Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интелект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе № 9 по курсу «Операционные системы»

Тема Системный вызов open()

Студент Волков Г. В.

Группа ИУ7-61Б

Оценка (баллы)

Преподаватель Рязанова Н. Ю.

Москва — 2023 г.

1 Системный вызов open()

Системный вызов *open()* открывает файл, указанный в *pathname*. Если файл не существует и указан флаг *O_CREAT*, файл будет создан с правами доступа, указанными в *mode*.

```
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 #include <fcntl.h>
4
5 int open (const char *pathname, int flags);
6 int open (const char *pathname, int flags, mode_t mode);
```

Системный вызов *open()* возвращает дескриптор файла — неотрицательное число, которое затем используется в системных вызовах *read()* , *write()* , *lseek()* , и т.д. для ссылки на открытый файл.

Первый аргумент — имя файла в файловой системе. Второй аргумент — режим открытия файла — один или несколько флагов открытия, объединенных оператором побитового ИЛИ. Могут быть использованны следующие флаги:

- ***O_APPEND*** — файл открывается в режиме добавления: перед каждой операцией записи файловый указатель будет устанавливать в конец файла;
- ***O_ASYNC*** — включить ввод/вывод на основе сигналов (*SIGIO* по умолчанию);
- ***O_CLOEXEC*** — при вызове *exec* файл не будет оставаться открытым;
- ***O_CREAT*** — создать файл, если он не существует;
- ***O_DIRECT*** — попытаться минимизировать эффект от кеширования ввода/вывода в/из этого файла;
- ***O_DIRECTORY*** — вернуть ошибку, если файл не является каталогом;

- ***O_DSYNC*** — операции записи будут выполняться в соответствии с требованиями для целостности данных;
- ***O_EXCL*** — при использовании вместе с *O_CREAT* вызов *open()* вернет ошибку, если файл уже существует;
- ***O_LARGEFILE*** — позволяет открывать файлы, размер которых не может быть представлен типом *off_t*;
- ***O_NOATIME*** — не обновлять время последнего доступа к файлу;
- ***O_NOCITY*** — если файл указывает на терминальное устройство, то оно не станет терминалом управления процесса, даже при его отсутствии;
- ***O_NOFOLLOW*** — вернуть ошибку, если часть пути является символической ссылкой;
- ***O_NONBLOCK*** — если возможно, открыть файл в неблокирующем режиме;
- ***O_PATH*** — получить файловый дескриптор, который может быть использован для индикации расположения файла в файловой системе или для операций на уровне дескриптора;
- ***O_SYNC*** — операции записи будут выполняться в соответствии с требованиями для целостности файла;
- ***O_TMPFILE*** — создать неименованный временный файл;
- ***O_TRUNC*** — если файл существует, его длина будет "урезана" до нуля;

Если вызов *open()* создает новый файл, он будет создан с правами доступа, которые были переданы в *mode*. Для установки значения *mode* определены следующие константы:

- ***S_IRWXU*** — права на чтение, запись, выполнение для пользователя;
- ***S_IRUSR*** — права на чтение для пользователя;

- ***S_IWUSR*** — права на запись для пользователя;
- ***S_IXUSR*** — права на выполнение для пользователя;
- ***S_IRWXG*** — права на чтение, запись, выполнение для группы;
- ***S_IRGRP*** — права на чтение для группы;
- ***S_IWGRP*** — права на запись для группы;
- ***S_IXGRP*** — права на выполнение для группы;
- ***S_IRWXO*** — права на чтение, запись, выполнение для остальных;
- ***S_IROTH*** — права на чтение для остальных;
- ***S_IWOTH*** — права на запись для остальных;
- ***S_IXOTH*** — права на выполнение для остальных;
- ***S_ISUID*** — бит *set-user-ID*;
- ***S_ISGID*** — бит *set-group-ID*;
- ***S_ISVTX*** — ”липкий” бит;

2 Используемые структуры

Листинг 2.1 – Структура open_flags

```
struct open_flags {
    int open_flag;
    umode_t mode;
    int acc_mode;
    int intent;
    int lookup_flags;
};
```

Листинг 2.2 – Структура filename

```
struct filename {
    const char      *name;  /* pointer to actual string */
    const __user char   *uptr;  /* original userland pointer */
    int         refcnt;
    struct audit_names  *aname;
    const char      iname[];
};

struct audit_names {
    struct list_head    list;       /* audit_context->names_list */
    struct filename     *name;
    int         name_len;   /* number of chars to log */
    bool            hidden;     /* don't log this record */
    unsigned long       ino;
    dev_t           dev;
    umode_t         mode;
    kuid_t          uid;
    kgid_t          gid;
    dev_t           rdev;
    u32         osid;
    struct audit_cap_data   fcap;
    unsigned int        fcap_ver;
    unsigned char       type;       /* record type */
    bool            should_free;
};
```

Листинг 2.3 – Структура nameidata

```
struct nameidata {
    struct path path;
    struct qstr last;
    struct path root;
    struct inode    *inode; /* path.dentry.d_inode */
    unsigned int    flags, state;
    unsigned    seq, next_seq, m_seq, r_seq;
    int     last_type;
    unsigned    depth;
    int     total_link_count;
    struct saved {
        struct path link;
        struct delayed_call done;
        const char *name;
        unsigned seq;
    } *stack, internal[EMBEDDED_LEVELS];
    struct filename *name;
    struct nameidata *saved;
    unsigned    root_seq;
    int     dfd;
    kuid_t      dir_uid;
    umode_t     dir_mode;
} __randomize_layout;
```

3 Схема выполнения системного вызова open()

