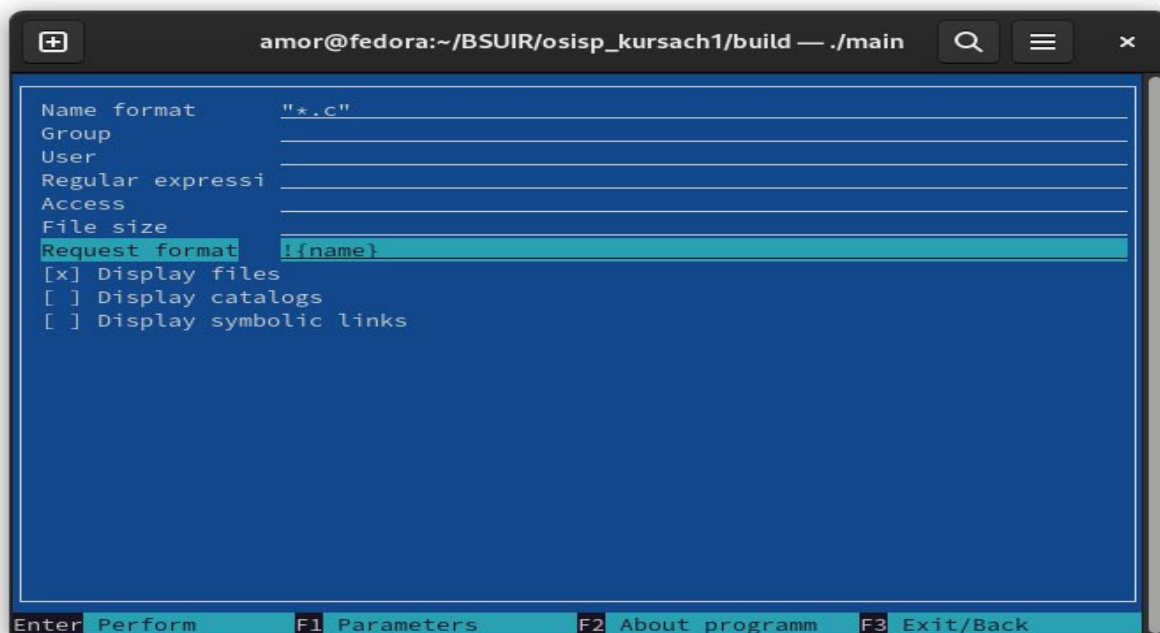


Рисунок 5.6 – графический интерфейс окна настроек при поиске файлов по имени, обратному указанному

Результатом поиска по параметрам, заданным на рисунке 5.6, будут все файлы по указанному пути, расширение которых не «.c». На рисунке 5.7 представлен графический интерфейс основного окна при заданных параметрах поиска.



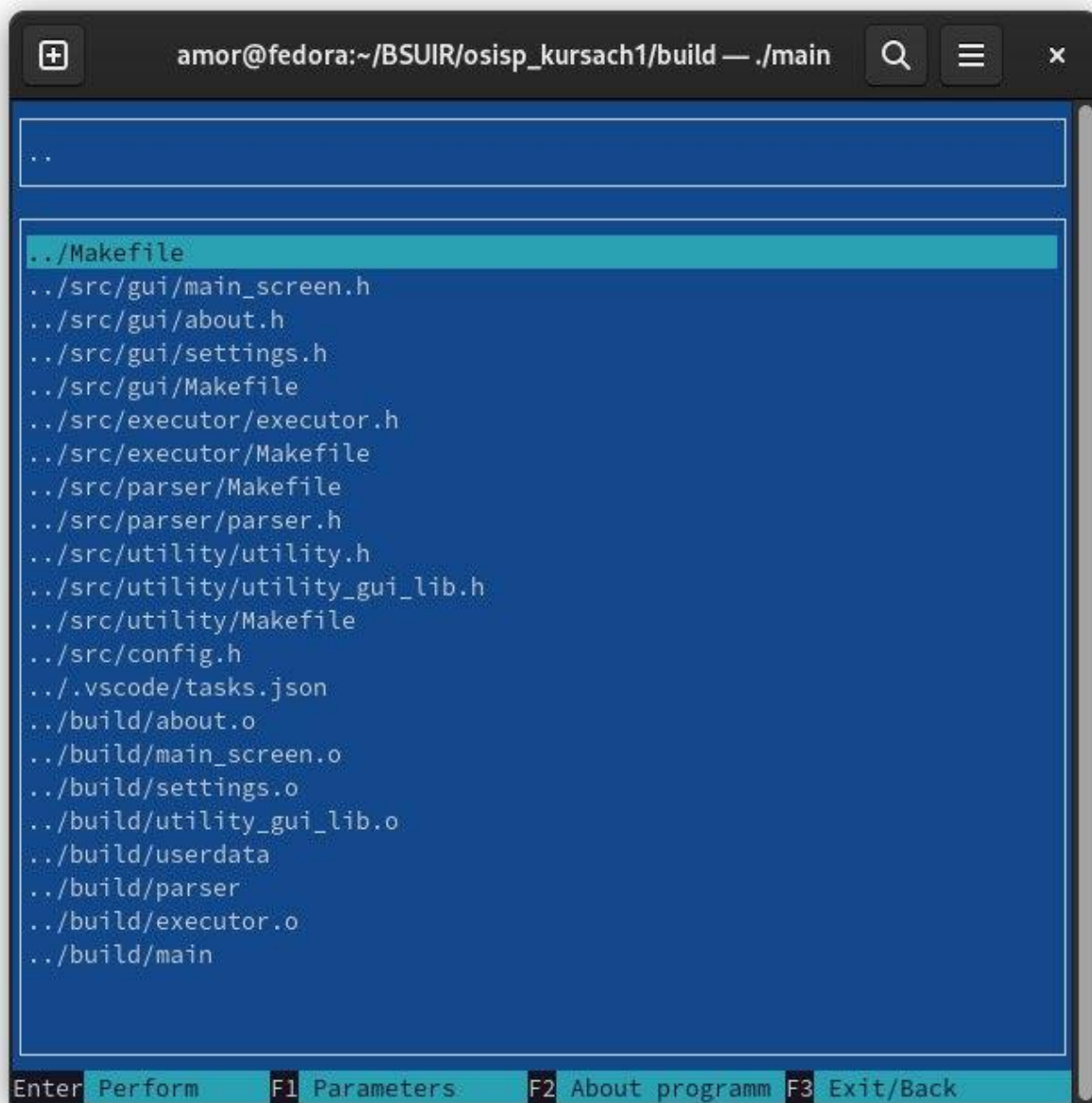


Рисунок 5.7 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по имени и пользователю, окно настроек будет выглядеть следующим образом. На рисунке 5.8 представлен графический интерфейс окна настроек при поиске файлов по имени и пользователю.

