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 и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
по курсу «Операционные системы»

«Системный вызов open»

Студент Маслова Марина Дмитриевна

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Рязанова Наталья Юрьевна

2022 г.

1 Используемые структуры

Версия ядра: 5.15.38

```
struct filename {
    const char          *name;
    const __user char   *uptr;
    int                 refcnt;
    struct audit_names  *aname;
    const char          iname[];
};
```

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

```
struct audit_names {
    struct list_head list;

    struct filename *name;
    int             name_len;
    bool            hidden;

    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;
    bool                    should_free;
};
```

```c
#define EMBEDDED_LEVELS 2
struct nameidata {
    struct path path;
    struct qstr last;
    struct path root;
    struct inode *inode;
    unsigned int flags, state;
    unsigned     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;
```

```c
struct open_how {
    __u64 flags;
    __u64 mode;
    __u64 resolve;
};
```

```c
struct path {
    struct vfsmount *mnt;
    struct dentry *dentry;
} __randomize_layout;
```

Флаги системного вызова `open()`

`O_CREAT` — если файл не существует, то он будет создан.

`O_EXCL` — если используется совместно с `O_CREAT`, то при наличии уже созданного файла вызов завершится ошибкой.

`O_NOCTTY` — если файл указывает на терминальное устройство, то оно не станет терминалом управления процесса, даже при его отсутствии.

`O_TRUNC` — если файл уже существует, он является обычным файлом и заданный режим позволяет записывать в этот файл, то его длина будет урезана до нуля.

`O_APPEND` — файл открывается в режиме добавления, перед каждой операцией записи файловый указатель будет устанавливаться в конец файла.

`O_NONBLOCK`, `O_NDELAY` — файл открывается, по возможности, в режиме non-blocking, то есть никакие последующие операции над дескриптором файла не заставляют в дальнейшем вызывающий процесс ждать.

`O_SYNC` — файл открывается в режиме синхронного ввода-вывода, то есть все операции записи для соответствующего дескриптора файла блокируют вызывающий процесс до тех пор, пока данные не будут физически записаны

`O_NOFOLLOW` — если файл является символической ссылкой, то open вернёт ошибку.

`O_DIRECTORY` — если файл не является каталогом, то open вернёт ошибку.

`O_LARGEFILE` — позволяет открывать файлы, размер которых не может быть представлен типом off_t (long).

`O_DSYNC` — операции записи в файл будут завершены в соответствии с требованиями целостности данных синхронизированного завершения ввода-вывода.

`O_NOATIME` — запрет на обновление времени последнего доступа к файлу при его чтении.

`O_TMPFILE` — при наличии данного флага создаётся неименованный временный обычный файл.

`O_CLOEXEC` — включает флаг close-on-exec для нового файлового дескриптора, указание этого флага позволяет программе избегать дополнительных операций `fcntl F_SETFD` для установки флага `FD_CLOEXEC`.

