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 Хамзина Р. Р.

Группа ИУ7-63Б

Оценка (баллы)

Преподаватель Рязанова Н. Ю.

Москва — 2022 г.

1 Необходимые структуры

Версия ядра: 5.13.0

```
1  struct filename {
2      const char      *name;  /* pointer to actual string */
3      const __user char   *uptr;  /* original userland pointer */
4      int          refcnt;
5      struct audit_names  *aname;
6      const char      iname[];
7  };
8
9  struct open_flags {
10     int open_flag;
11     umode_t mode;
12     int acc_mode;
13     int intent;
14     int lookup_flags;
15 };
16
17 struct audit_names {
18     struct list_head    list;       /* audit_context->names_list
           */
19
20     struct filename     *name;
21     int          name_len;   /* number of chars to log */
22     bool             hidden;     /* don't log this record */
23
24     unsigned long       ino;
25     dev_t            dev;
26     umode_t          mode;
27     kuid_t           uid;
28     kgid_t           gid;
29     dev_t            rdev;
30     u32          osid;
31     struct audit_cap_data   fcap;
32     unsigned int        fcap_ver;
33     unsigned char       type;       /* record type */
34     /*
35      * This was an allocated audit_names and not from the array of
```

```
36          * names allocated in the task audit context.  Thus this name
37          * should be freed on syscall exit.
38          */
39         bool            should_free;
40 };
41
42 #define EMBEDDED_LEVELS 2
43 struct nameidata {
44         struct path path;
45         struct qstr last;
46         struct path root;
47         struct inode    *inode; /* path.dentry.d_inode */
48         unsigned int    flags;
49         unsigned    seq, m_seq, r_seq;
50         int     last_type;
51         unsigned    depth;
52         int     total_link_count;
53         struct saved {
54             struct path link;
55             struct delayed_call done;
56             const char *name;
57             unsigned seq;
58         } *stack, internal[EMBEDDED_LEVELS];
59         struct filename *name;
60         struct nameidata *saved;
61         unsigned    root_seq;
62         int     dfd;
63         kuid_t      dir_uid;
64         umode_t     dir_mode;
65 } __randomize_layout;
66
67 struct open_how {
68         __u64 flags;
69         __u64 mode;
70         __u64 resolve;
71 };
72
73 struct path {
74         struct vfsmount *mnt;
75         struct dentry *dentry;
76 } __randomize_layout;
```

2 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.

Схема алгоритма работы системного вызова open()