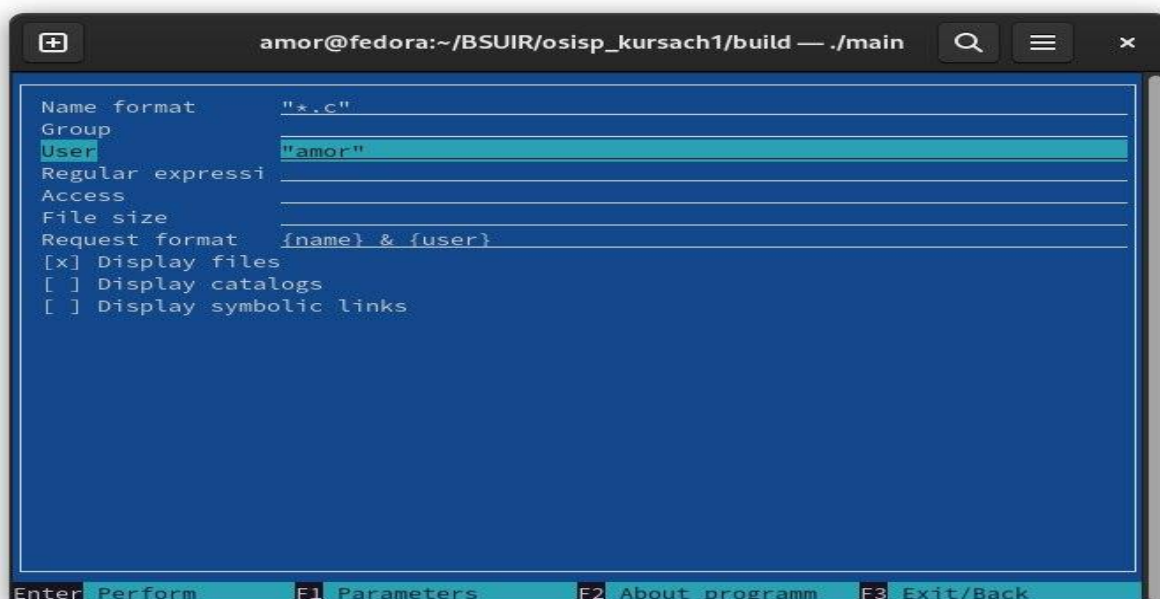


Рисунок 5.8 – графический интерфейс окна настроек при поиске файлов по имени и пользователю

Результатом поиска по параметрам, заданным на рисунке 5.8, будут все файлы по указанному пути, расширение которых «.c» и имя пользователя «amor». На рисунке 5.9 представлен графический интерфейс основного окна при заданных параметрах поиска.

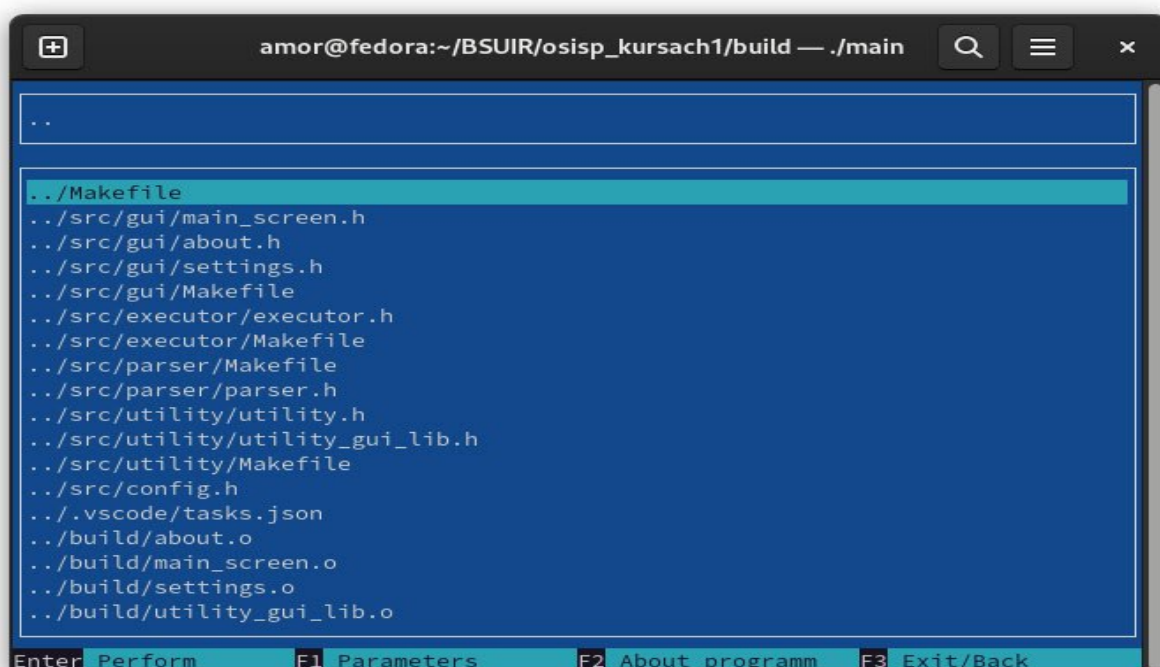


Рисунок 5.9 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по размеру реализована возможность поиска файлов больше или меньше определённого размера. При поиске файлов, размер которых больше 10 килобайт «+10k», окно настроек будет выглядеть следующим образом. Если введены значения в другие поля, то пока не составлен пользовательский запрос, они не будут учитываться при поиске. На рисунке 5.10 представлен графический интерфейс окна настроек при поиске файлов по размеру.