2 Схема алгоритма

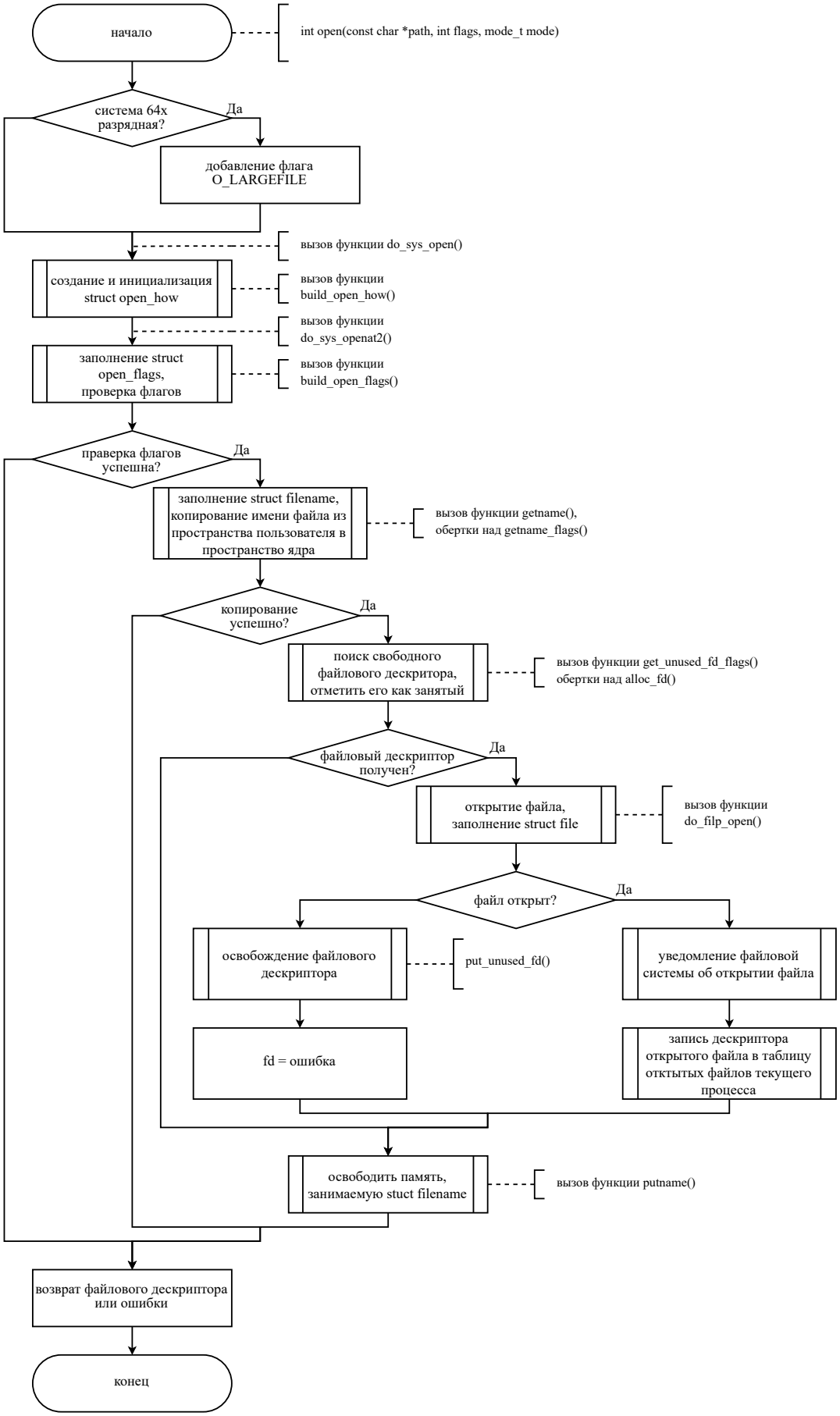

Рисунок 2.1 – Схема алгоритма работы системного вызова `open`