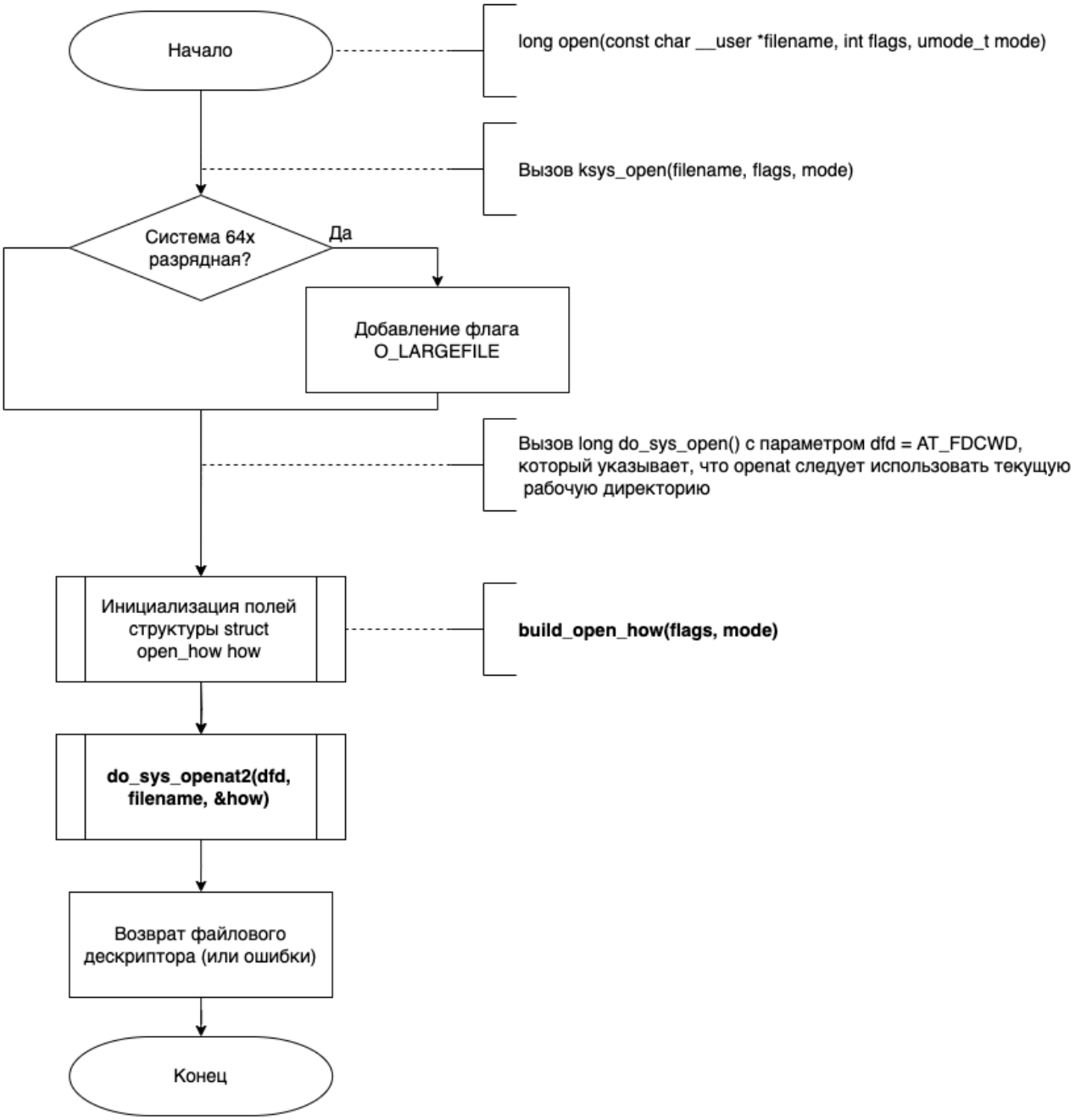

Рисунок 3.1 – Схема алгоритма функции open()

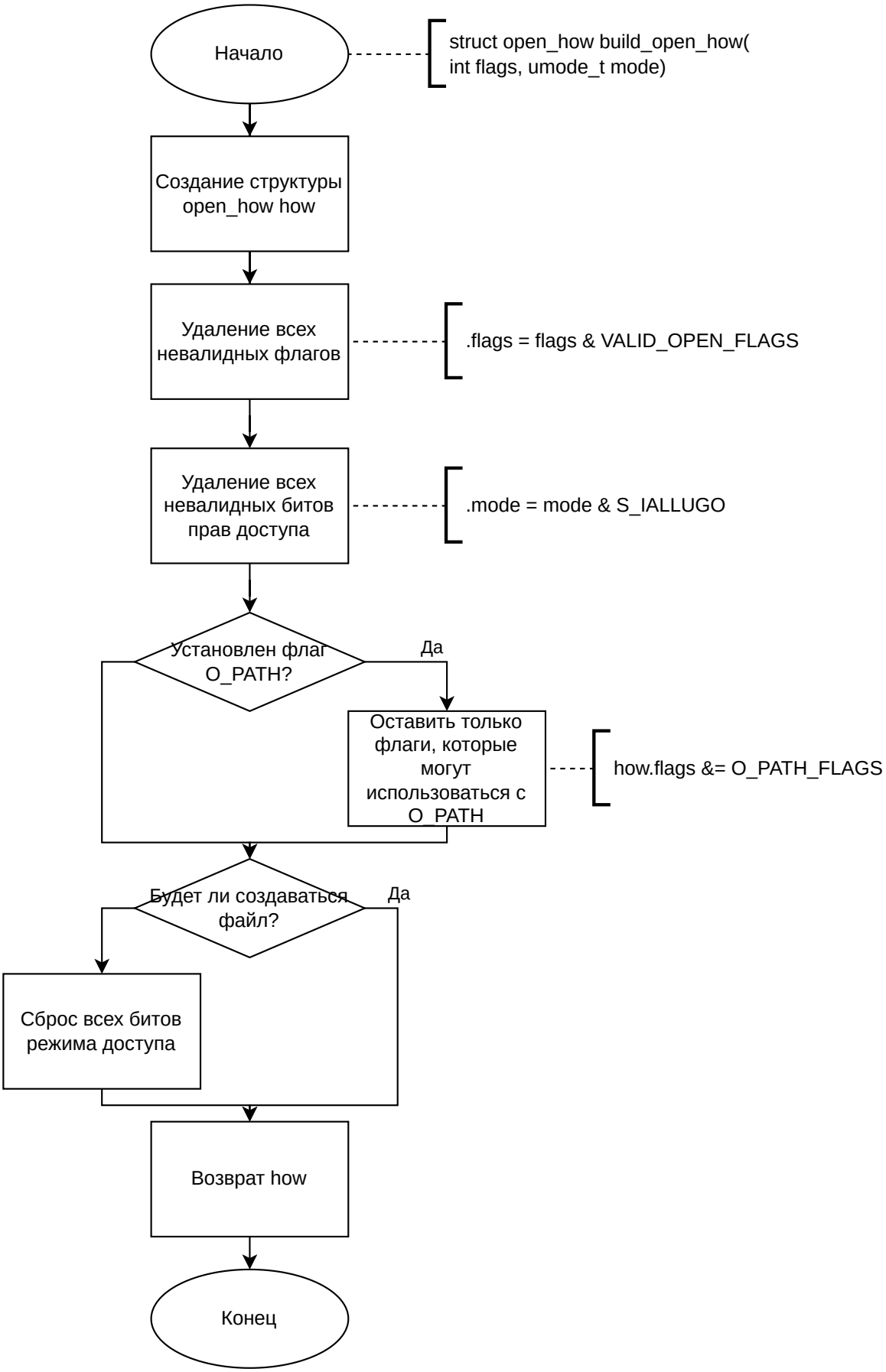

Рисунок 3.2 – Схема алгоритма функции build_open_how()

