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»

Тема Системный вызов open

Студент Морозов Д.В.

Группа ИУ7-62Б

Оценка (баллы)

Преподаватель Рязанова Н.Ю.

Москва — 2023 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;
};
```

```
/* When fs/namei.c:getname() is called, we store the pointer in
   name and bump
 * the refcnt in the associated filename struct.
 *
 * Further, in fs/namei.c:path_lookup() we store the inode and
    device.
 */
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;
```

```
17      kgid_t          gid;
18      dev_t           rdev;
19      u32         osid;
20      struct audit_cap_data   fcap;
21      unsigned int        fcap_ver;
22      unsigned char       type;       /* record type */
23      /*
24       * This was an allocated audit_names and not from the array of
25       * names allocated in the task audit context.  Thus this name
26       * should be freed on syscall exit.
27       */
28      bool            should_free;
29  };
```

```c
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;
    vfsuid_t    dir_vfsuid;
    umode_t     dir_mode;
} __randomize_layout;
```

```c
/*
 * Arguments for how openat2(2) should open the target path. If only @flags and
 * @mode are non-zero, then openat2(2) operates very similarly to openat(2).
 *
 * However, unlike openat(2), unknown or invalid bits in @flags result in
 * -EINVAL rather than being silently ignored. @mode must be zero unless one of
 * {O_CREAT, O_TMPFILE} are set.
 *
 * @flags: O_* flags.
 * @mode: O_CREAT/O_TMPFILE file mode.
 * @resolve: RESOLVE_* flags.
 */
struct open_how {
```

```
14      __u64 flags;
15      __u64 mode;
16      __u64 resolve;
17  };
```

```
1  struct path {
2      struct vfsmount *mnt;
3      struct dentry *dentry;
4  } __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 де-

скриптора, указание этого флага позволяет программе избегать дополнительных операций `fcntl F_SETFD` для установки флага `FD_CLOEXEC`.