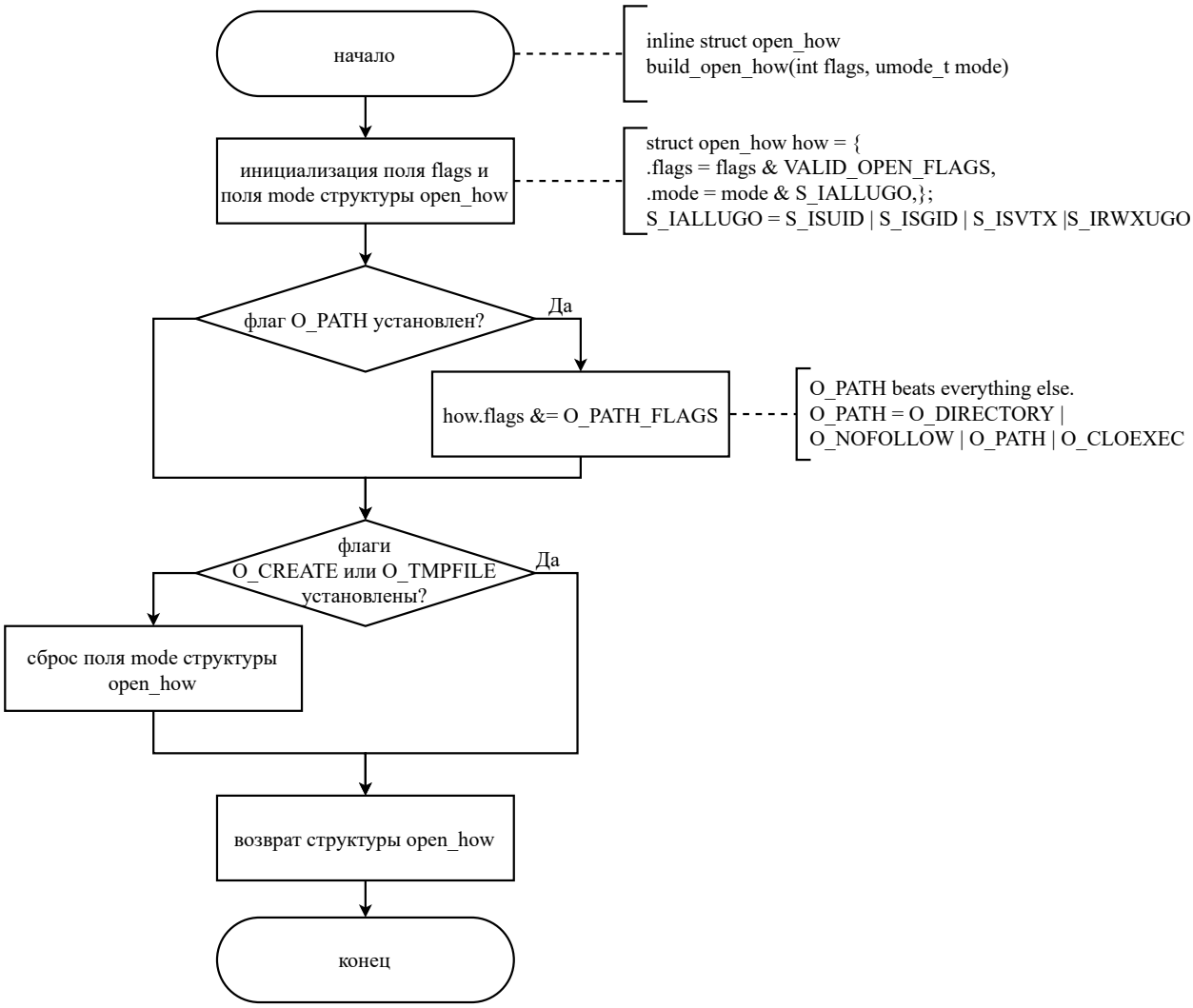

Рисунок 2.2 – Схема алгоритма работы функции `build_open_how`

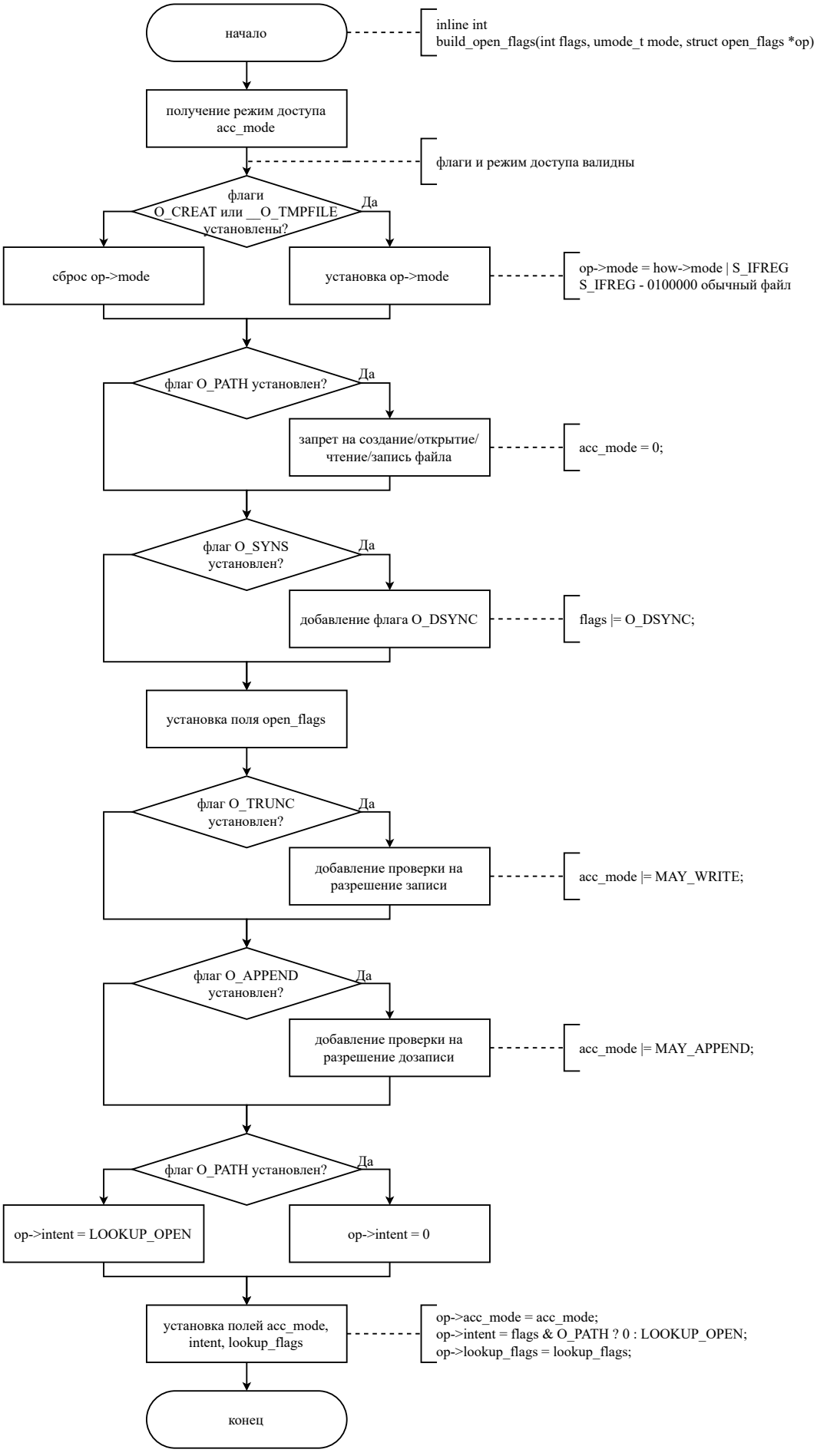