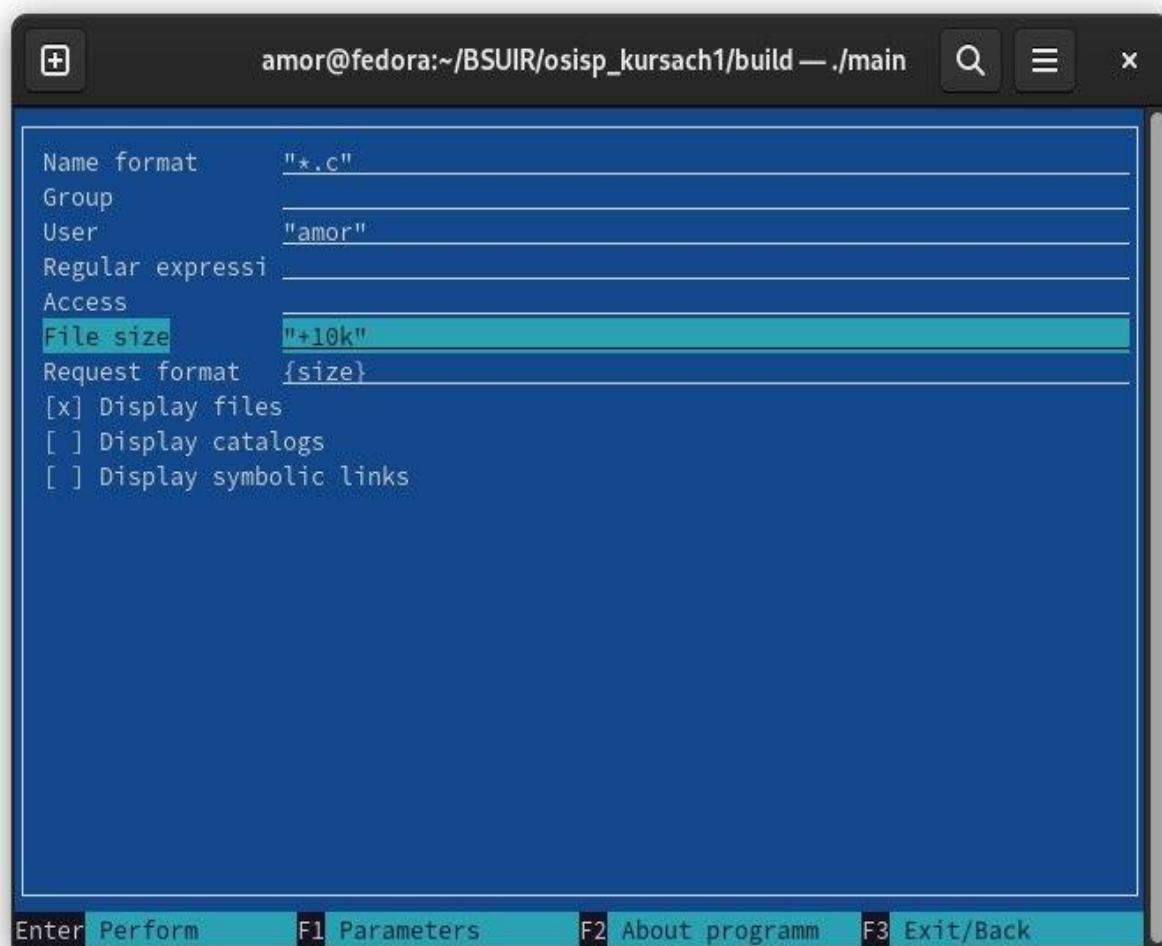


Рисунок 5.10 – графический интерфейс окна настроек при поиске файлов по размеру

Результатом поиска по параметрам, заданным на рисунке 5.10, будут все файлы в начальной директории, размеры которых превышает 10 килобайт. На рисунке 5.11 представлен графический интерфейс основного окна при заданных параметрах поиска.



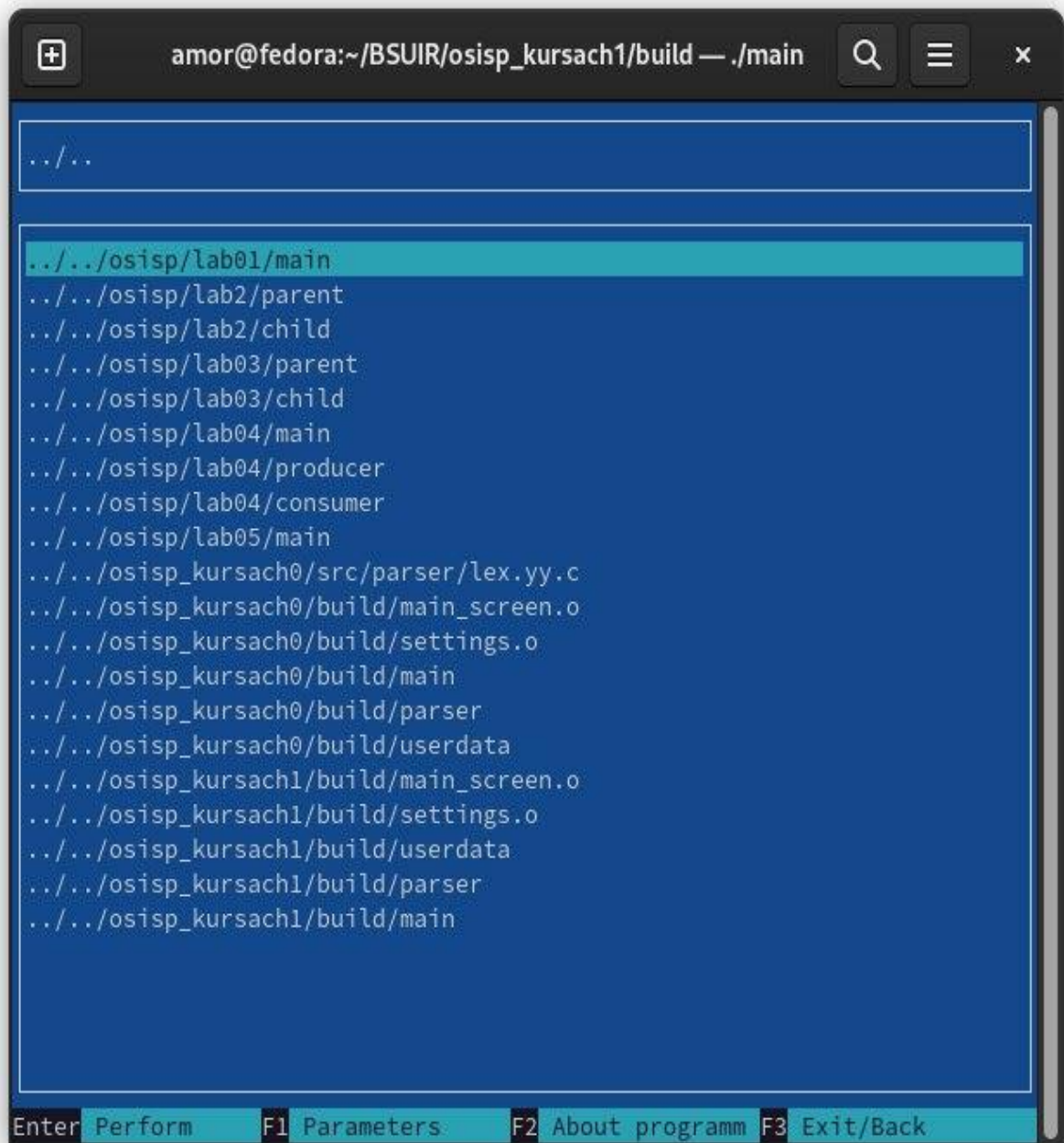


Рисунок 5.11 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по праву доступа 644 – владелец файла может читать и изменять файл, а остальные пользователи (в том числе и группа) – только читать. Окно параметров будет выглядеть следующим образом. На рисунке 5.12 представлен графический интерфейс окна настроек при поиске файлов по праву доступа 644.