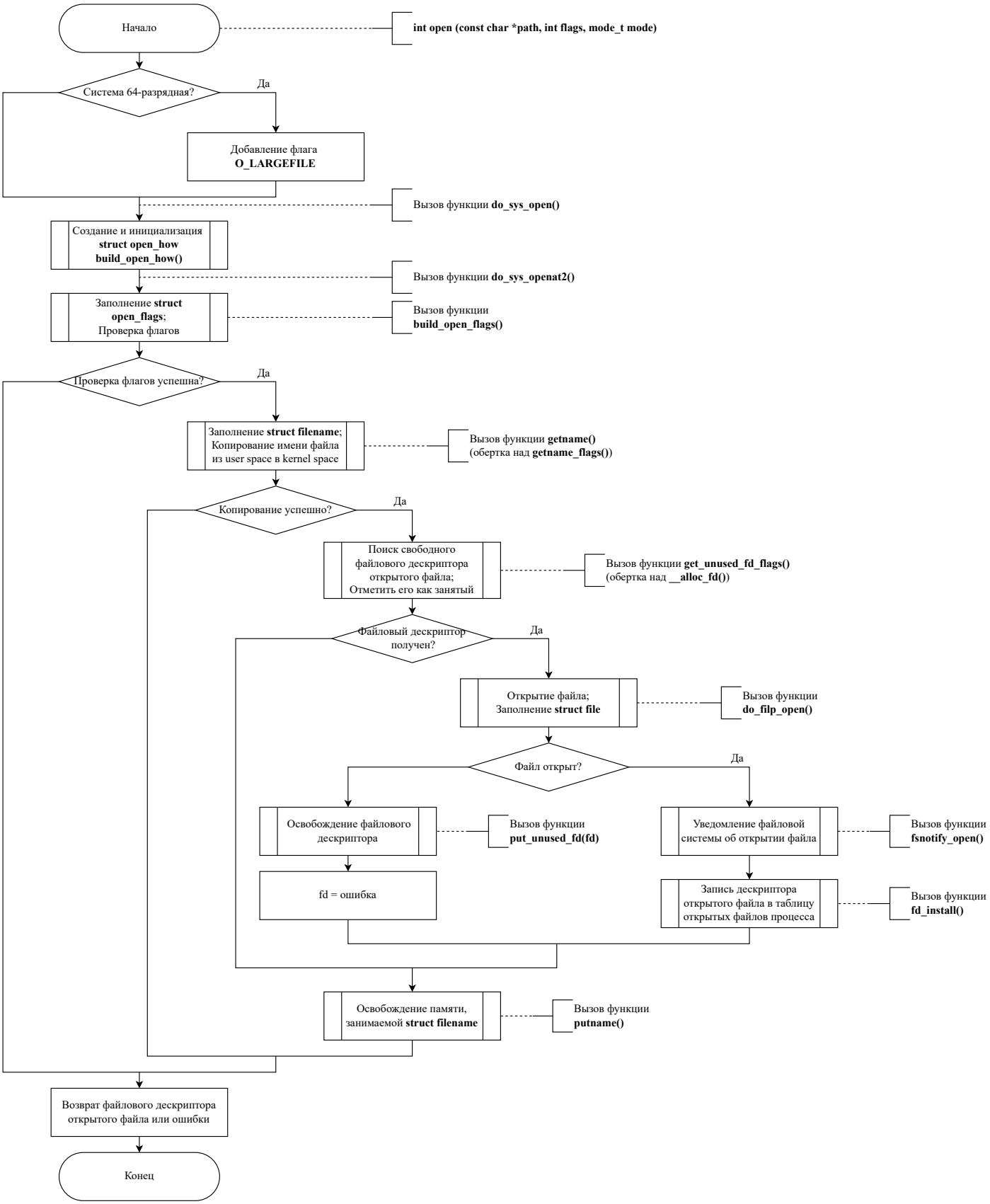

Схема алгоритма работы функции build_open_how()