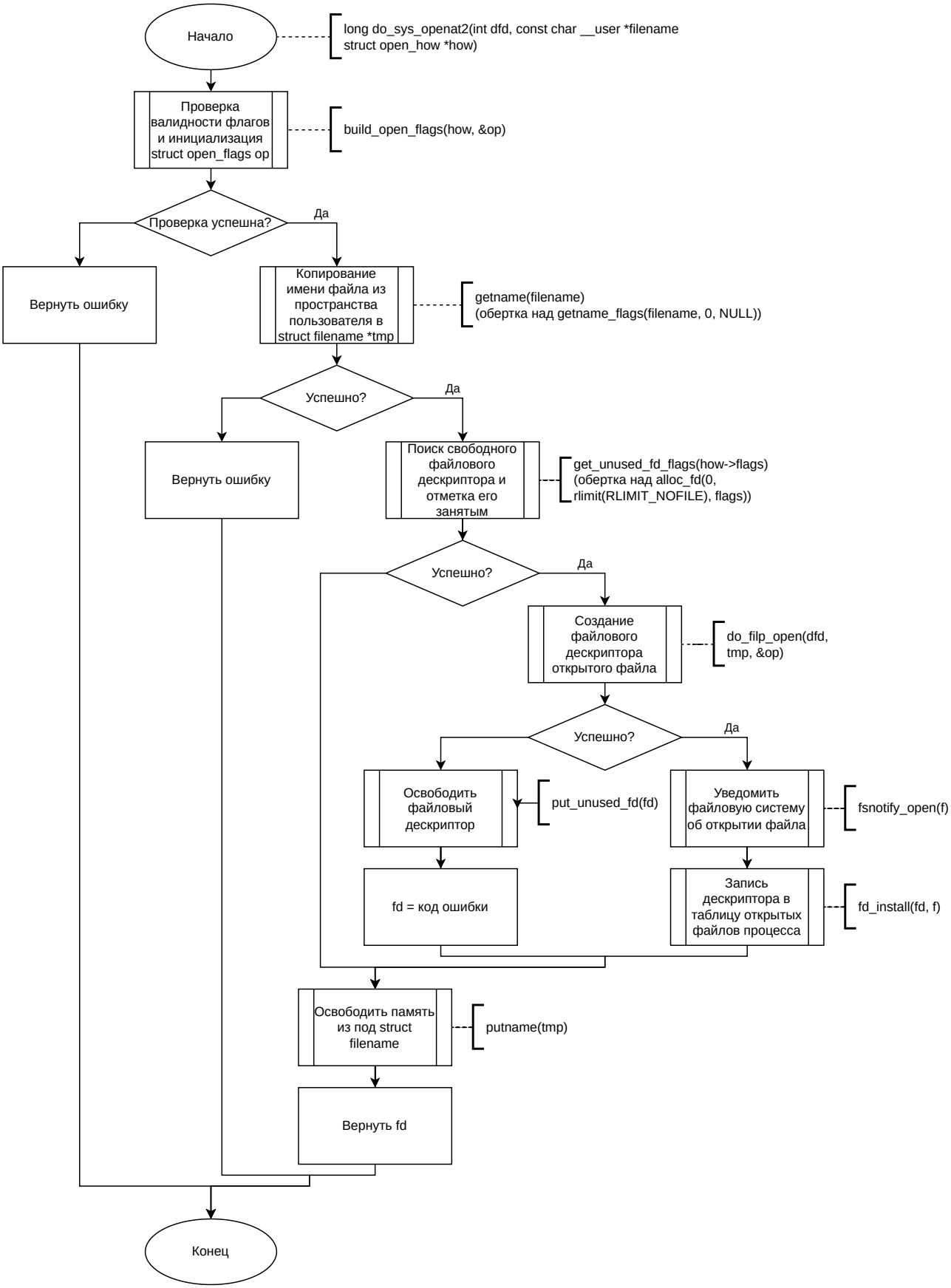

Рисунок 3.3 – Схема алгоритма функции do_sys_openat2()