Рисунок 2.3 – Схема алгоритма работы функции `build_open_flags`

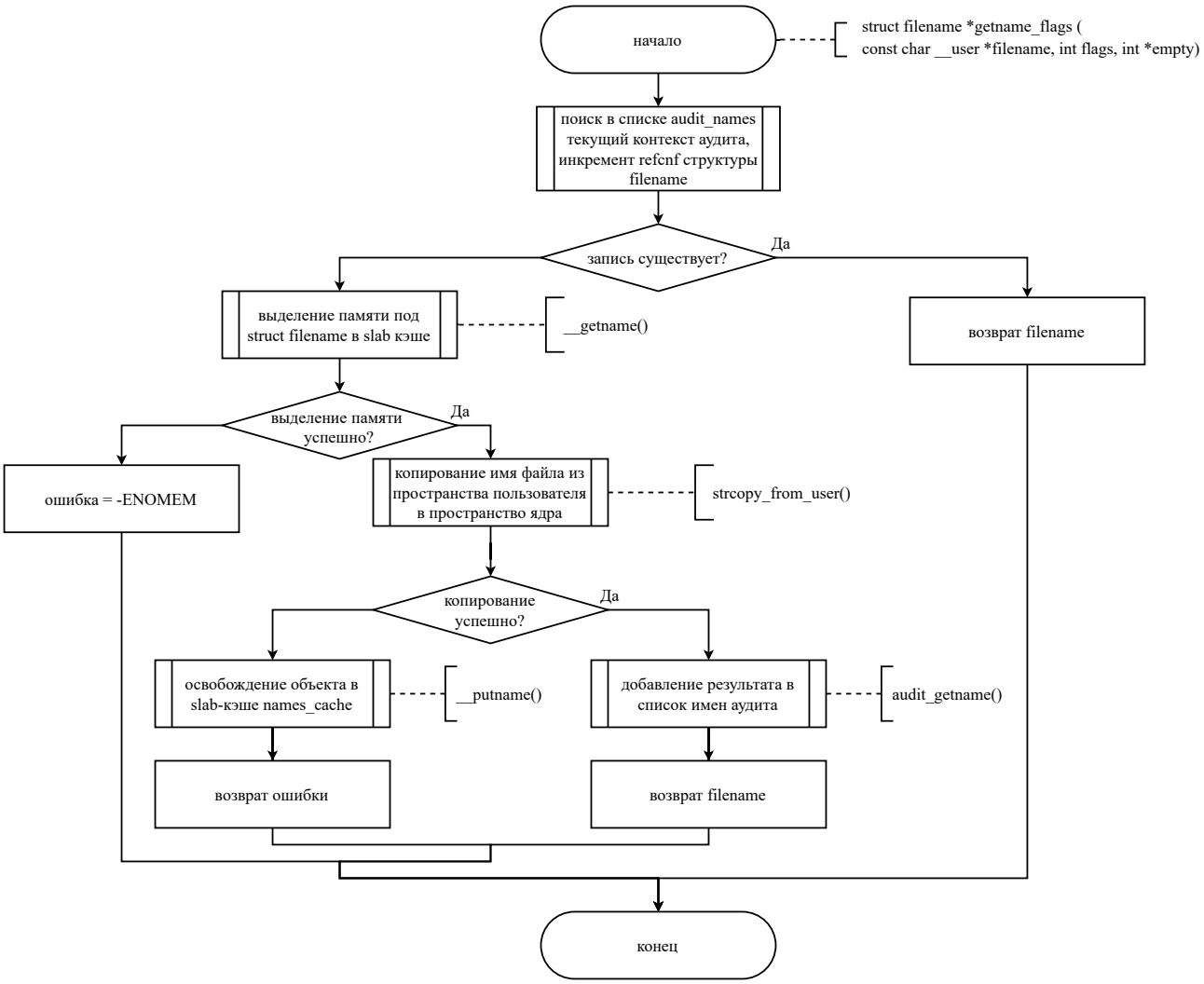

Рисунок 2.4 – Схема алгоритма