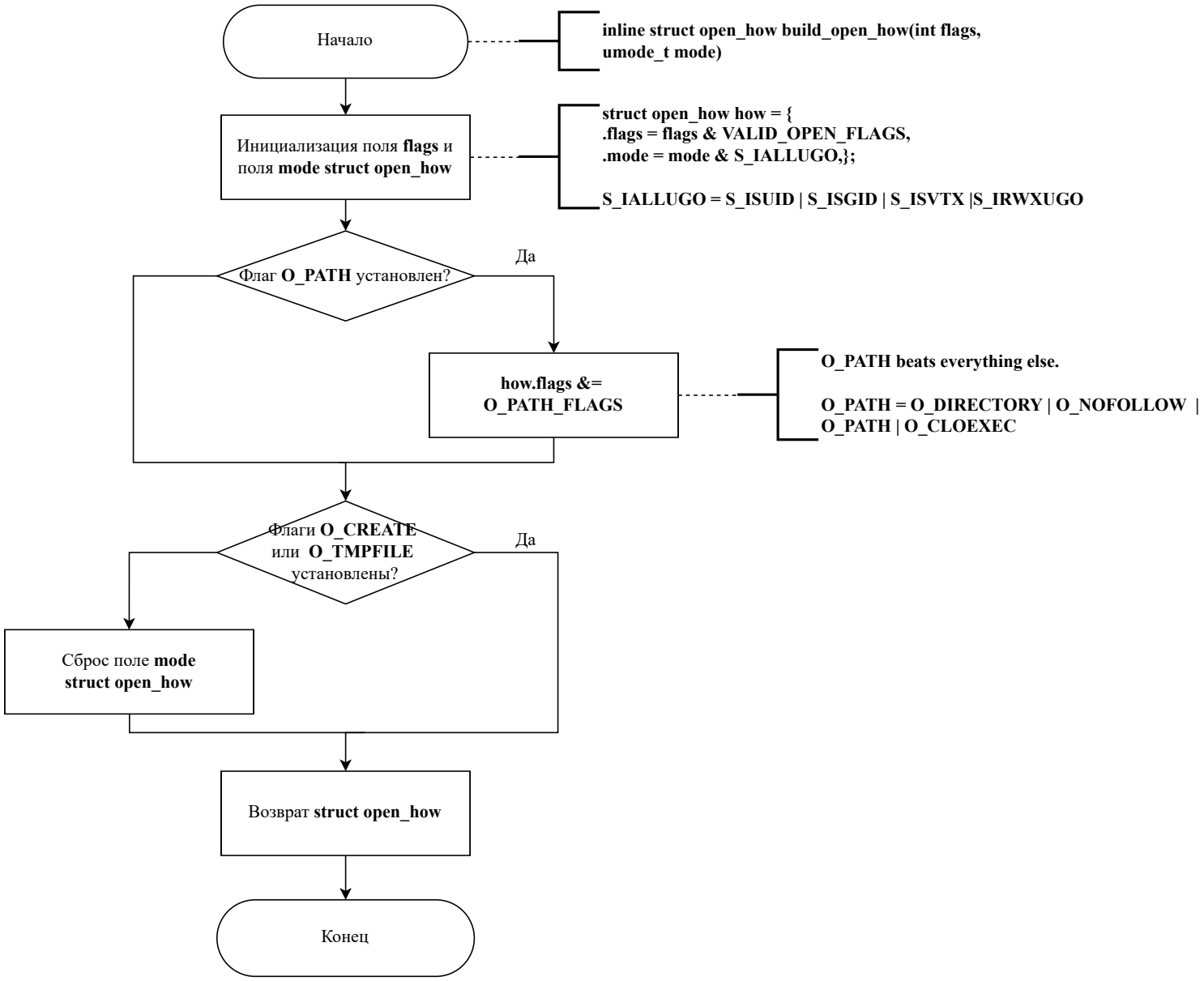

Схема алгоритма работы функции build_open_flags()