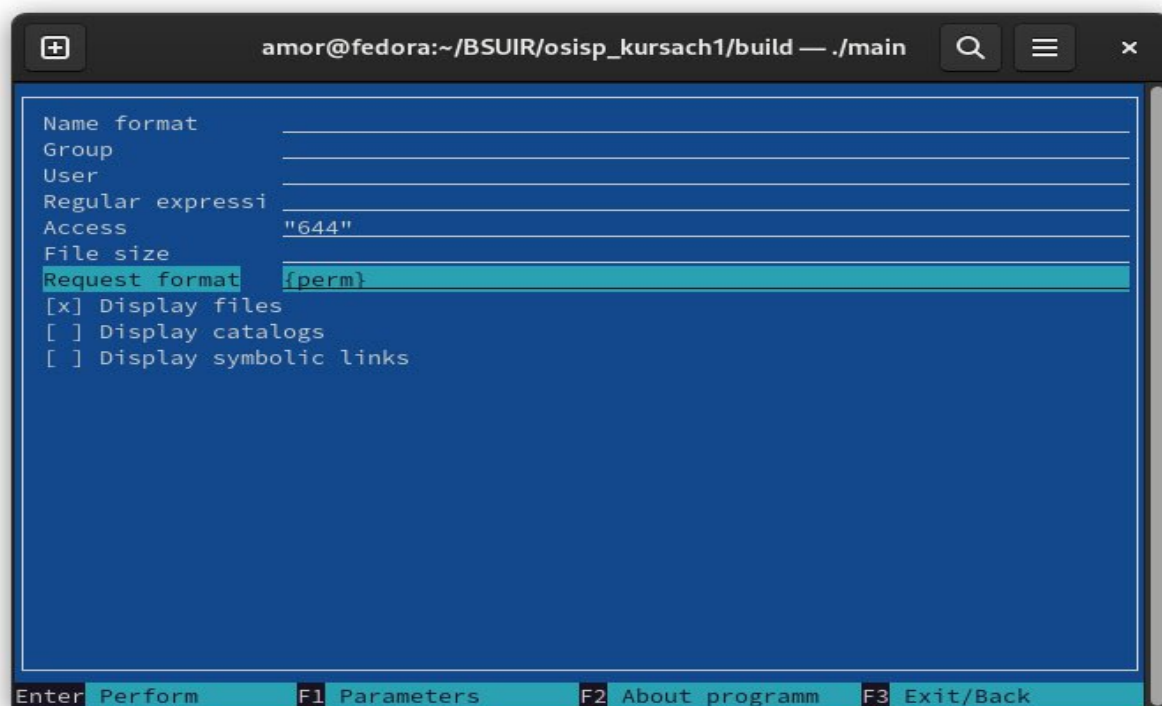


Рисунок 5.12 – графический интерфейс окна настроек при поиске файлов по праву доступа 644

Результатом поиска по параметрам, заданным на рисунке 5.12, будут все файлы в начальной директории, право доступа которых является 644. На рисунке 5.13 представлен графический интерфейс основного окна при заданных параметрах поиска.

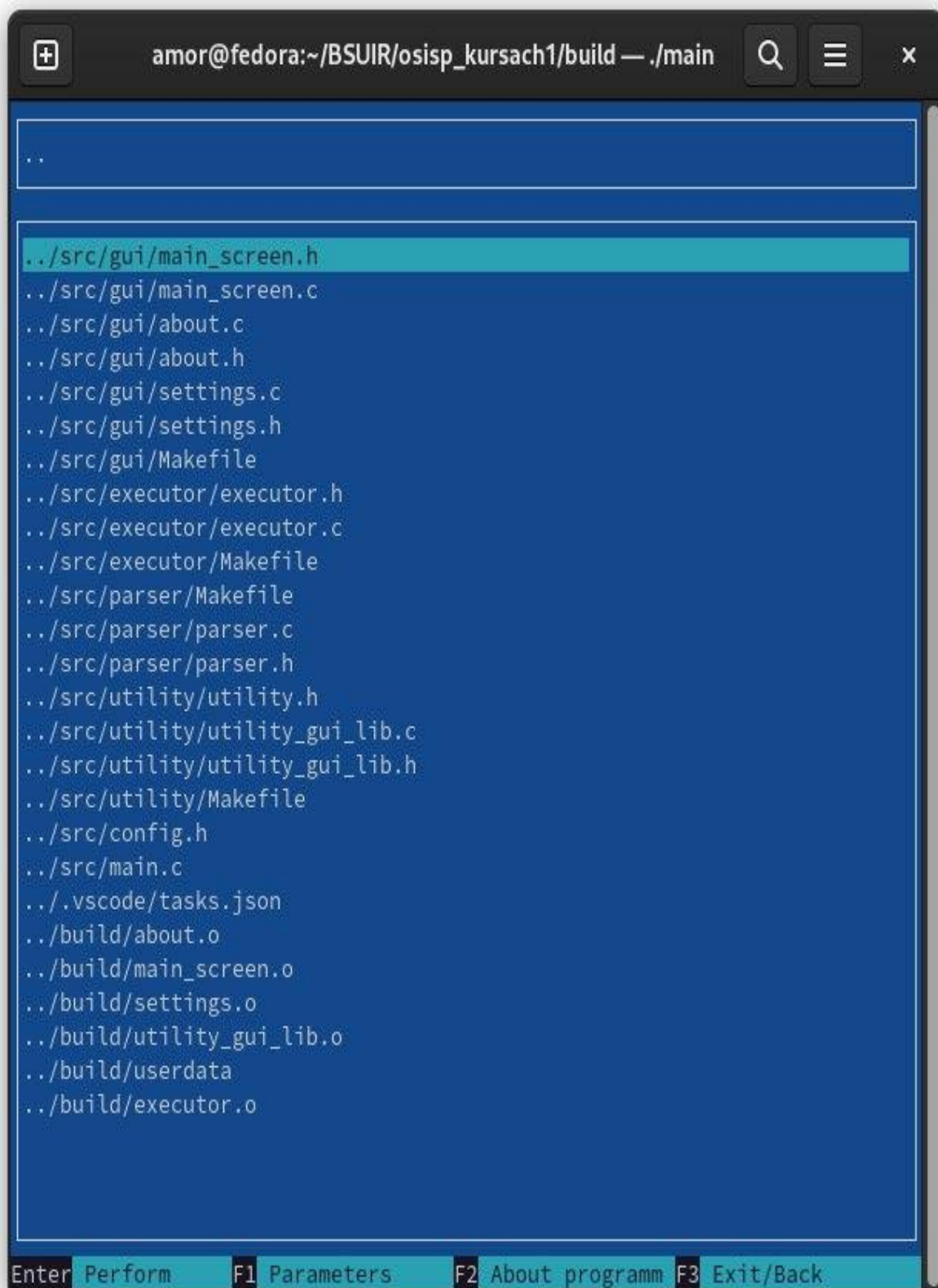


Рисунок 5.13 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по регулярному выражению, например необходимо найти все файлы с расширением «.c», окно параметров будет выглядеть следующим образом. На рисунке 5.14 представлен графический интерфейс окна настроек при поиске файлов с расширением «.c» по регулярному выражению.