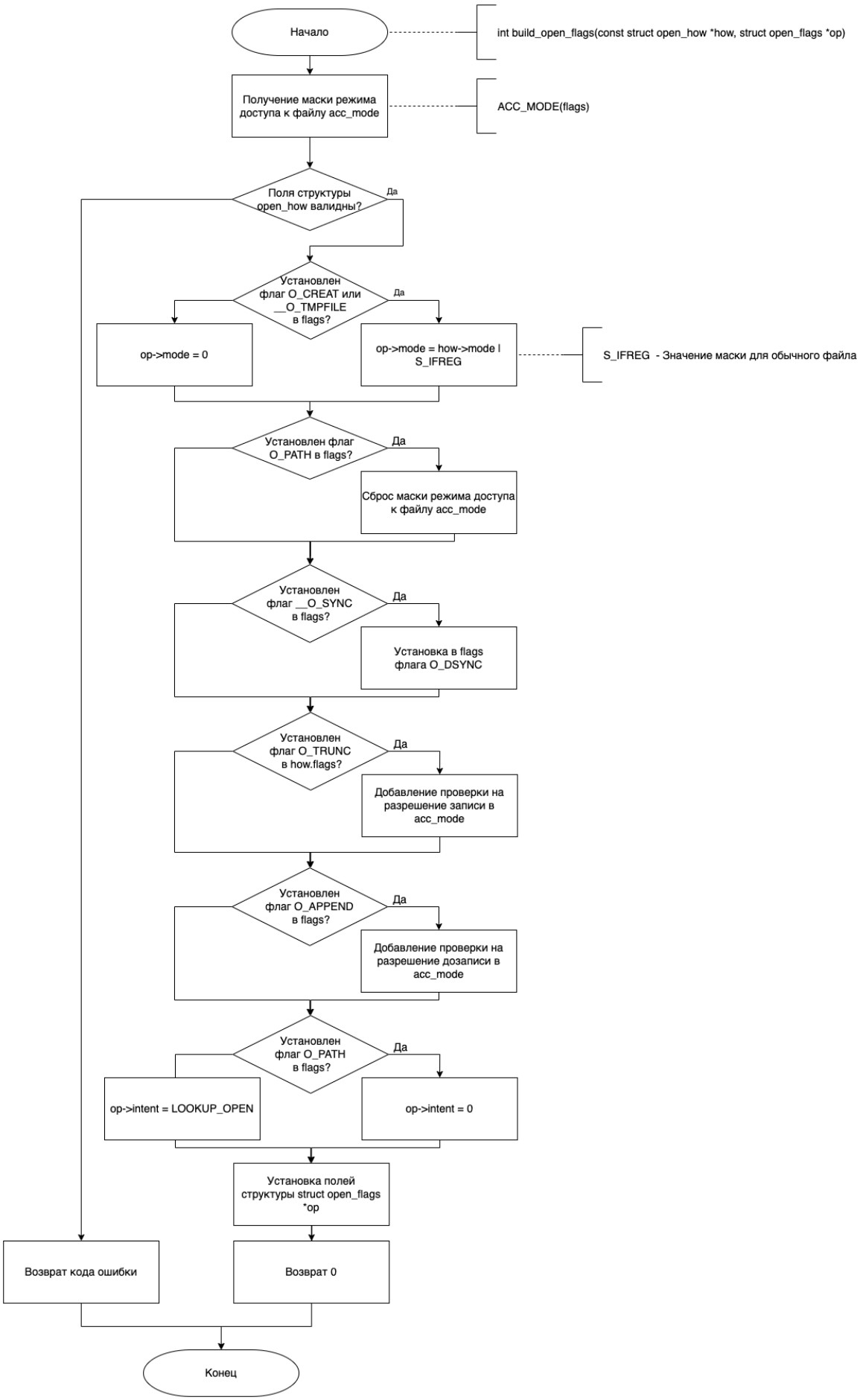

Рисунок 3.4 – Схема алгоритма функции build_open_flags()

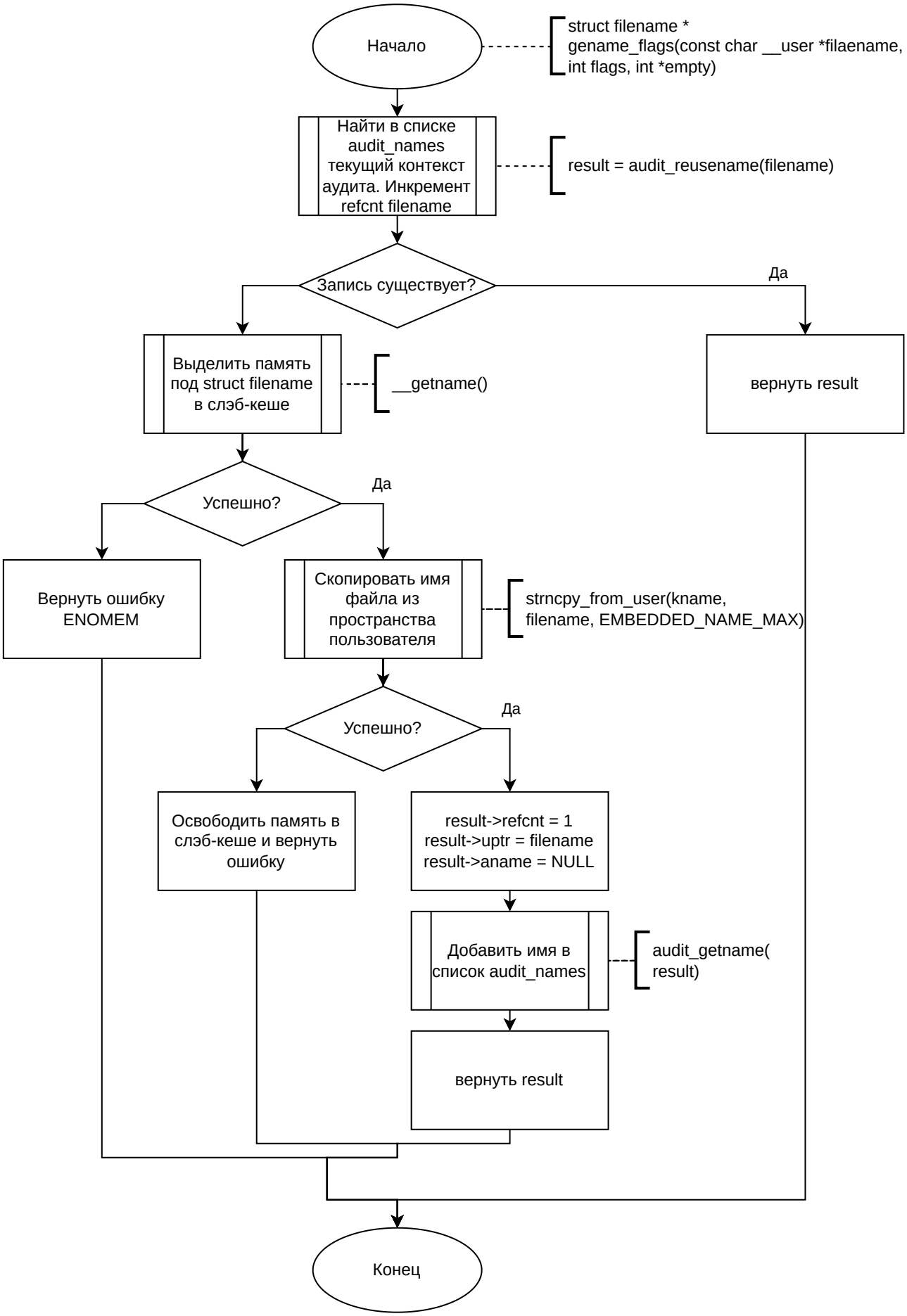

Рисунок 3.5 – Схема алгоритма функции getname_flags()