2 Схема алгоритма

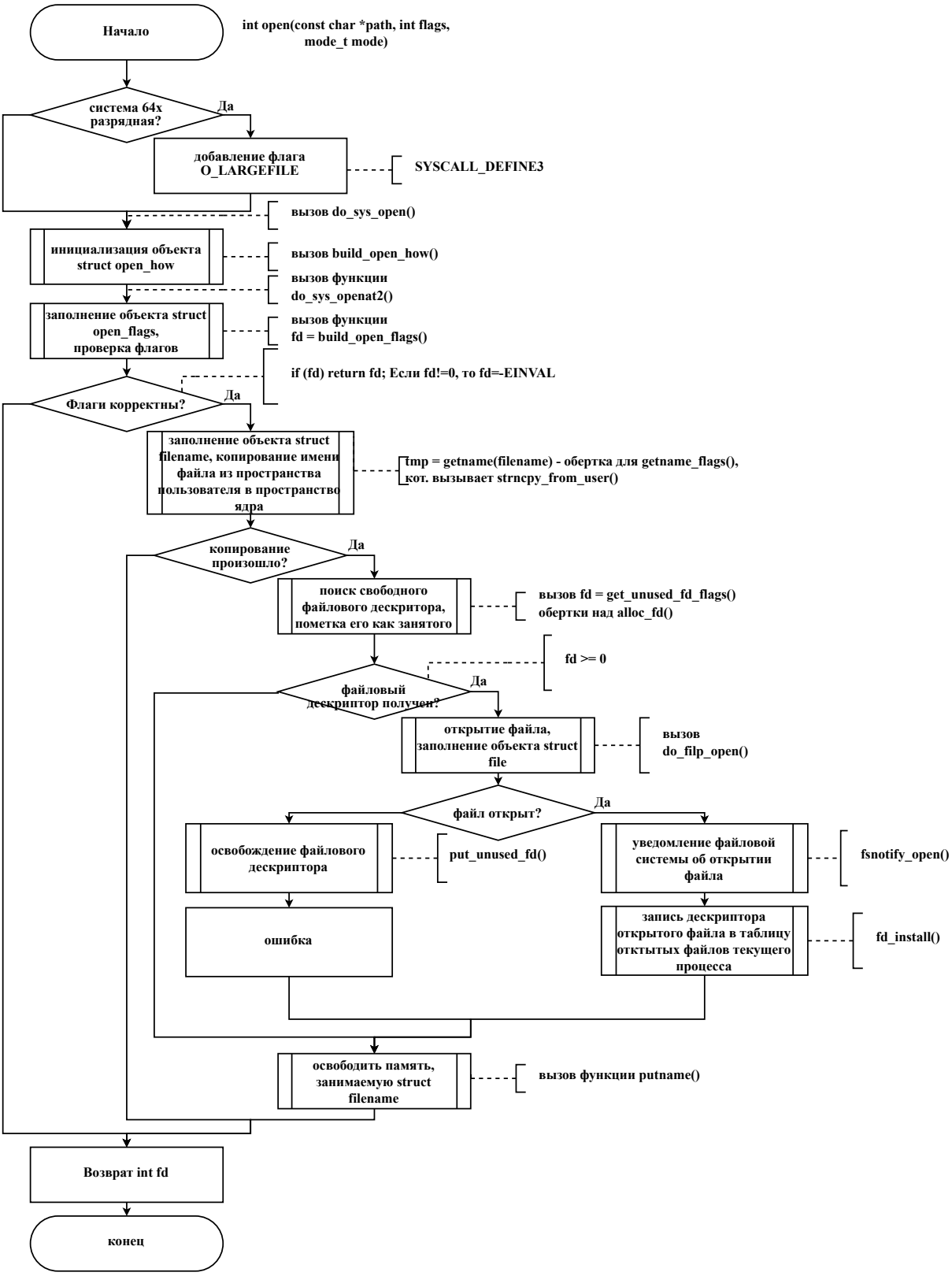

Рисунок 2.1 – Схема алгоритма работы системного вызова `open`