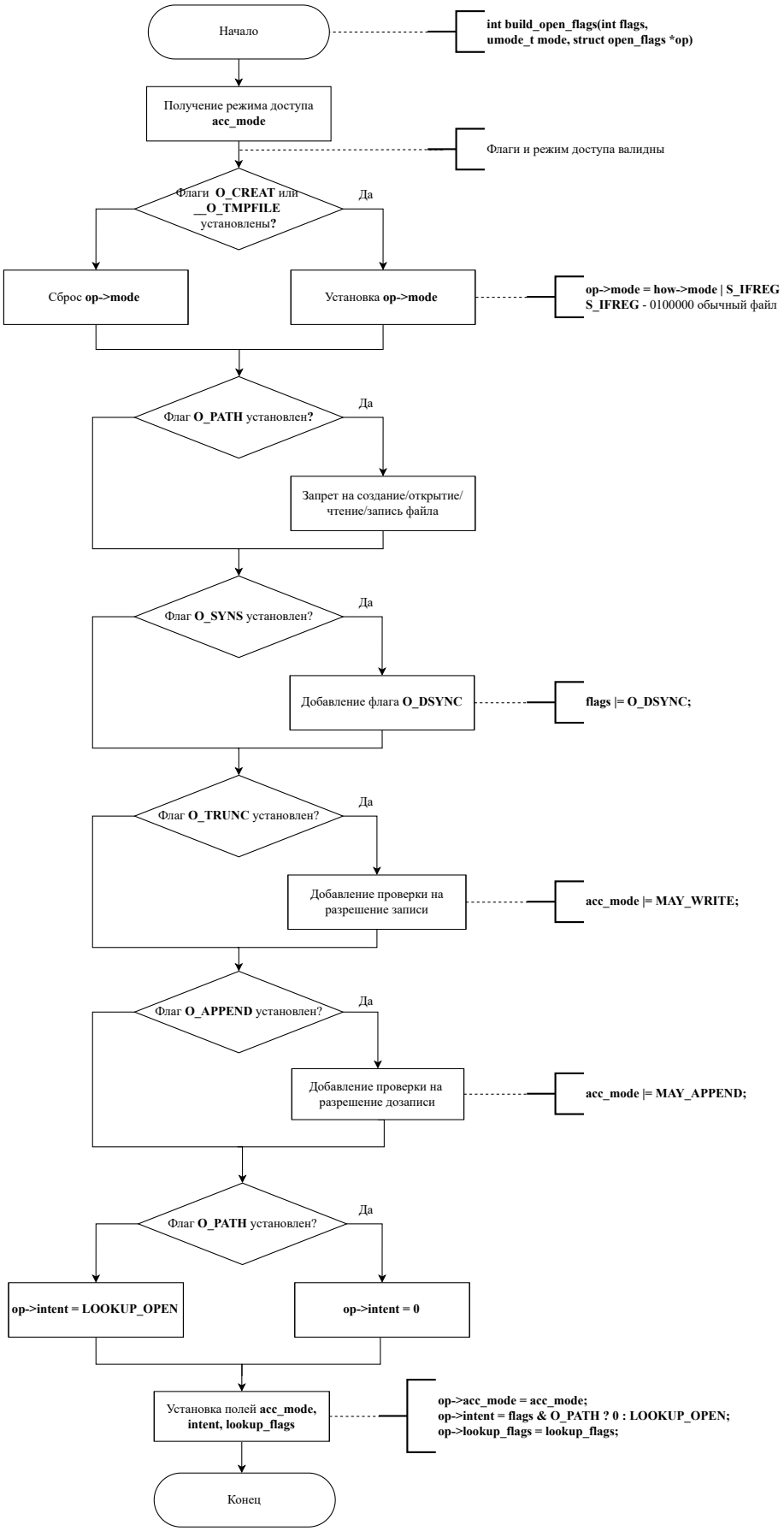

Схема алгоритма работы функции getname_flags()