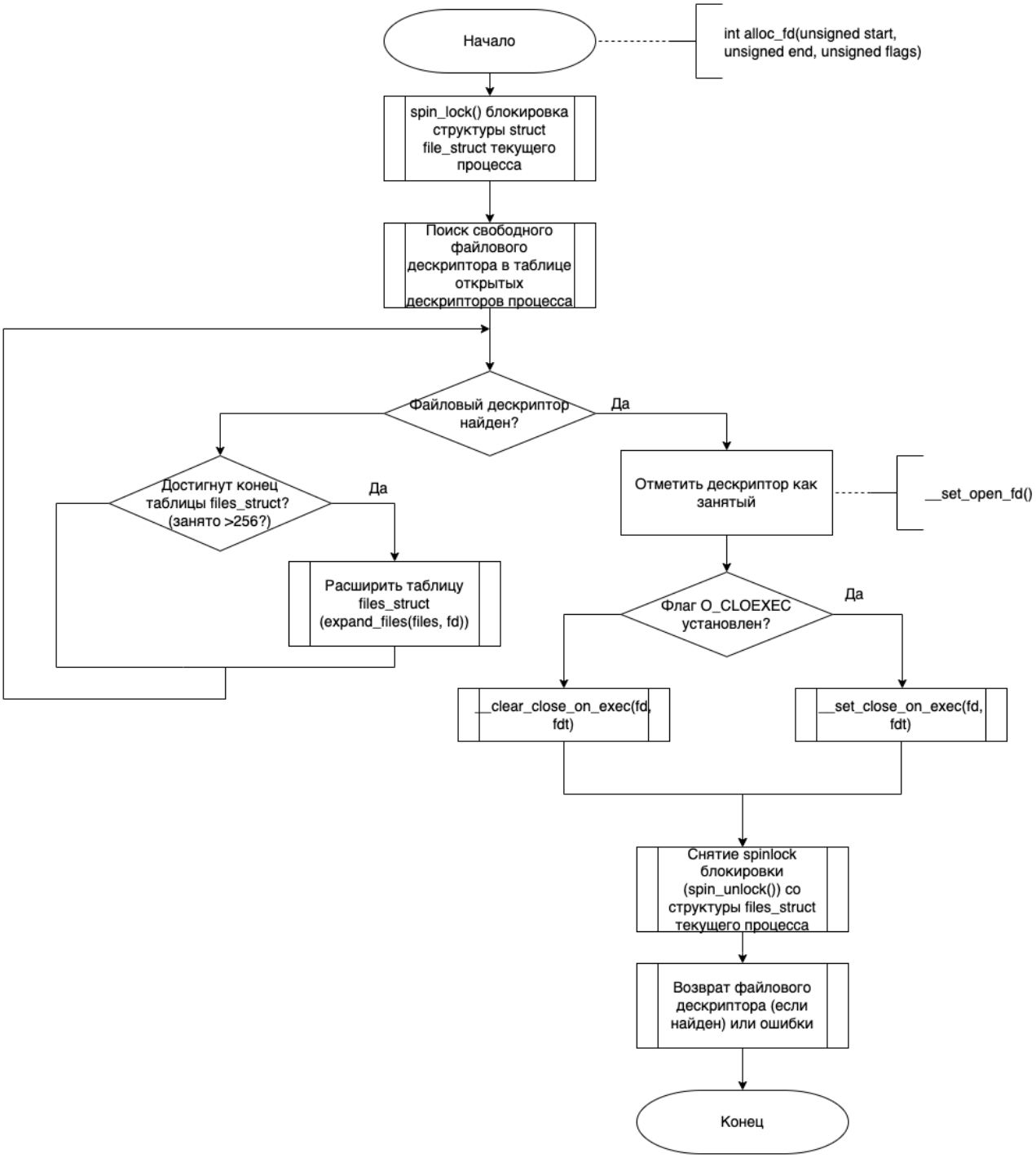

Рисунок 3.6 – Схема алгоритма функции __alloc_fd()

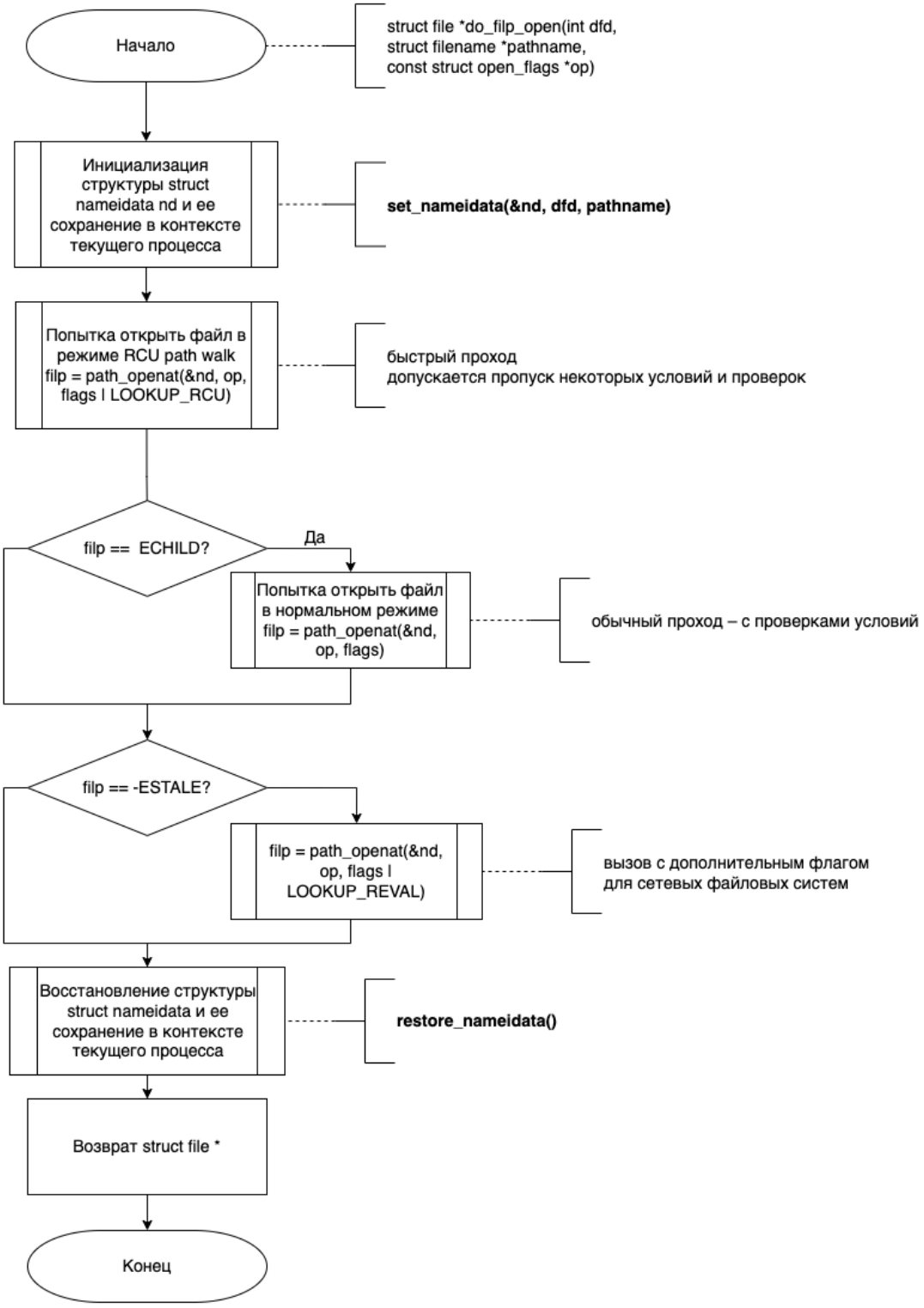

Рисунок 3.7 – Схема алгоритма функции do_filp_open()

LOOKUP_RCU — флаг используется в системе VFS для указания , что операция поиска должна выполняться с использованием RCU (Read-Copy-Update).