работы функции `getname_flags`

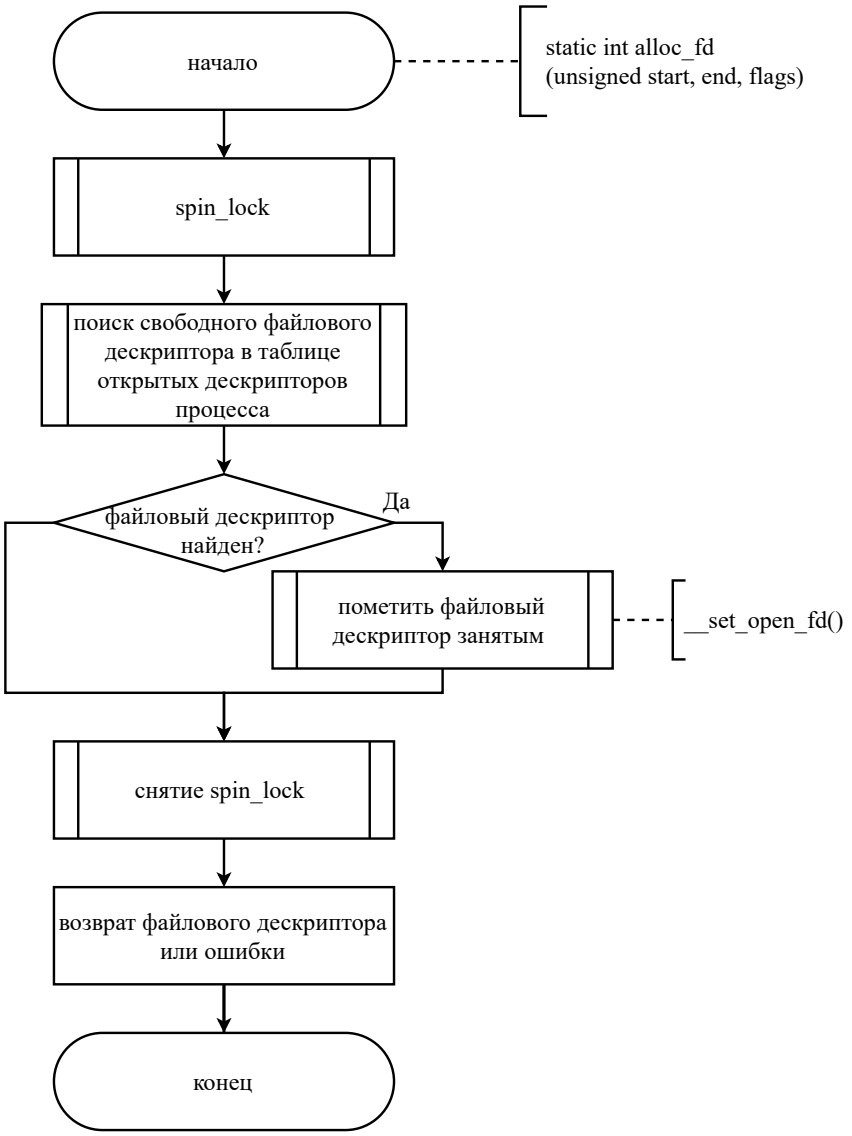

Рисунок 2.5 – Схема алгоритма работы функции `alloc_fd`