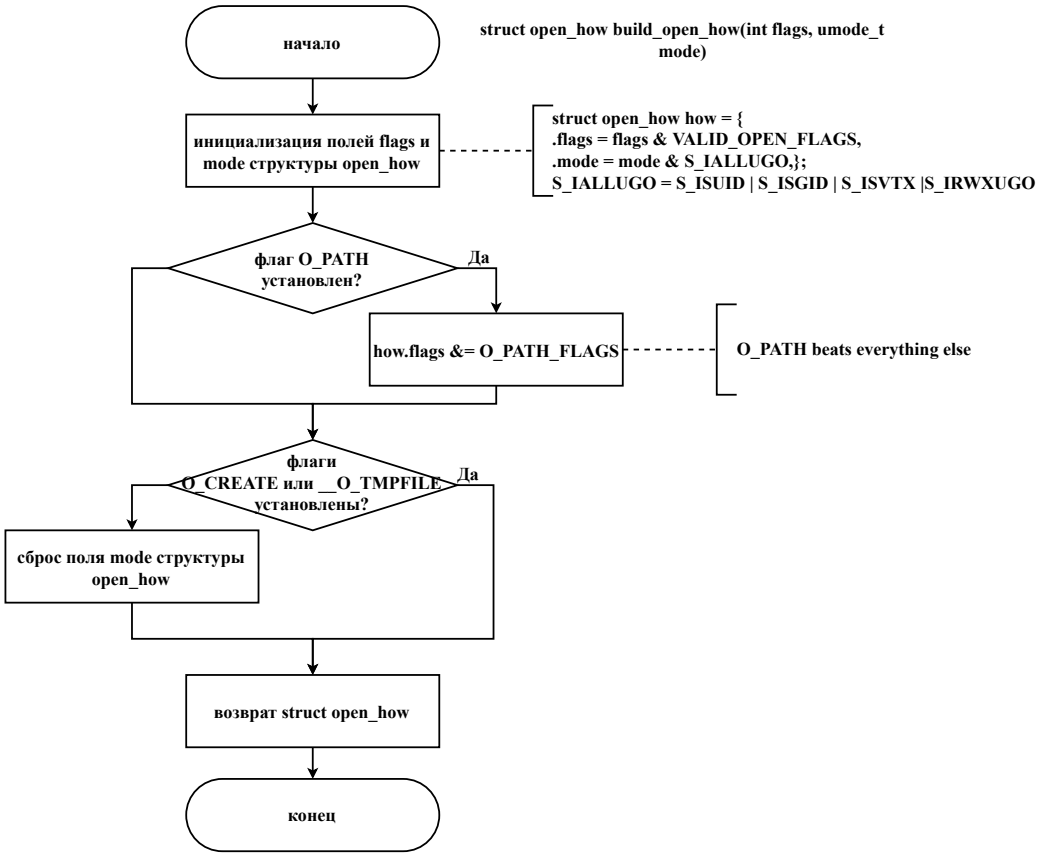

Рисунок 2.2 – Схема алгоритма работы функции `build_open_how`

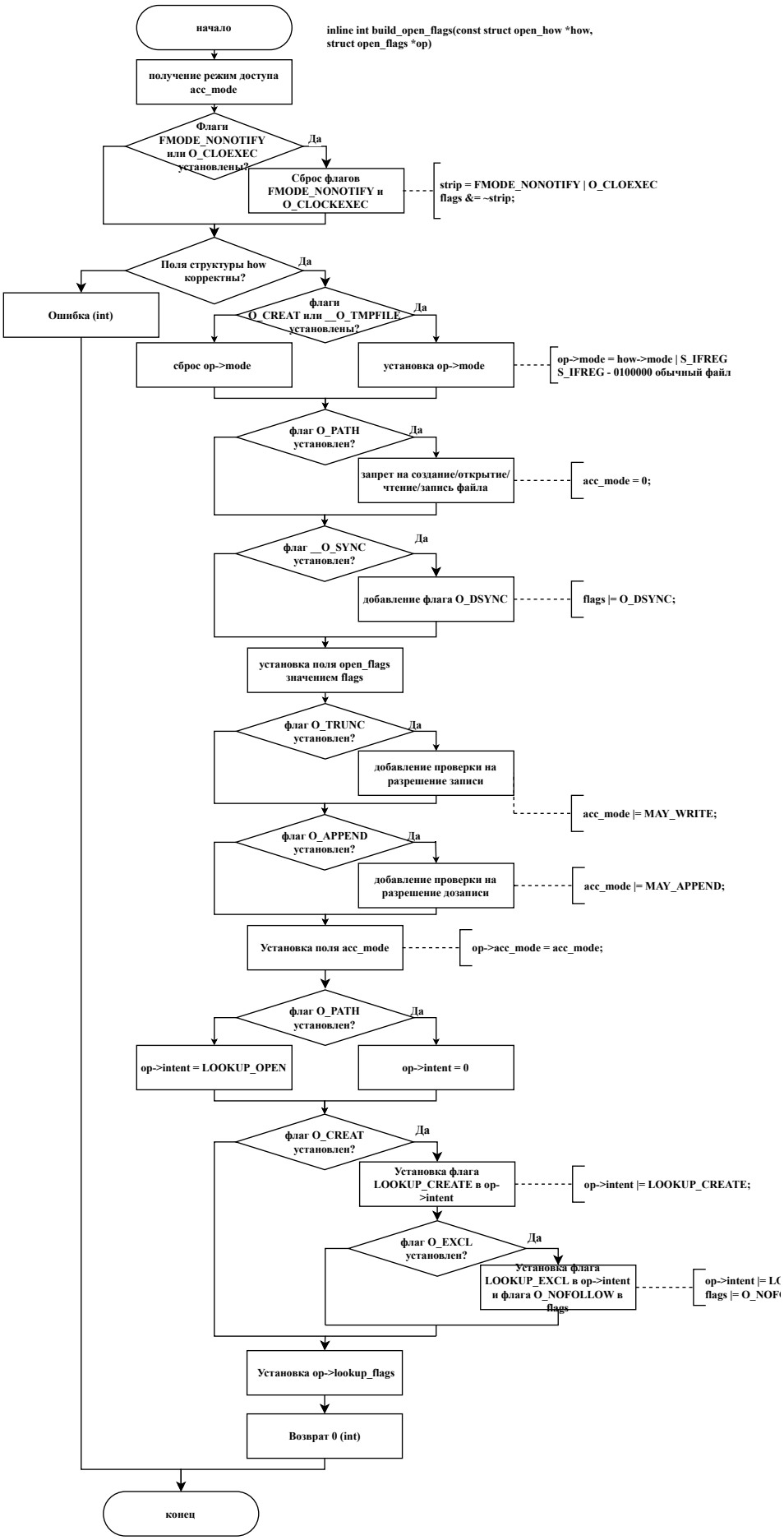