LOOKUP_REVAL — флаг для работы с NFS, указывает, что необходимо выполнить повторную проверку

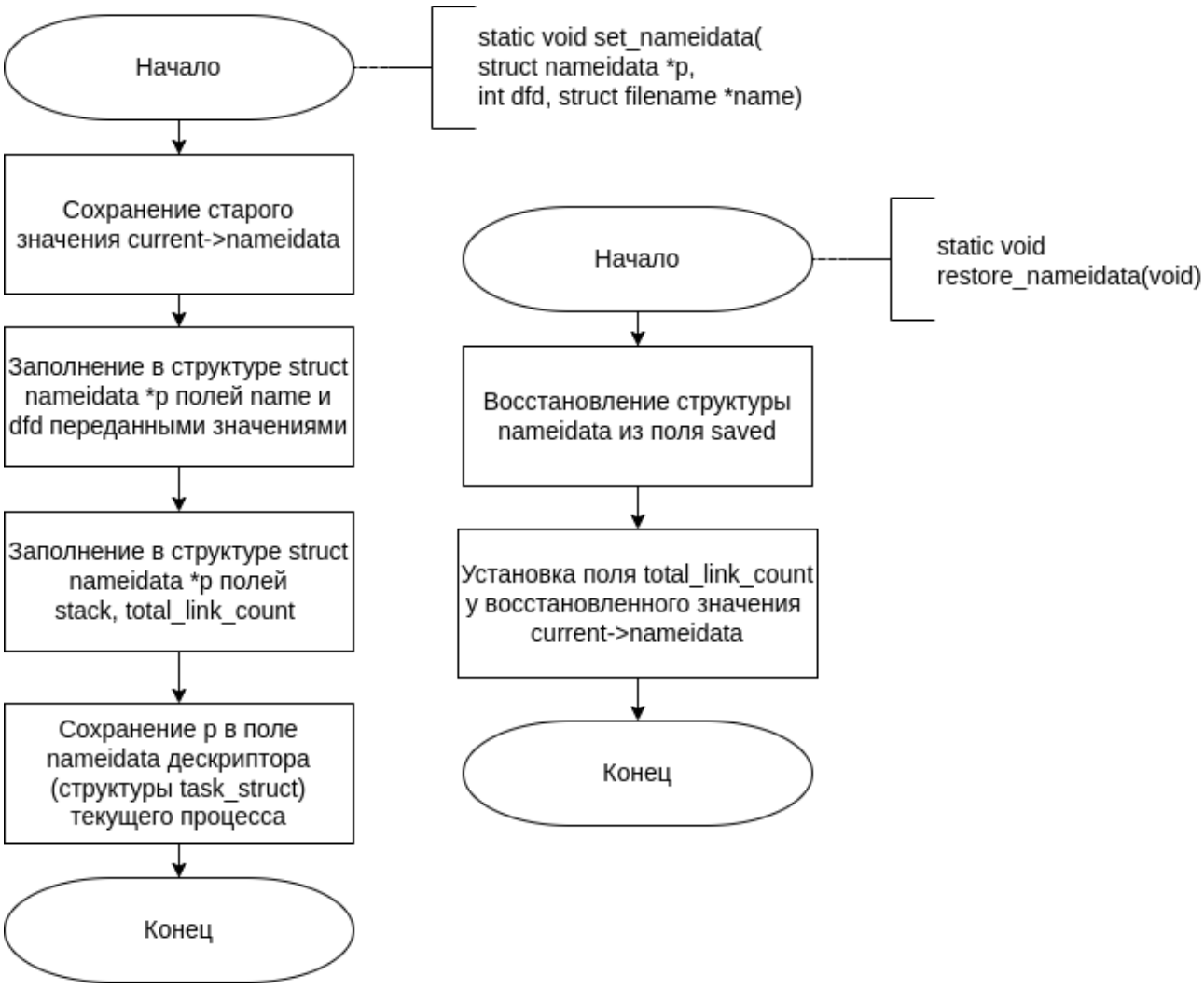

Рисунок 3.8 – Схемы алгоритмов функций set_nameidata() и restore_nameidata()