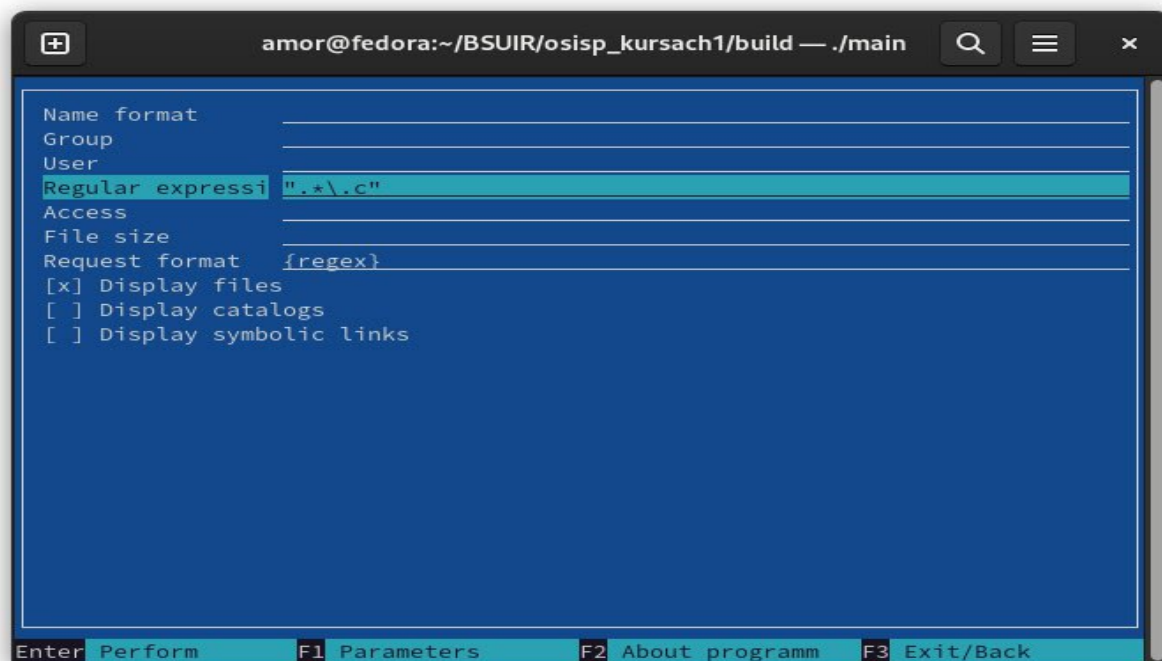


Рисунок 5.14 – графический интерфейс окна настроек при поиске файлов с расширением «.c» по регулярному выражению

Результатом поиска по параметрам, заданным на рисунке 5.14, будут все файлы с расширением «.c» по указанному пути. На рисунке 5.15 представлен графический интерфейс основного окна при заданных параметрах поиска.

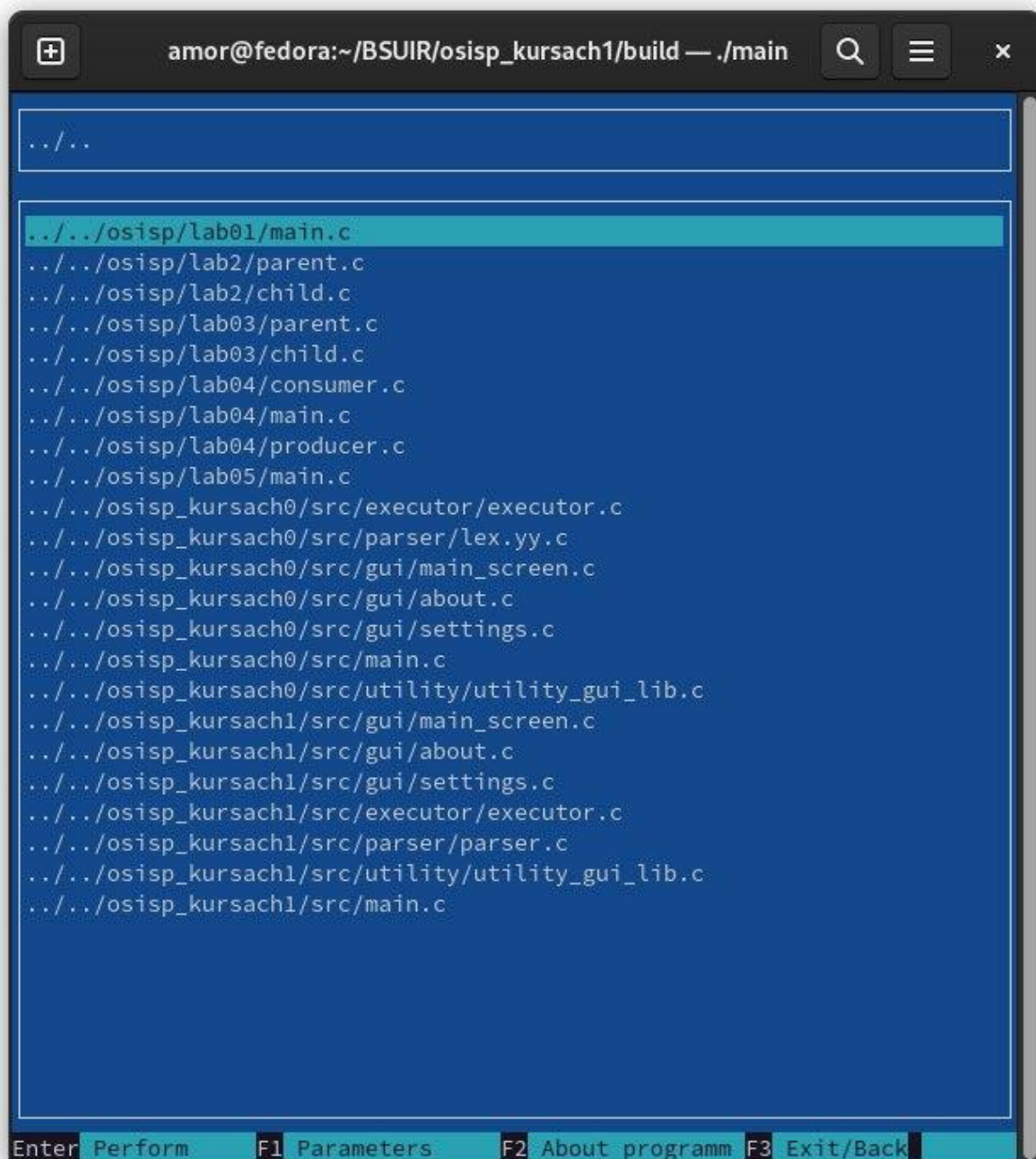


Рисунок 5.15 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по группе, например попробуем найти файлы по группе, обратной несуществующей, окно параметров будет выглядеть следующим образом. На рисунке 5.16 представлен графический интерфейс окна настроек при поиске файлов по группе, обратной несуществующей.