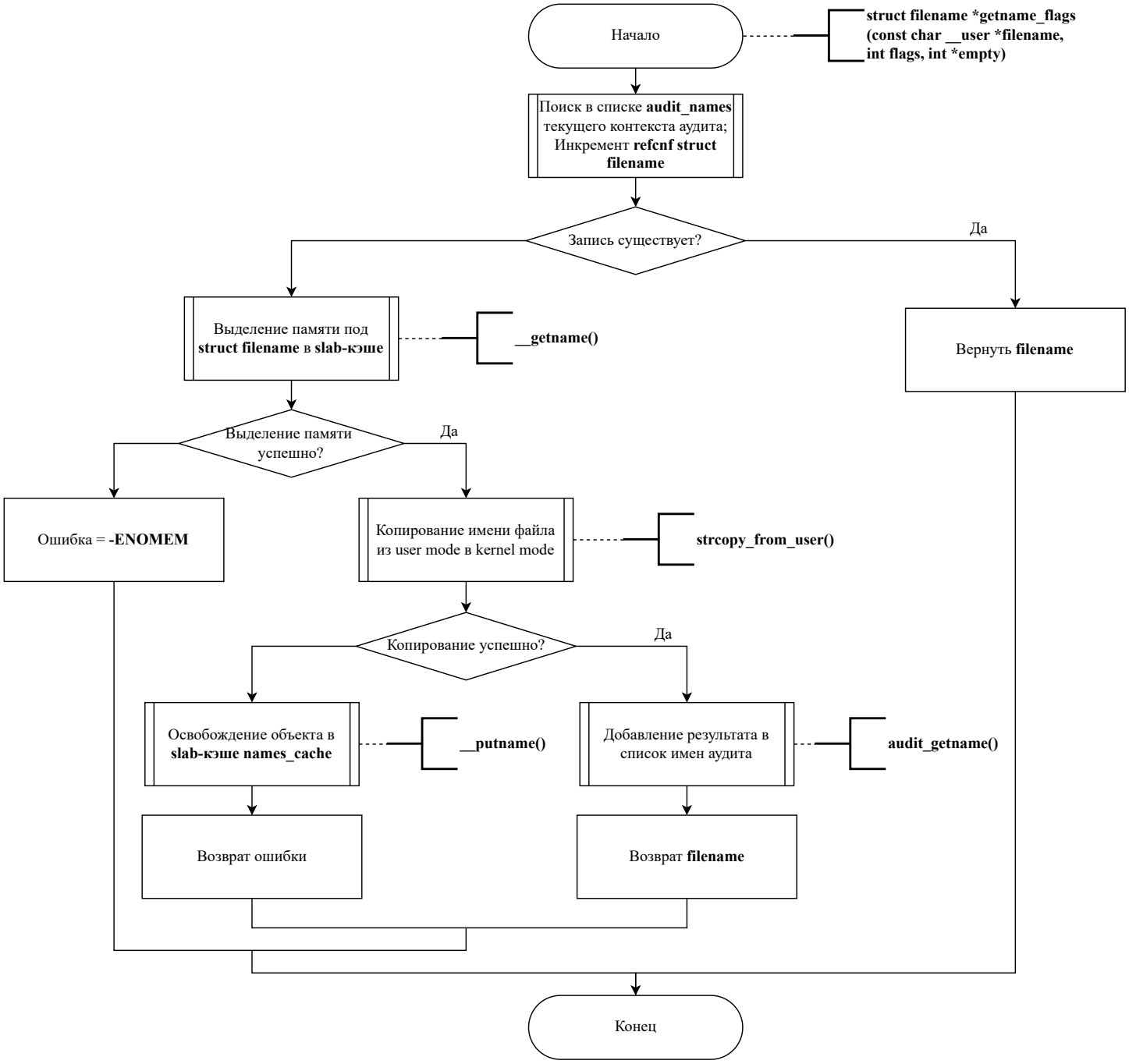

Схема алгоритма работы функции alloc_fd()