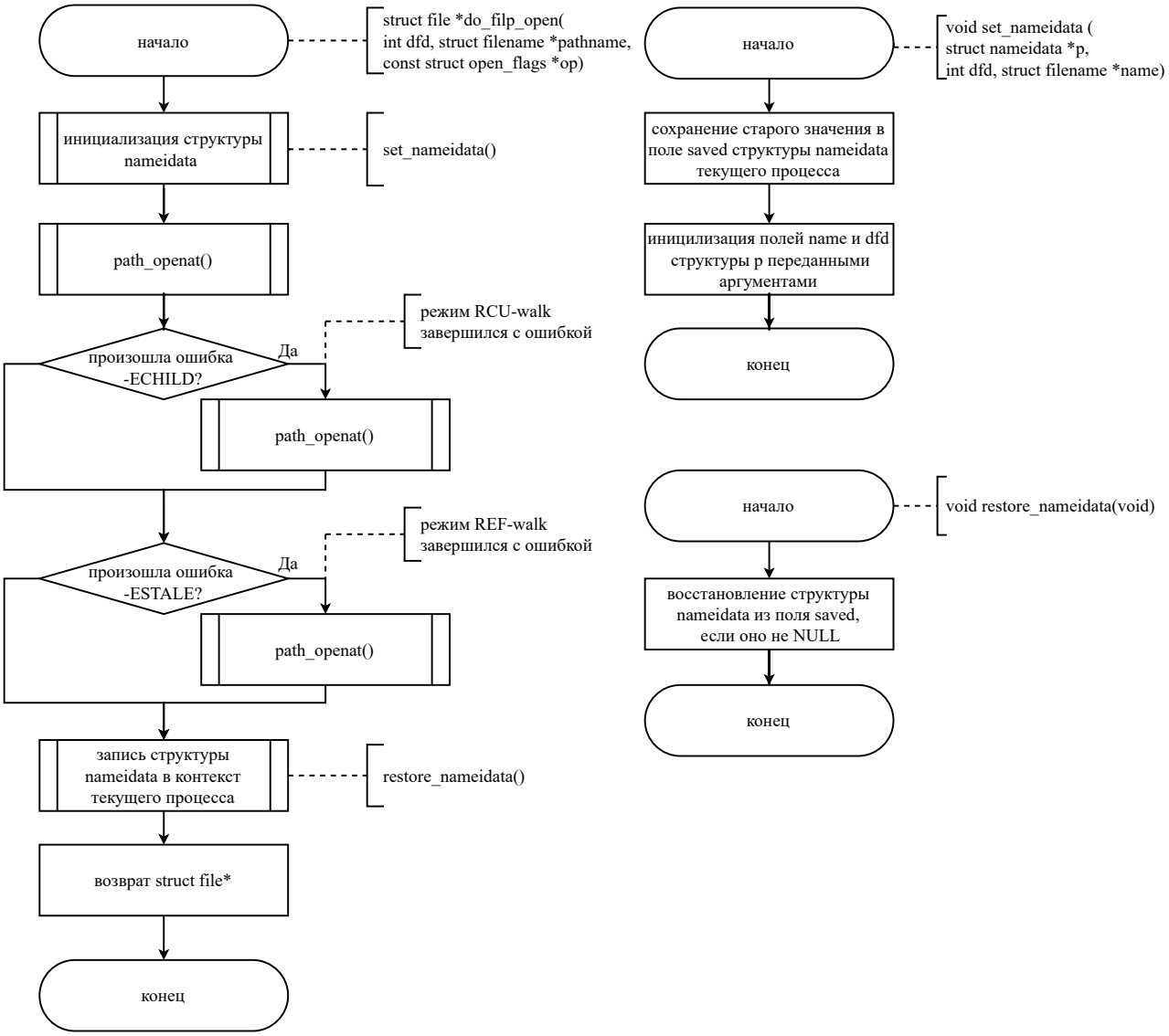

Рисунок 2.6 – Схема алгоритма работы функции `do_filp_open`

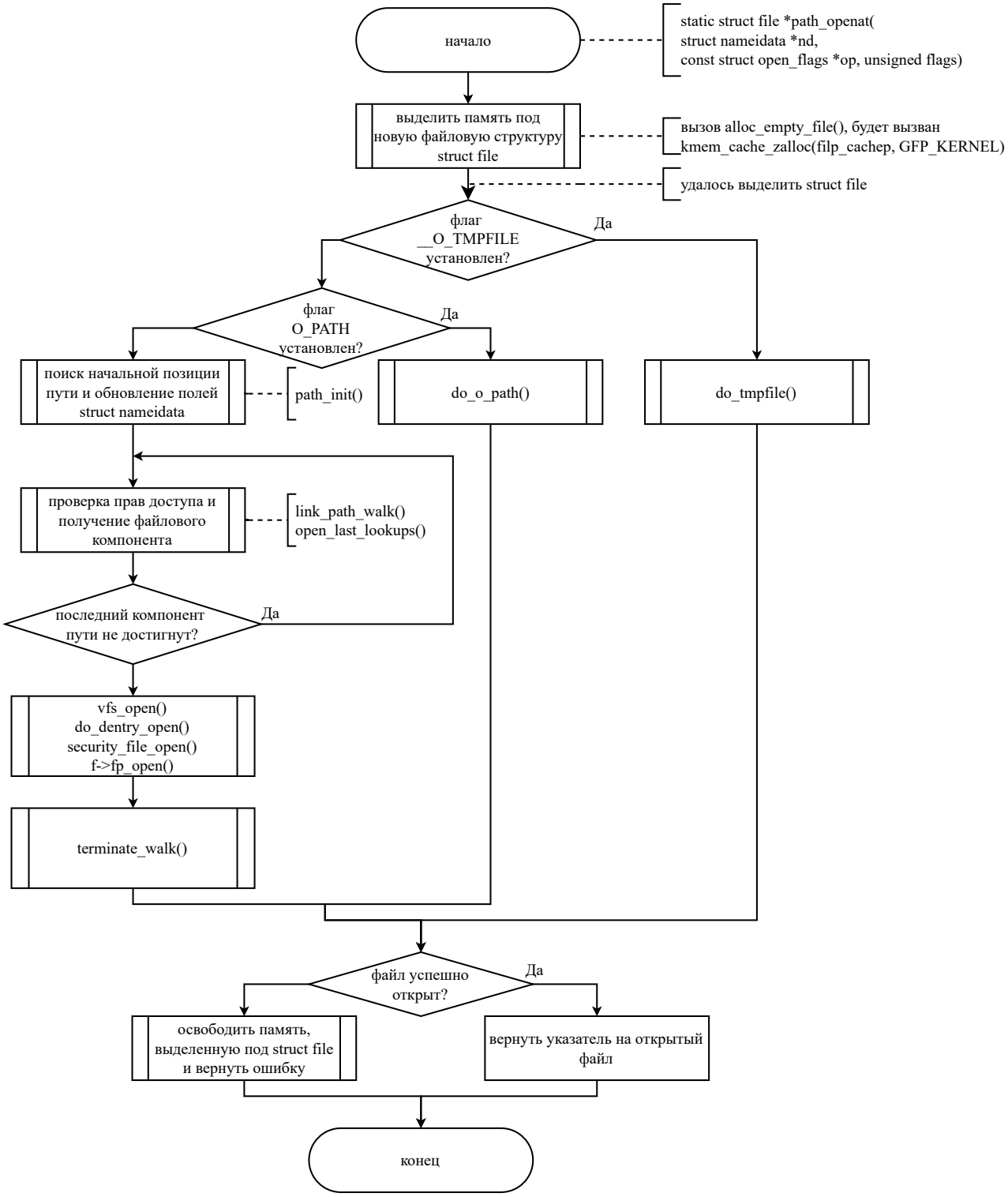