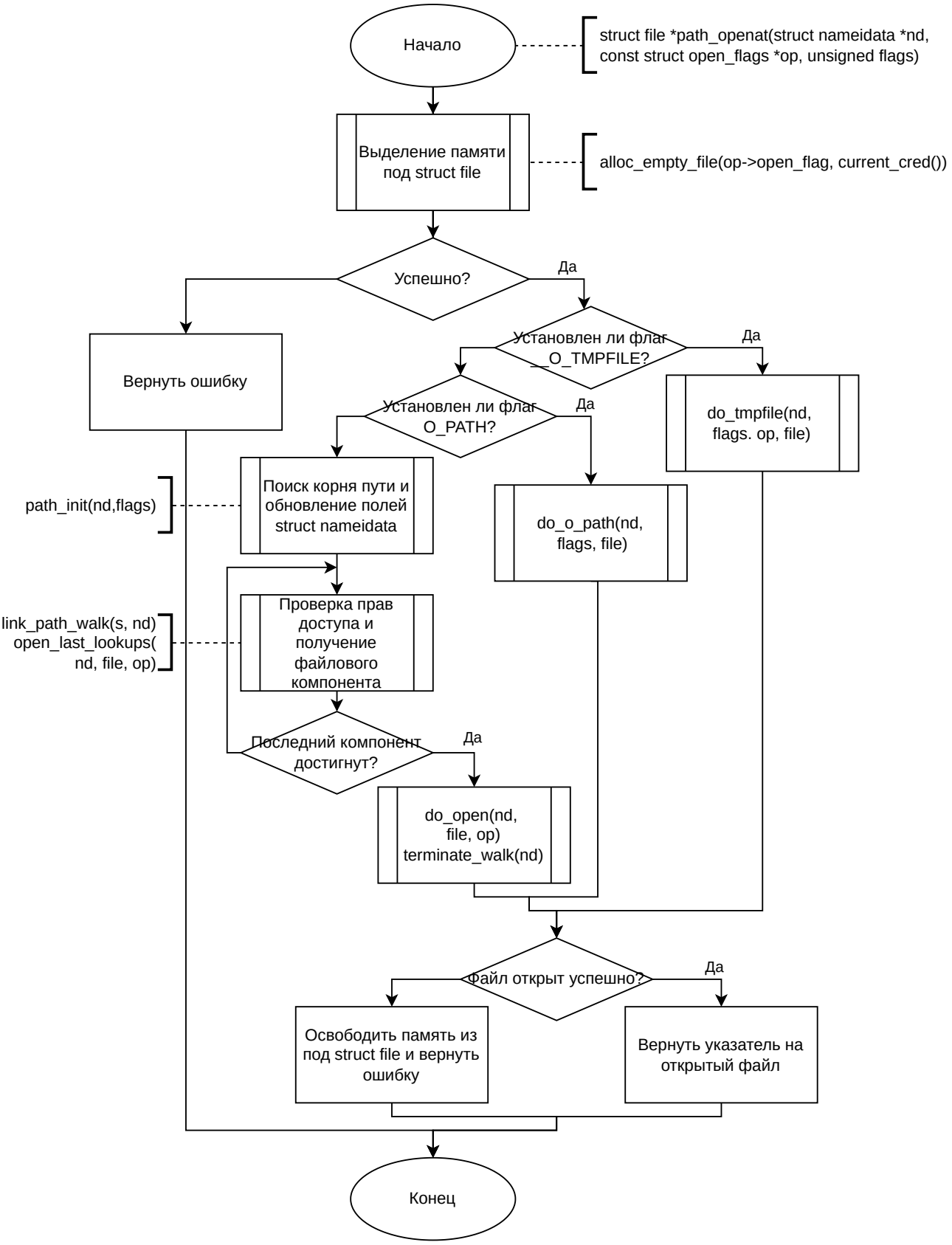

Рисунок 3.9 – Схема алгоритма функции path_openat()

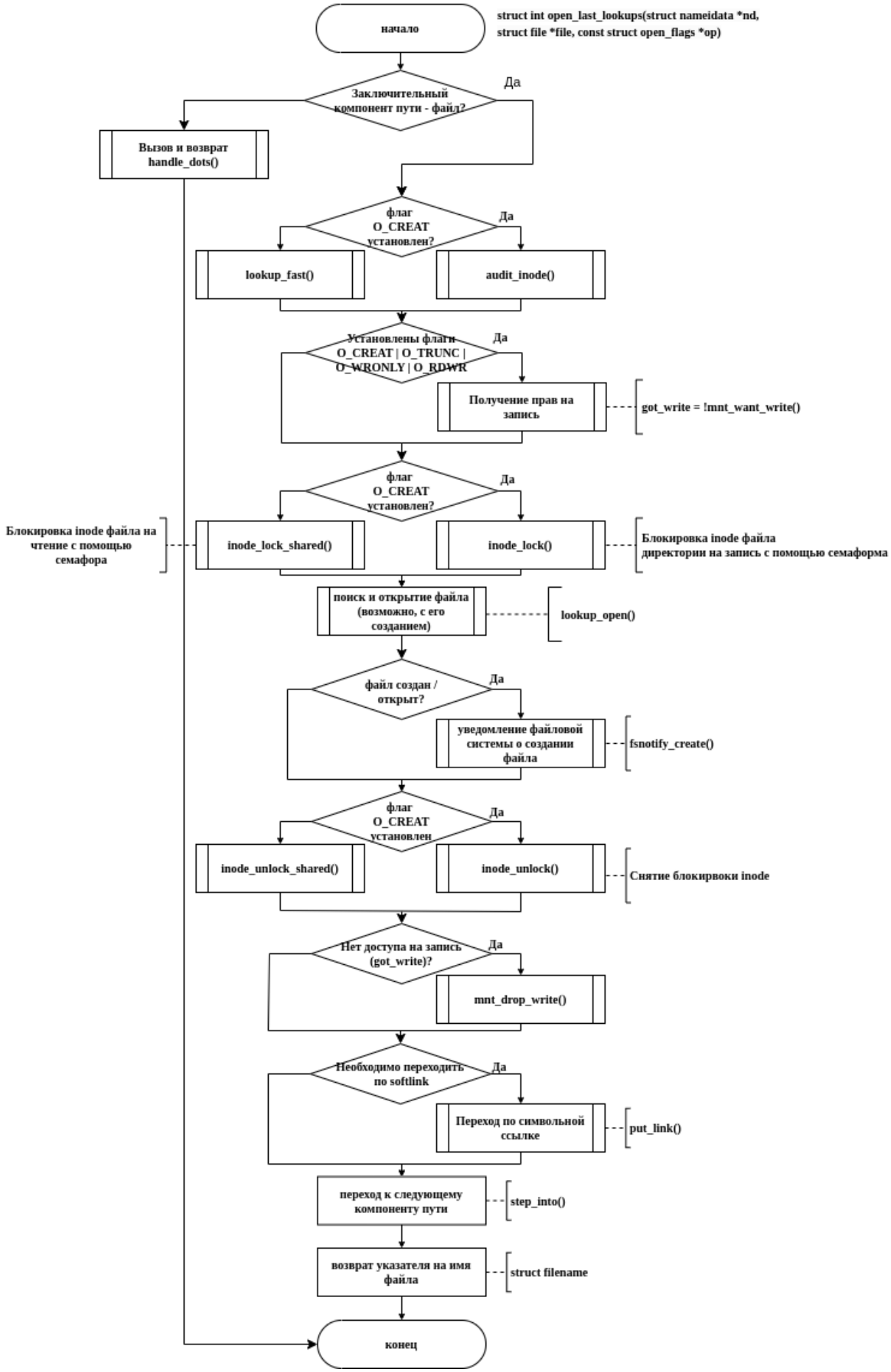

Рисунок 3.10 – Схема алгоритма функции open_last_lookup()