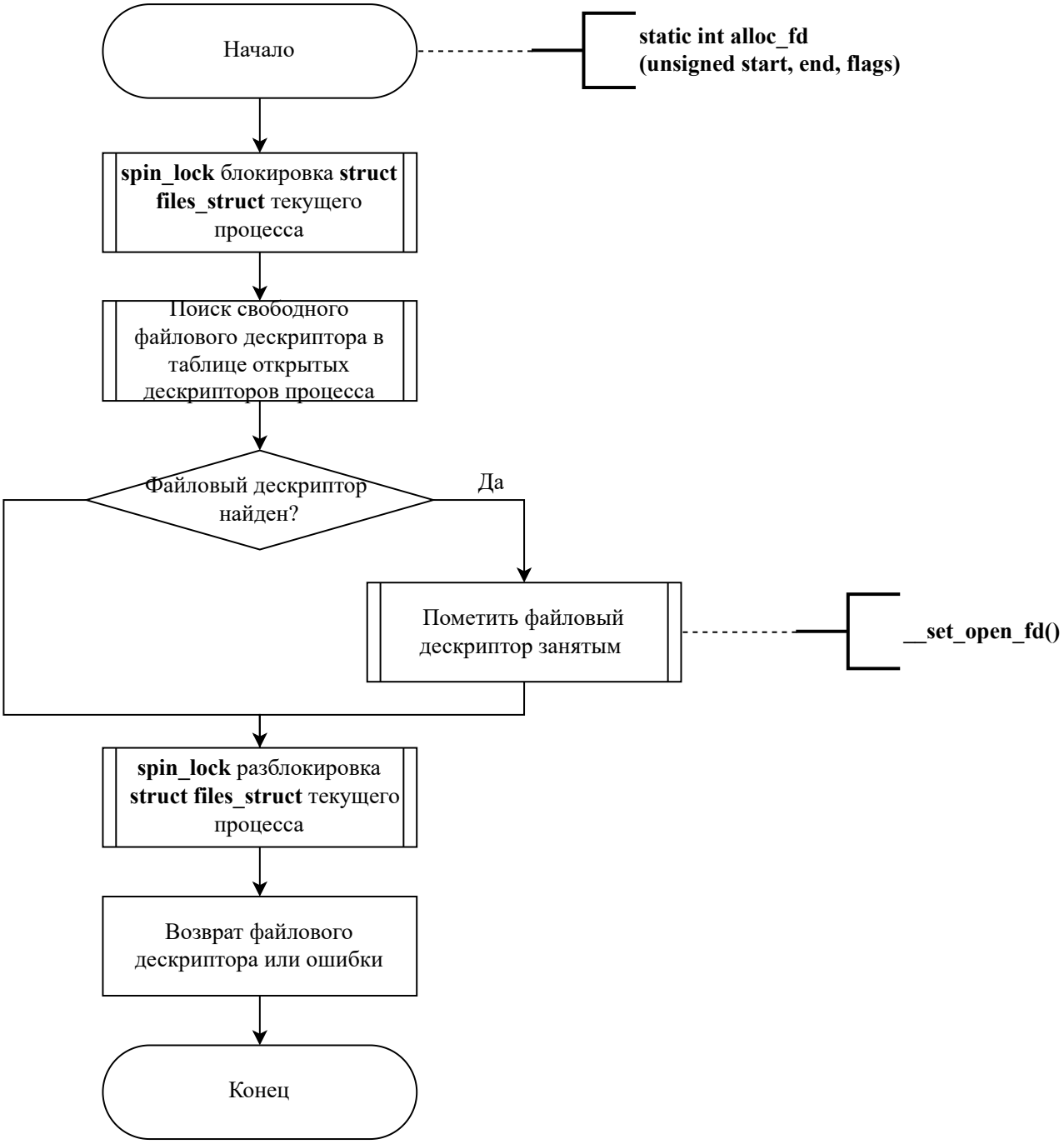

Схема алгоритма работы функции do_filp_open()