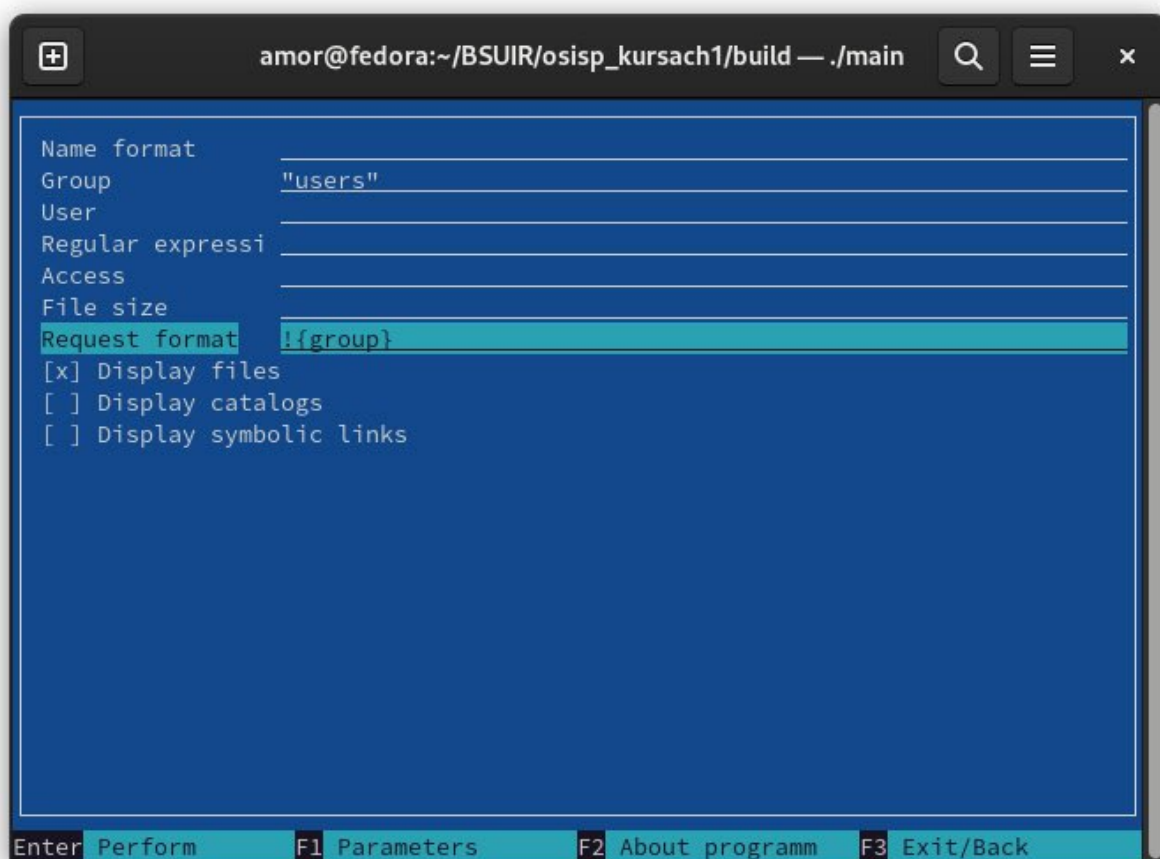


Рисунок 5.16 – графический интерфейс окна настроек при поиске файлов по группе, обратной несуществующей

Результатом поиска по параметрам, заданным на рисунке 5.16, будут все файлы по указанному пути. На рисунке 5.17 представлен графический интерфейс основного окна при заданных параметрах поиска.