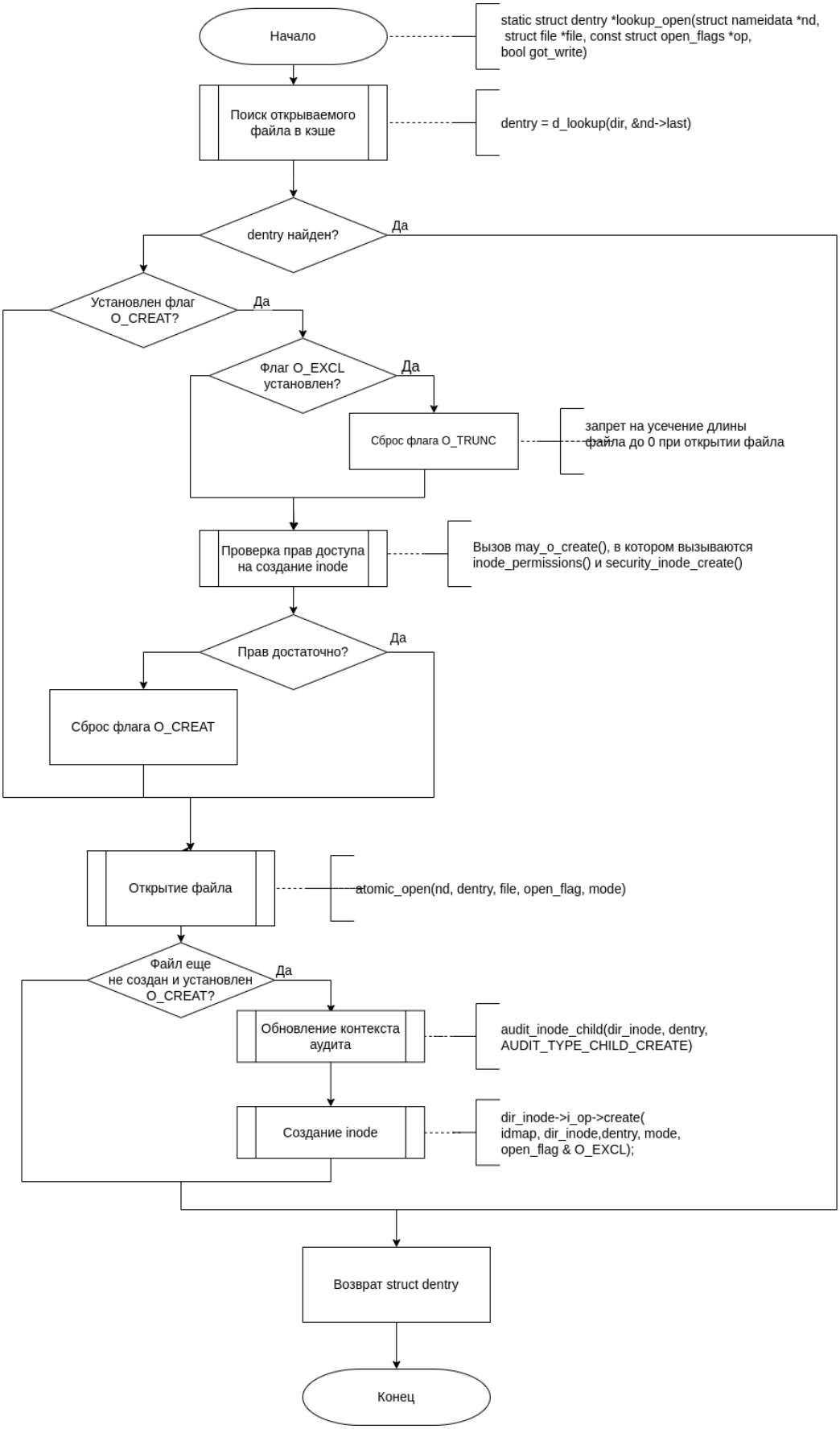

Рисунок 3.11 – Схема алгоритма функции lookup_open()

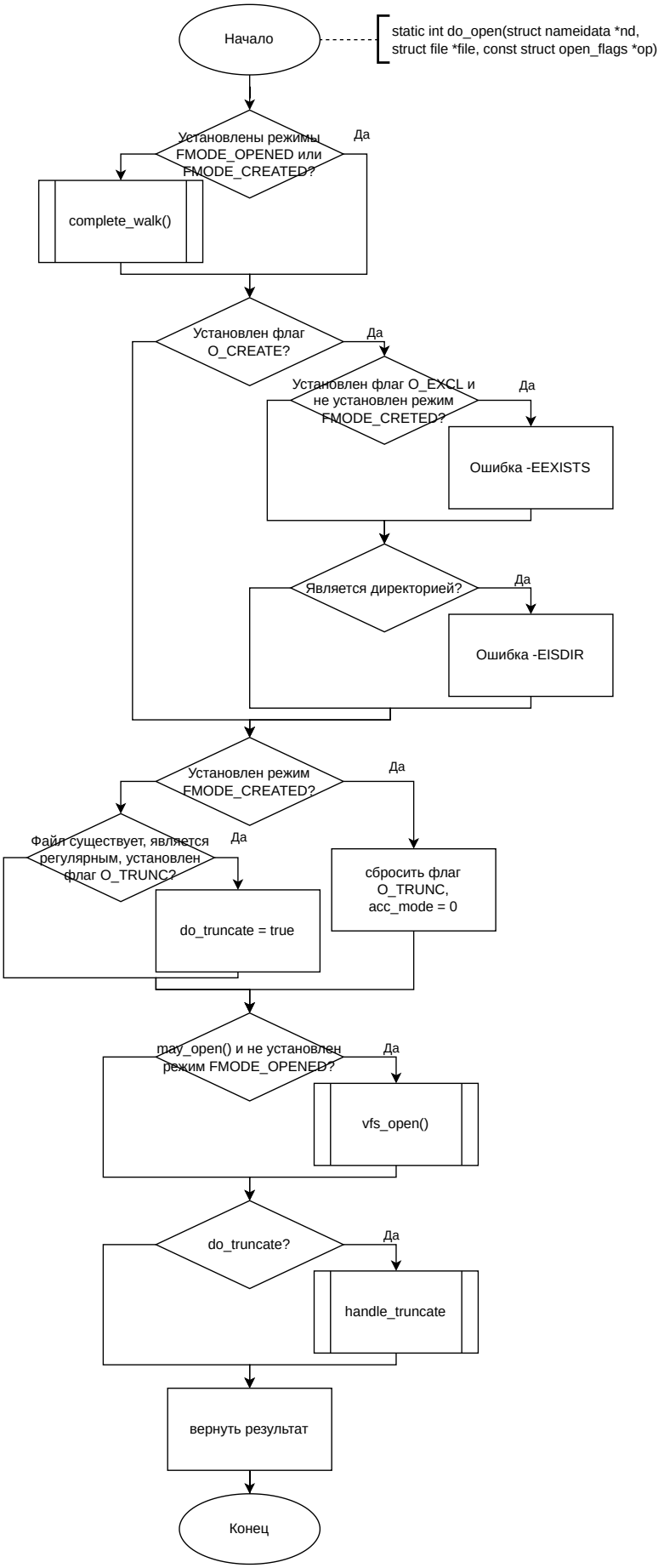

Рисунок 3.12 – Схема алгоритма функции do_open()