Рисунок 2.3 – Схема алгоритма работы функции `build_open_flags`

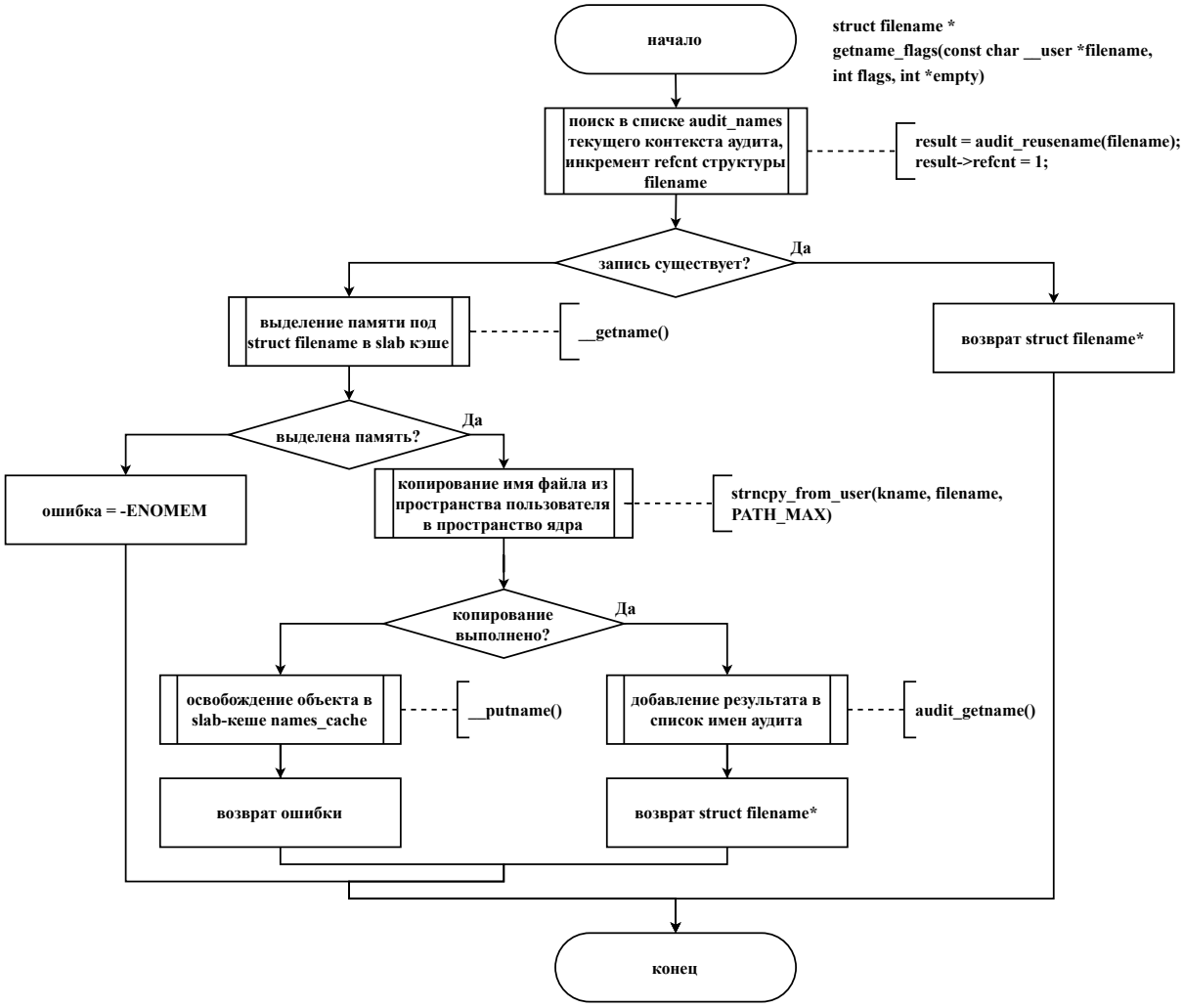

Рисунок 2.4 – Схема алгоритма