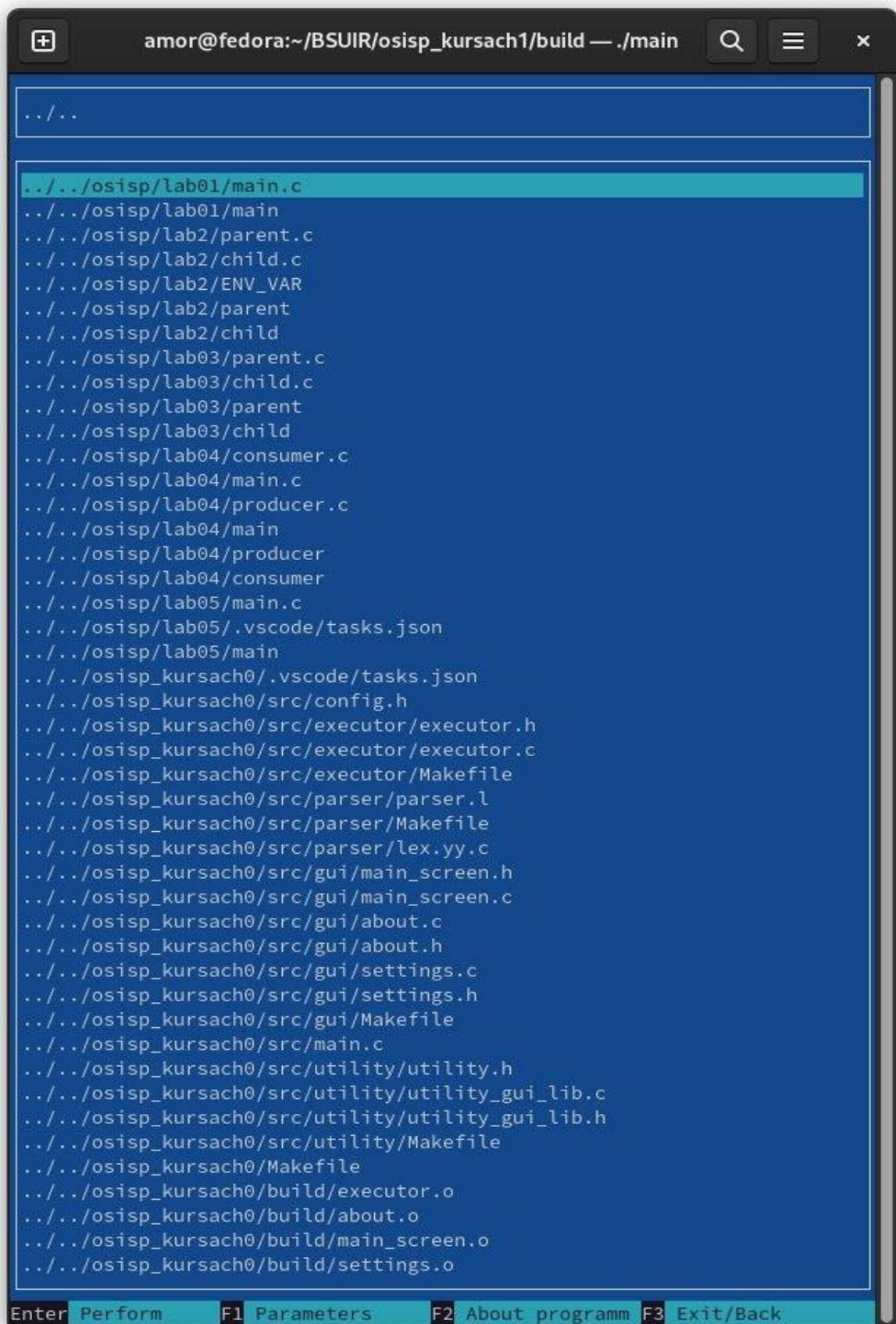


Рисунок 5.17 – графический интерфейс основного окна при заданных параметрах поиска

При поиске файлов по имени или размеру более 10 килобайт окно параметров будет выглядеть следующим образом. На рисунке 5.18 представлен графический интерфейс окна настроек при поиске файлов по имени или размеру более 10 килобайт.

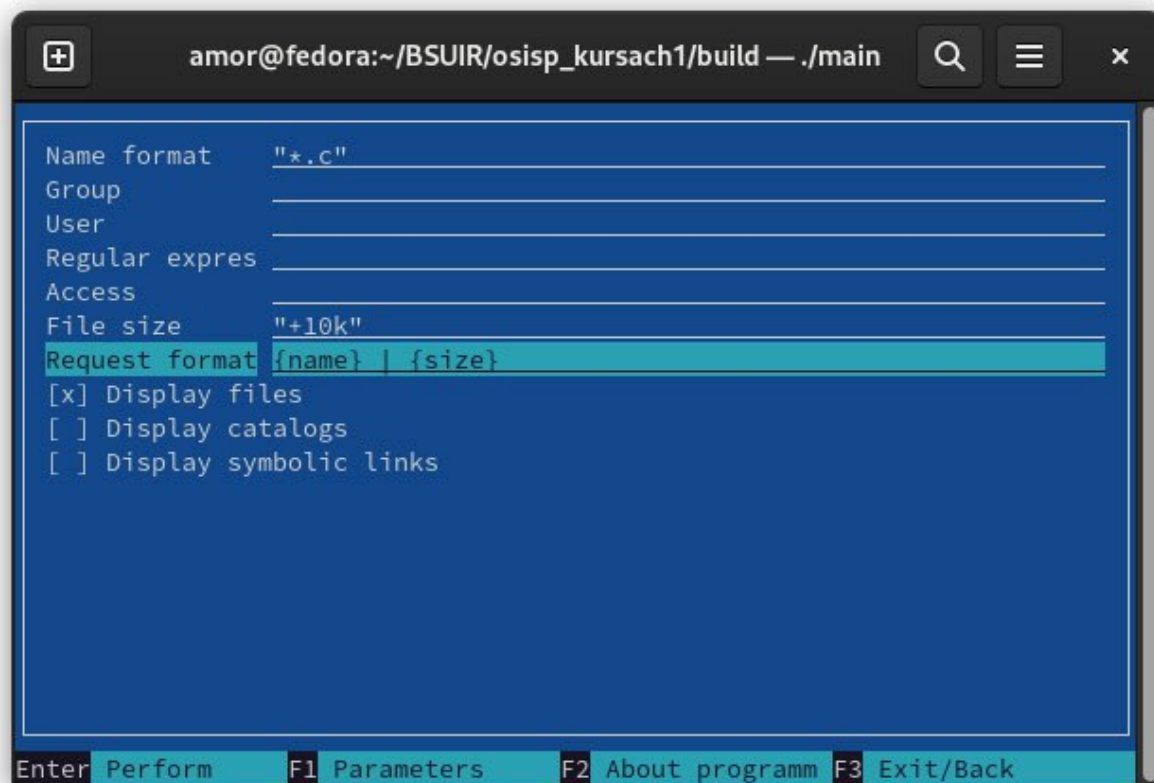


Рисунок 5.18 – графический интерфейс окна настроек при поиске файлов по имени или размеру более 10 килобайт