Рисунок 2.7 – Схема алгоритма работы функции `path_openat`

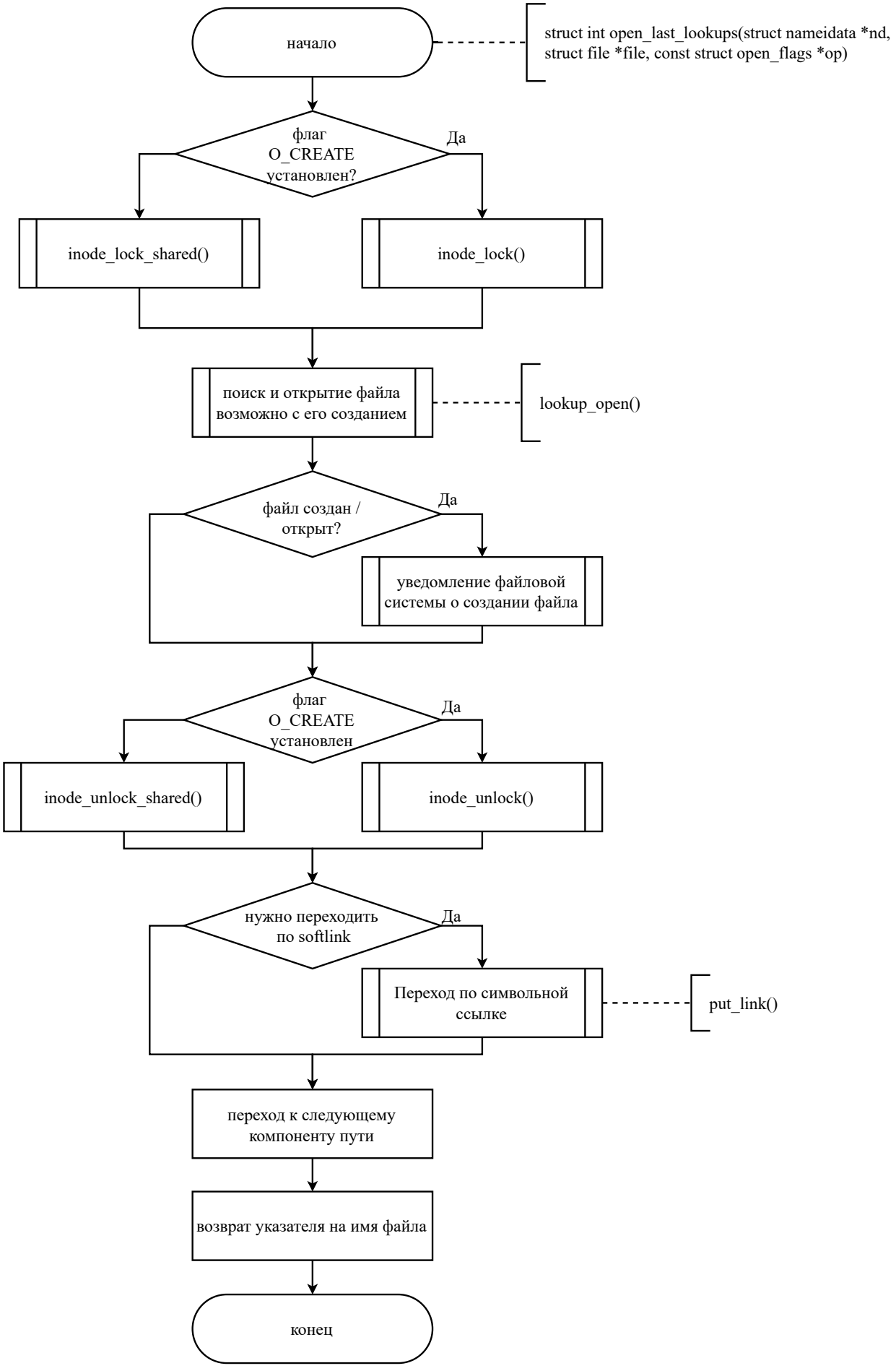

Рисунок 2.8 – Схема алгоритма