работы функции `getname_flags`

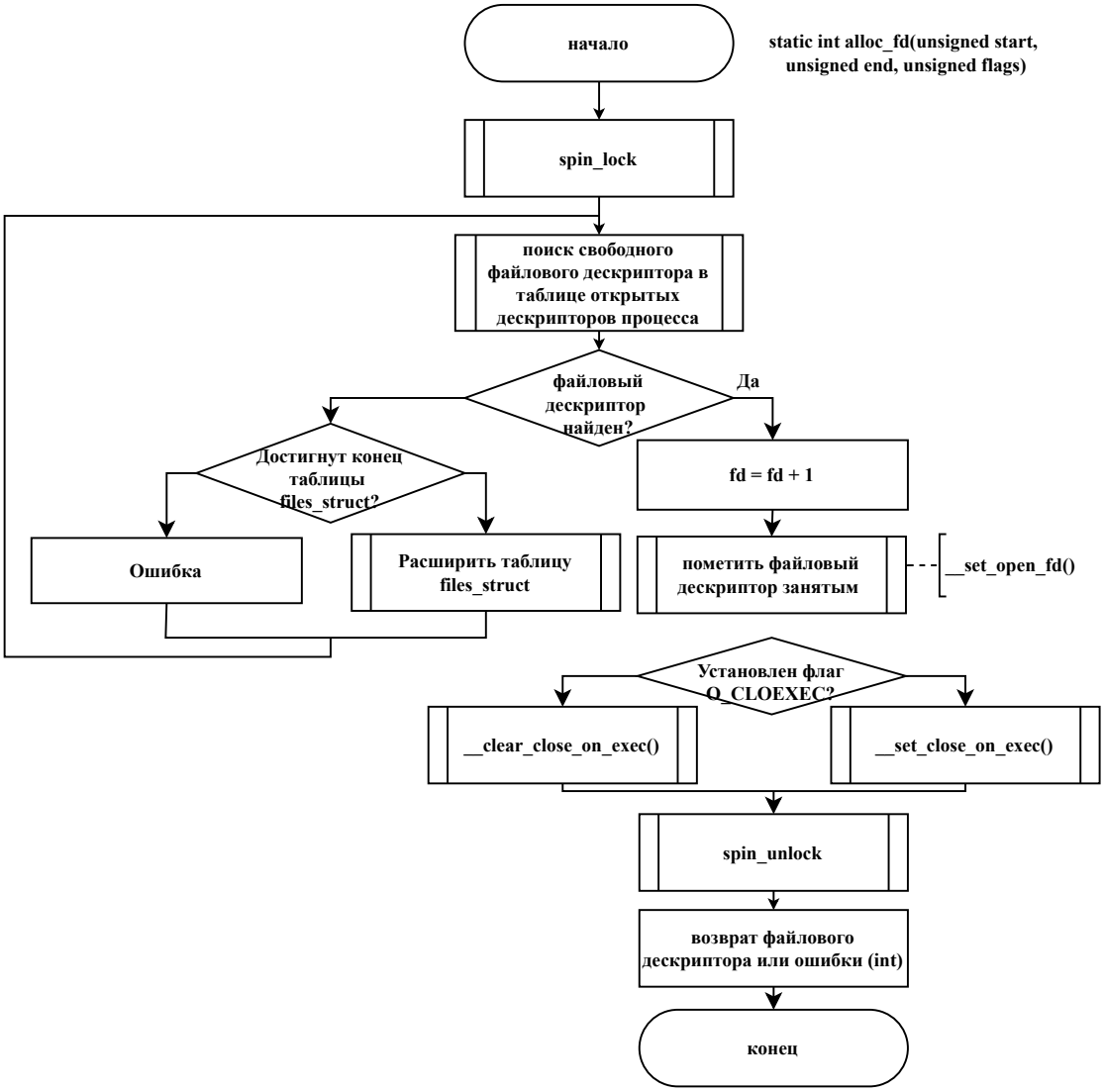

Рисунок 2.5 – Схема алгоритма работы функции `alloc_fd`