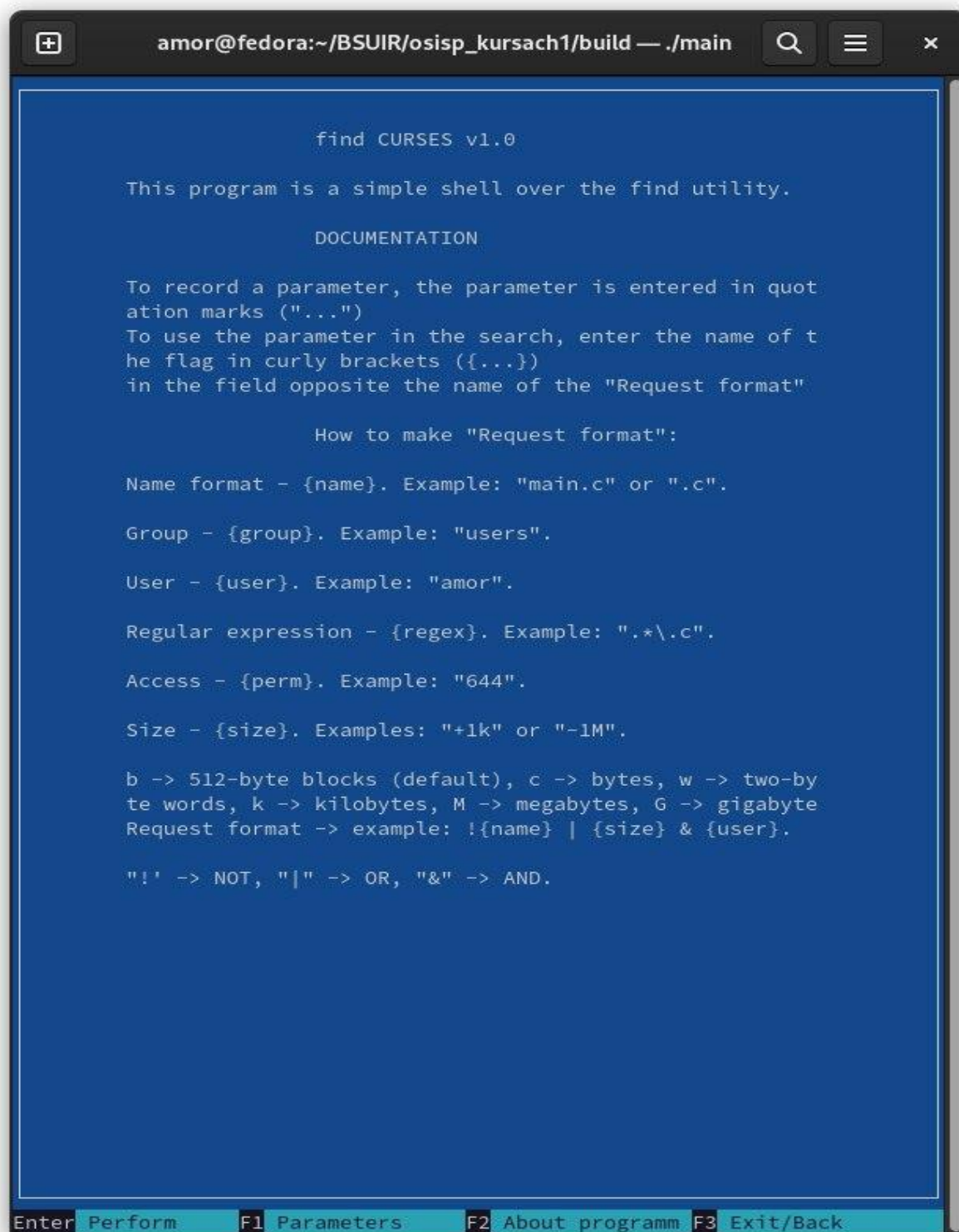
Результатом поиска по параметрам, заданным на рисунке 5.18, будут все файлы с расширением «.c» или размером более 10 килобайт по указанному пути. На рисунке 5.19 представлен графический интерфейс основного окна при заданных параметрах поиска.





Рисунок 5.19 – графический интерфейс основного окна при заданных параметрах поиска

При нажатии «F2» открывается окно «О программе», содержащее информацию о программе и как пользователю ею быстро и правильно управлять. Предоставлена информация о названии, версии программы, инструкция по составлению запросу, примеры параметров. На рисунке 5.20 представлен графический интерфейс окна «О программе».



```
find CURSES v1.0

This program is a simple shell over the find utility.

DOCUMENTATION

To record a parameter, the parameter is entered in quotation marks ("...")
To use the parameter in the search, enter the name of the flag in curly brackets ({...})
in the field opposite the name of the "Request format"

How to make "Request format":

Name format - {name}. Example: "main.c" or ".c".
Group - {group}. Example: "users".
User - {user}. Example: "amor".
Regular expression - {regex}. Example: ".*\.c".
Access - {perm}. Example: "644".
Size - {size}. Examples: "+1k" or "-1M".

b -> 512-byte blocks (default), c -> bytes, w -> two-byte words, k -> kilobytes, M -> megabytes, G -> gigabyte
Request format -> example: !{name} | {size} & {user}.

"!" -> NOT, "|" -> OR, "&" -> AND.
```

Enter Perform F1 Parameters F2 About program F3 Exit/Back

Рисунок 5.20 – графический интерфейс окна «О программе»

При нажатии «F3» программа завершается, сохраняя последние настройки поиска в файл «./userdata», и возвращается привычное окно терминала. На рисунке 5.21 представлен результат нажатия клавиши «F3» в основном окне.

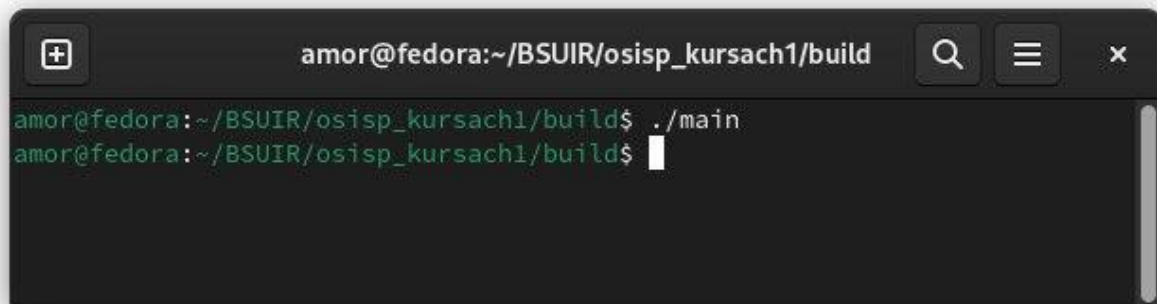


Рисунок 5.21 – результат нажатия клавиши «F3» в основном окне

## **ЗАКЛЮЧЕНИЕ**

В результате курсовой работы была создана функциональная оболочка, позволяющая пользователю выполнять поиск файлов и каталогов с помощью различных критериев, таких как имя файла, тип, размер и другие атрибуты. Интерфейс оболочки разработан с учетом удобства использования и эффективности выполнения поиска.

В качестве дальнейших шагов развития проекта можно рассмотреть добавление дополнительных функций и возможностей, таких как поддержка оставшихся критериев поиска, улучшение интерфейса пользователя, оптимизация производительности.

Рекомендуемые системные требования:

- ОС – Linux Fedora
- Среда разработки – Microsoft Visual Studio 2022
- Занимаемая память процессора – 50-100 мб

## СПИСОК ЛИТЕРАТУРЫ

- [1] Питер В. Хрисистомадулу - "Beginning GTK+ and GNOME" - 2008.
- [2] Андре Стайт - "Foundations of GTK+ Development" - 2007.
- [3] William E. Shotts, Jr. - "The Linux Command Line: A Complete Introduction" - 2012.
- [4] Christopher Negus - "Linux Bible" - 2020.
- [5] Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley - "UNIX and Linux System Administration Handbook" - 2017.
- [6] Daniel J. Barrett - "Linux Pocket Guide" - 2020.
- [7] Robert Love - "Linux Kernel Development" - 2010.

**ПРИЛОЖЕНИЕ А**  
*(обязательное)*  
Диаграмма классов

## **ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Схема метода

## **ПРИЛОЖЕНИЕ В**

*(обязательное)*

Схема метода



## **ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код программы

**ПРИЛОЖЕНИЕ Д**  
*(обязательное)*  
Ведомость документов