работы функции `open_last_lookups`

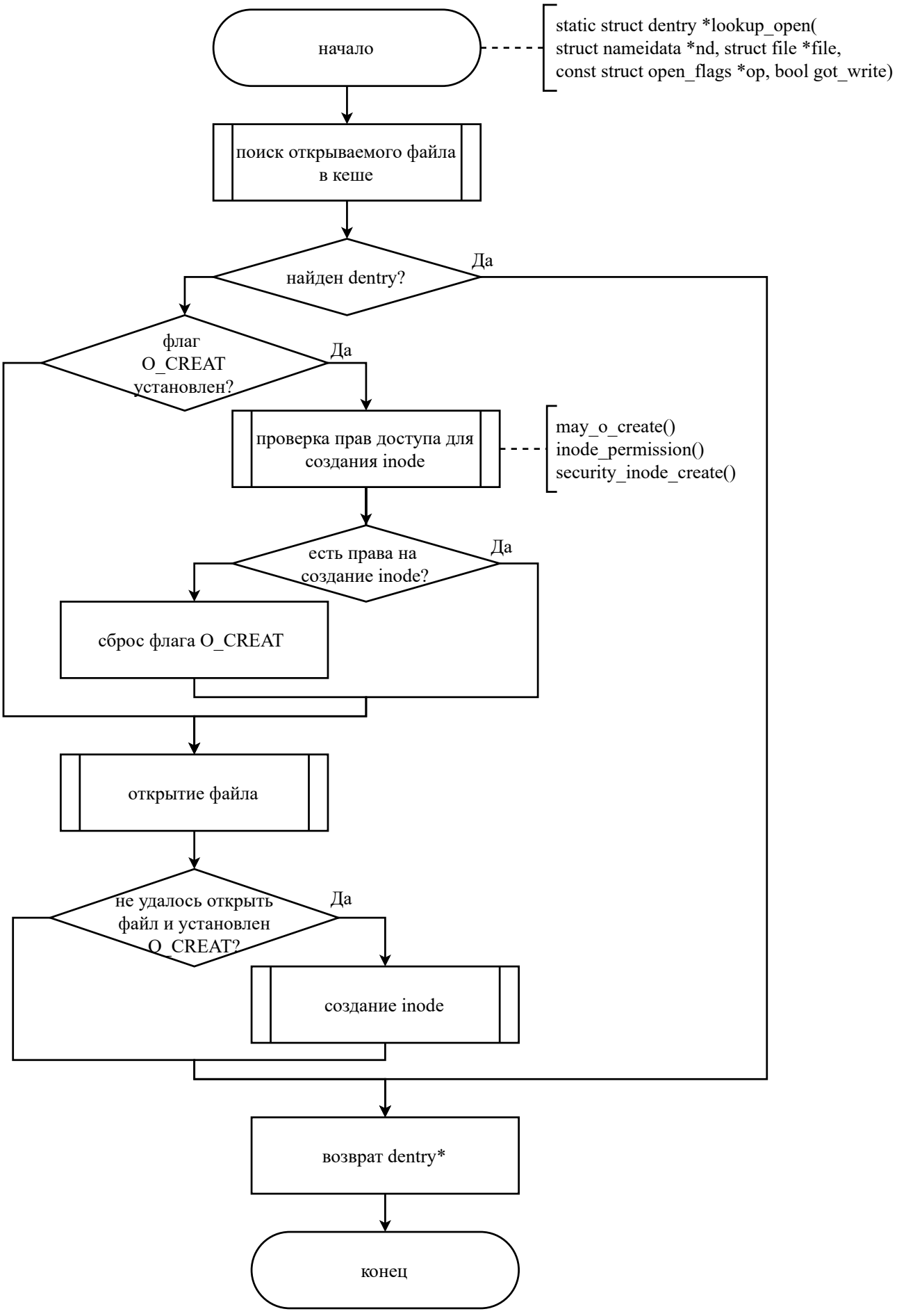

Рисунок 2.9 – Схема алгоритма работы функции `lookup_open`