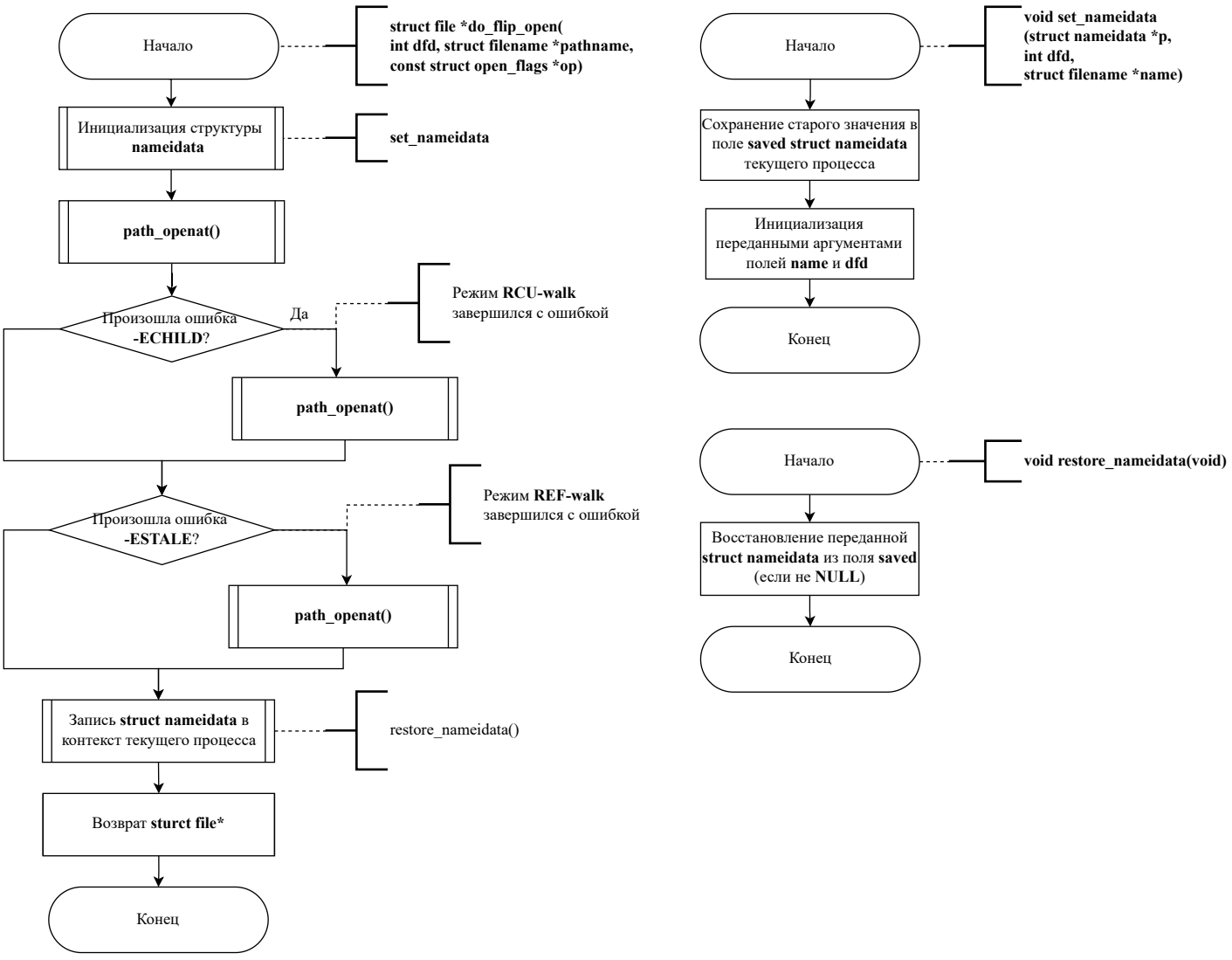

Схема алгоритма работы функции path_openat()

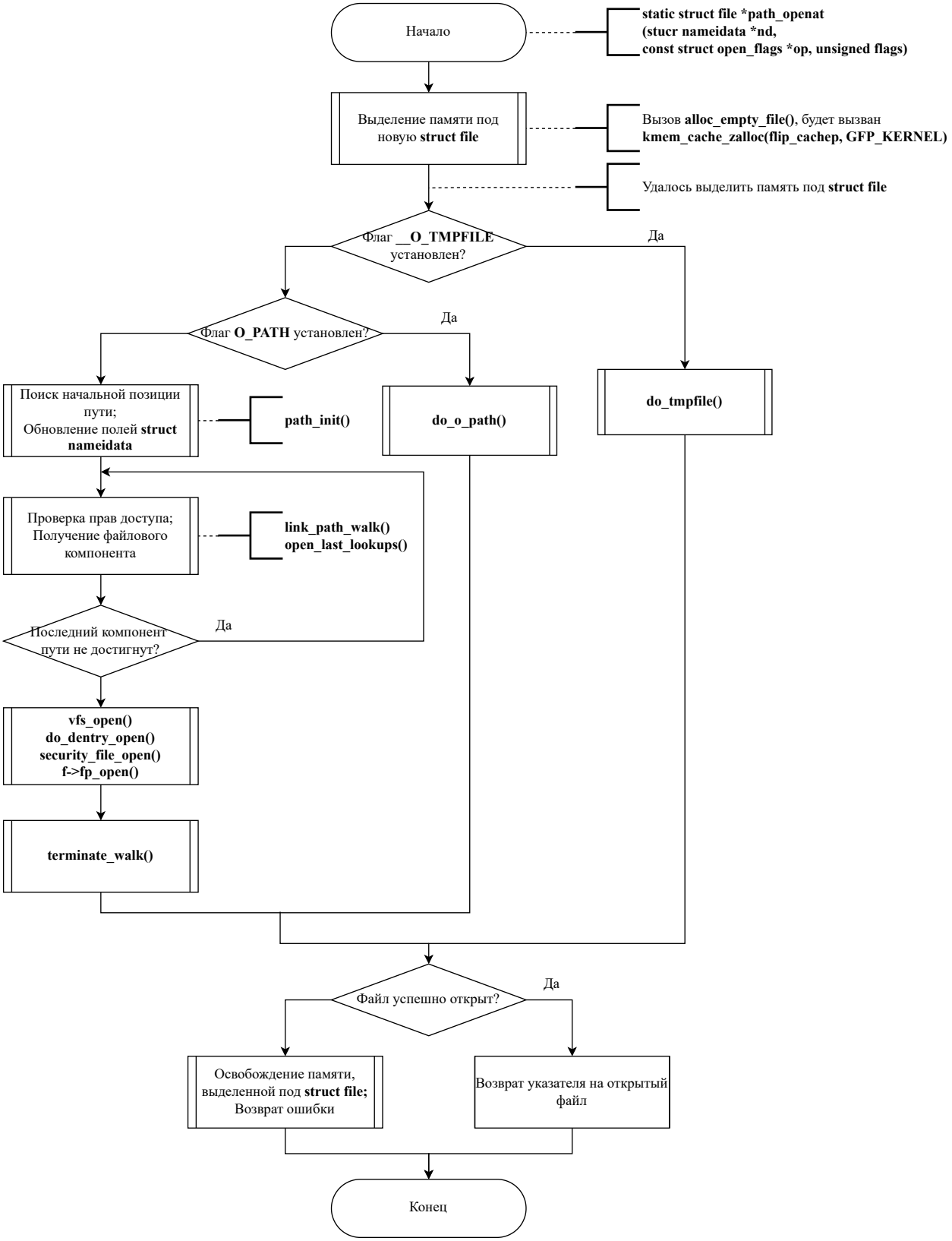

Схема алгоритма работы функции open_last_lookups()