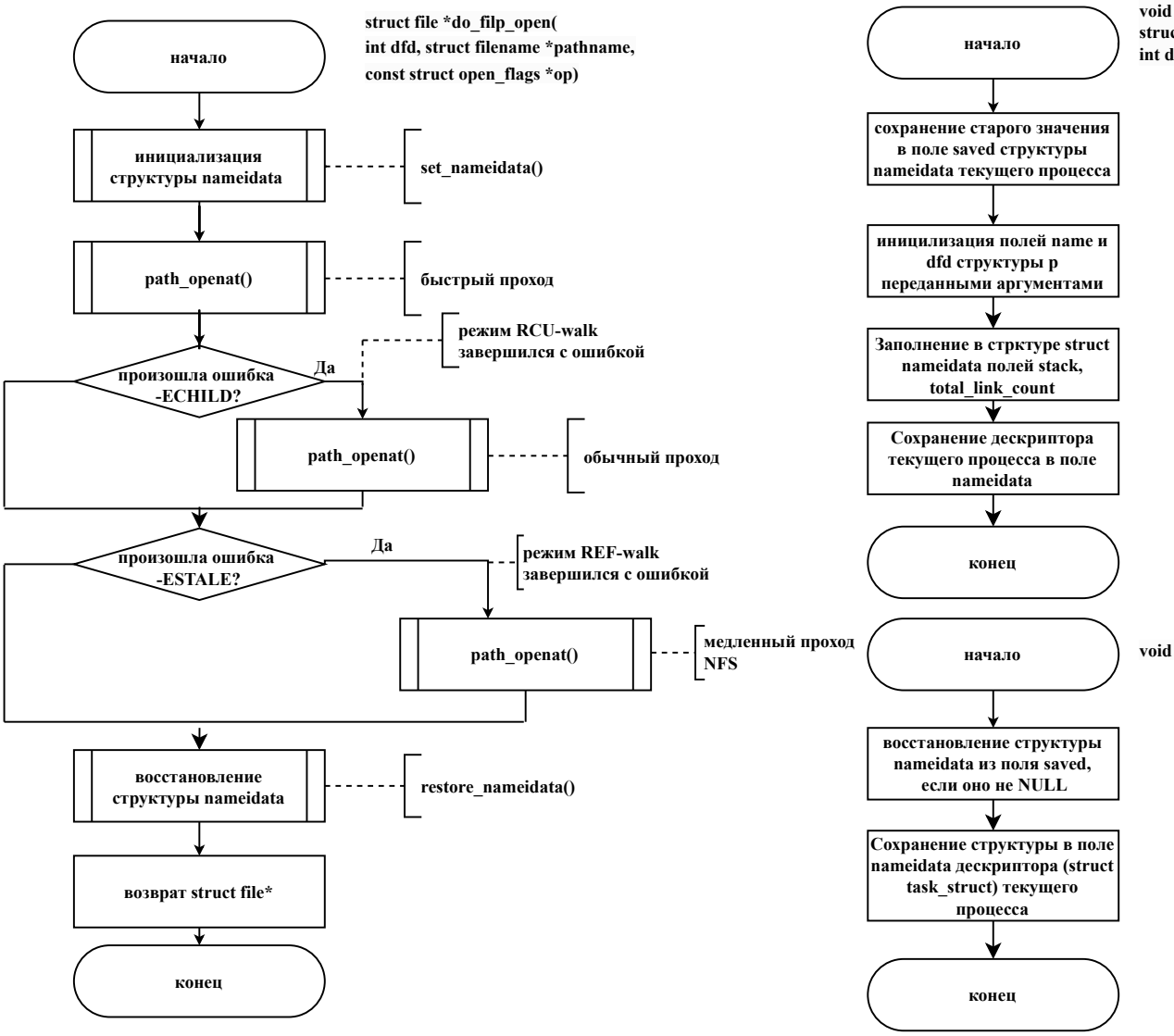

Рисунок 2.6 – Схема алгоритма работы функции `do_filp_open`

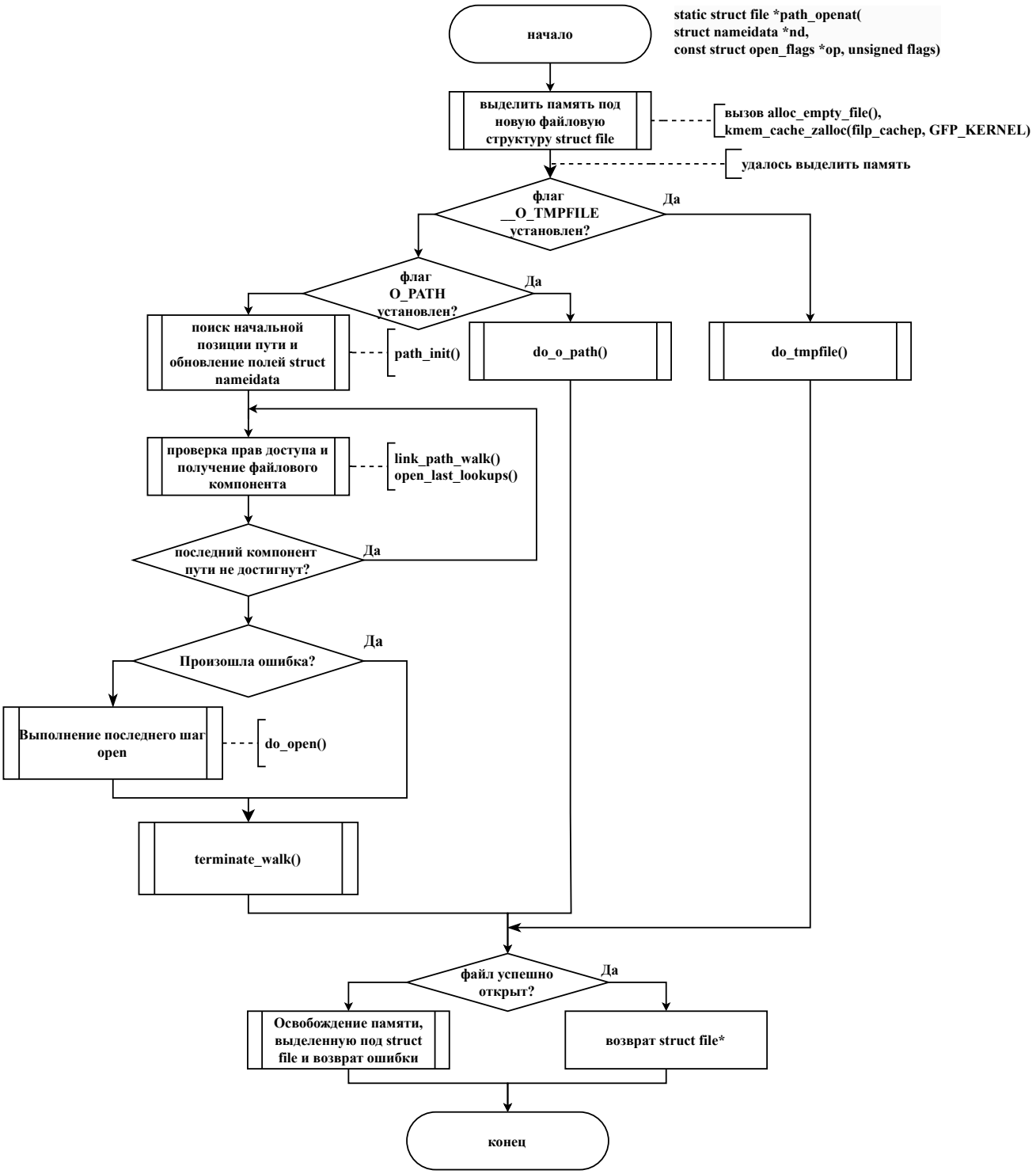