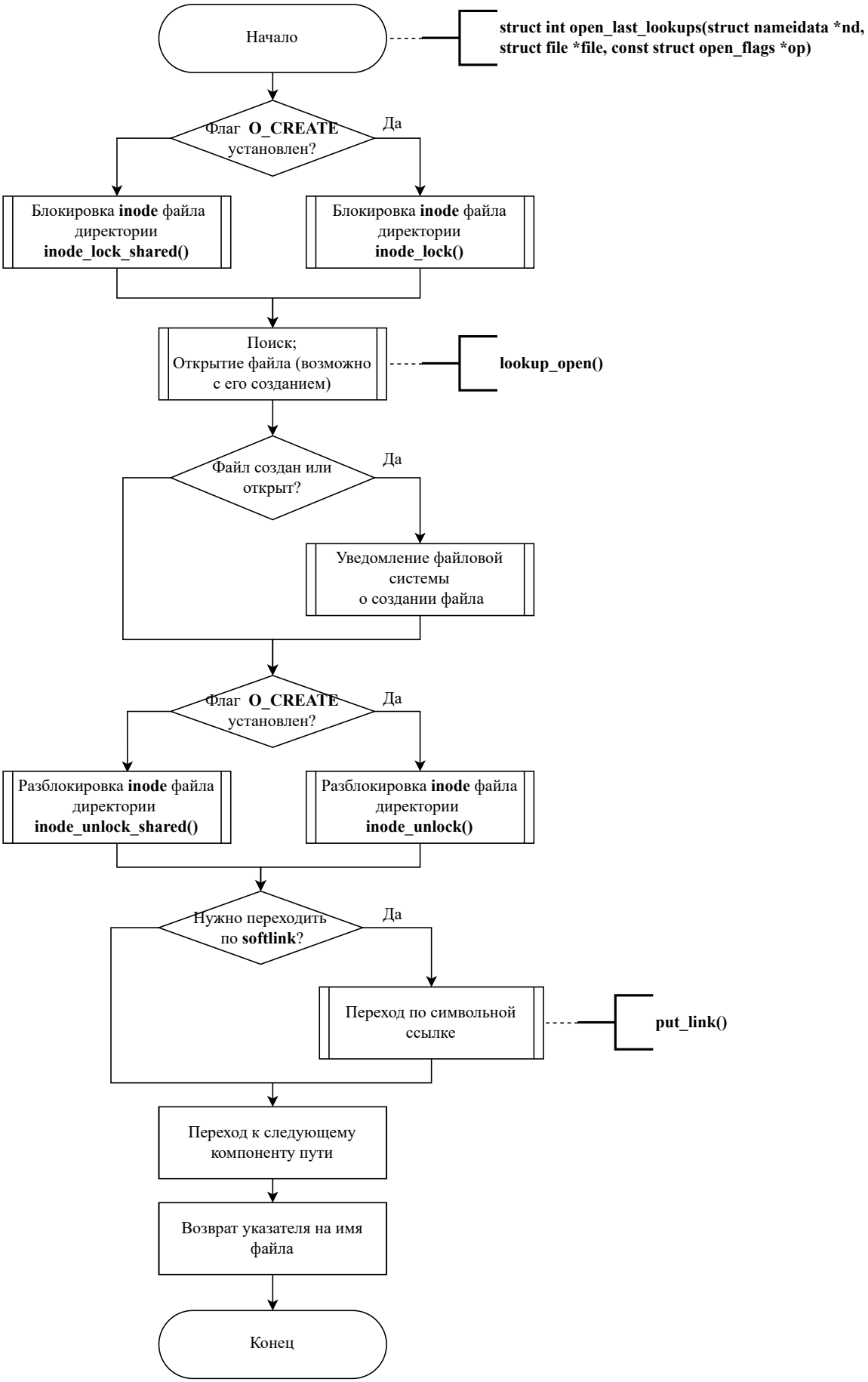

Схема алгоритма работы функции lookup_open()