Рисунок 2.7 – Схема алгоритма работы функции `path_openat`

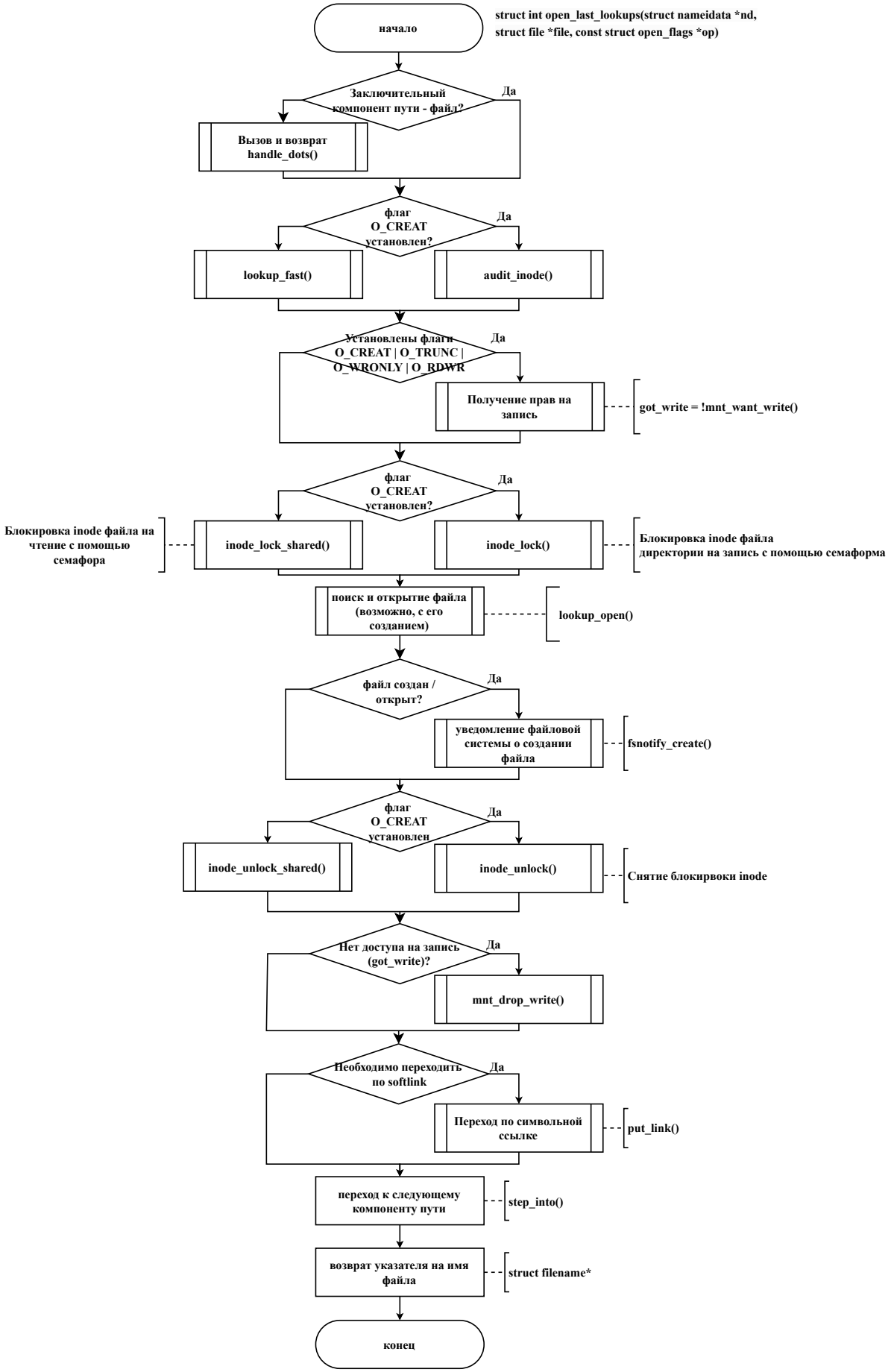

Рисунок 2.8 – Схема алгоритма работы функции `open_last_lookups`

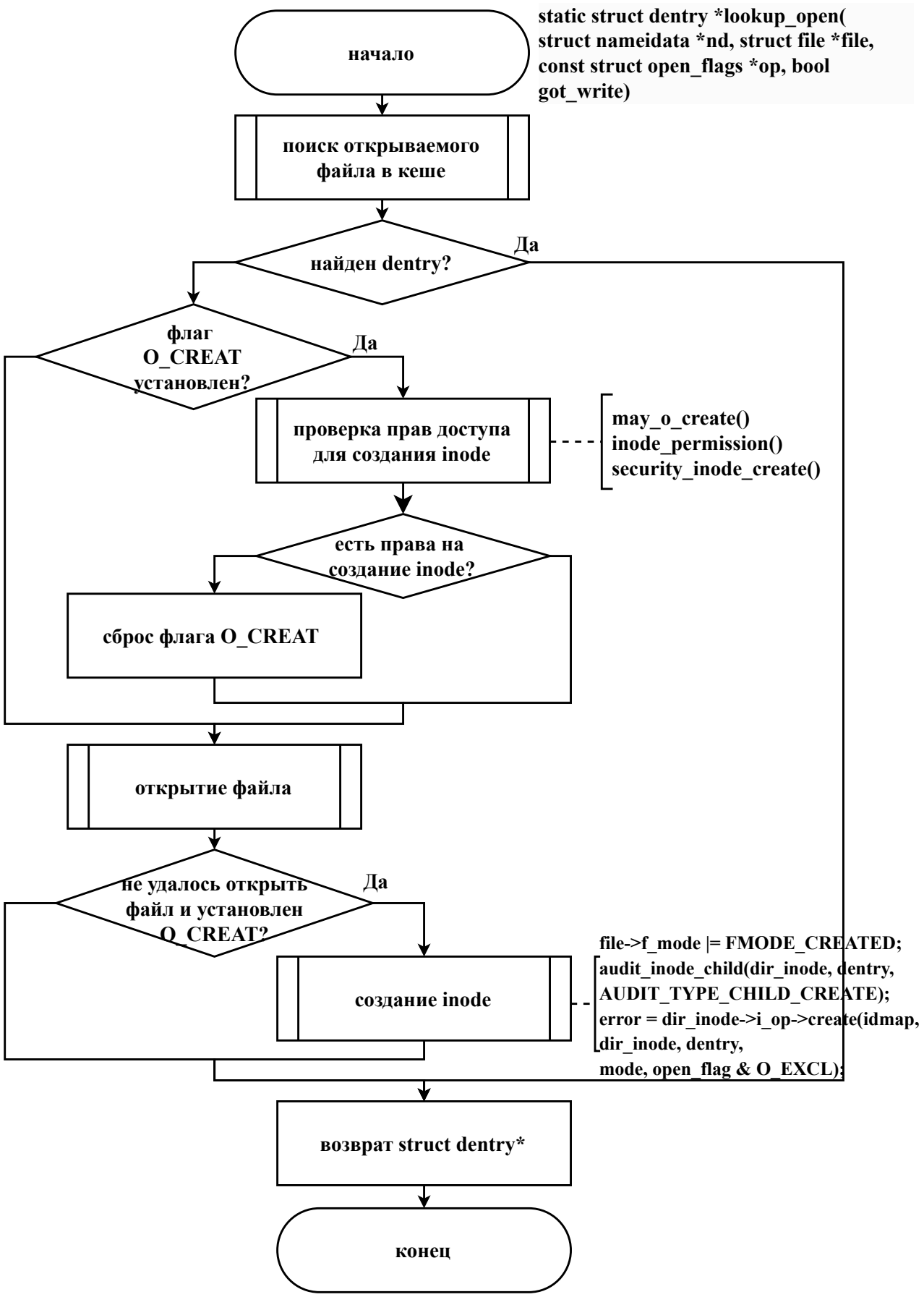

Рисунок 2.9 – Схема алгоритма работы функции `lookup_open`