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 №9 по курсу "Операционные системы"

Тема Системный вызов open()

Студент Малышев И.Н.

Группа ИУ7-64Б

Оценка (баллы)

Преподаватель Рязанова Н. Ю.

Москва — 2023 г.

1 Системный вызов open()

Системный вызов open() открывает файл, указанный в pathname. Если указанный файл не существует, он может (необязательно) (если указан флаг O_CREATE) был создан open().

Листинг 1.1 – Системный вызов open()

```
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);
```

Возвращаемое значение open() — дескриптор файла, неотрицательное целое число, которое используется в последующих системных вызовах для работы с файлом.

Параметр pathname — это имя файла в файловой системе: полный путь к файлу или сокращенное имя.

Параметр flags — это режим открытия файла, представляющий собой один или несколько флагов открытия, объединенных оператором побитового ИЛИ. Список доступных флагов:

- O_EXEC — открыть только для выполнения (результат не определен, при открытии директории).
- O_RDONLY — открыть только на чтение.
- O_RDWR — открыть на чтение и запись.
- O_SEARCH — открыть директорию только для поиска (результат не определен, при использовании с файлами, не являющимися директорией).
- O_WRONLY — открыть только на запись.

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

— 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.

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

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

— O_DSYNC — файл открывается в режиме синхронного ввода-вывода (все операции записи для соответствующего дескриптора файла блокируют вызывающий процесс до тех пор, пока данные не будут физически записаны).

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

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

— O_RSYNC — операции записи должны выполняться на том же уровне, что и O_SYNC.

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

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

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

Параметр mode всегда должен быть указан при использовании O_CREATE; во всех остальных случаях этот параметр игнорируется.

2 Используемые структуры

Листинг 2.1 – struct open_flags

```
1     struct open_flags {
2         int          open_flag;
3         umode_t      mode;
4         int          acc_mode;
5         int          intent;
6         int lookup_  flags;
7     };
```

Листинг 2.2 – struct filename и struct audit_names

```
1     struct audit_names;
2     struct filename {
3         const char          *name;  /* pointer to actual string */
4         const __user char   *uptr;  /* original userland pointer */
5         int                 refcnt;
6         struct audit_names  *aname;
7         const char          iname[];
8     };
9
10    struct audit_names {
11        struct list_head list;     /* audit_context -> names_list */
12        struct filename  *name;
13        int              name_len; /* number of chars to log */
14        bool             hidden;
15        unsigned long    ino;
16        dev_t            dev;
17        umode_t          mode;
18        kuid_t           uid;
19        kgid_t           gid;
20        dev_t            rdev;
21        u32              osid;
22        struct audit_cap_data fcap;
23        unsigned int          fcap_ver;
24        unsigned char         type;
25        bool                  should_free;
26    };
```

Листинг 2.3 – struct nameidata

```
1   struct nameidata {
2       struct path         path;
3       struct qstr         last;
4       struct path         root;
5       struct inode        *inode;
6       unsigned int        flags, state;
7       unsigned            seq, m_seq, r_seq;
8       int last_           type;
9       unsigned            depth;
10      int total_          link_count;
11      struct saved {
12          struct path link;
13          struct delayed_call done;
14          const char *name;
15          unsigned seq;
16      } *stack, internal[EMBEDDED_LEVELS];
17      struct filename     *name;
18      struct nameidata    *saved;
19      unsigned            root_ seq;
20      int                 dfd;
21      kuid_t              dir_uid;
22      umode_t             dir_mode;
23  } __randomize_layout;
```

3 Схемы алгоритмов

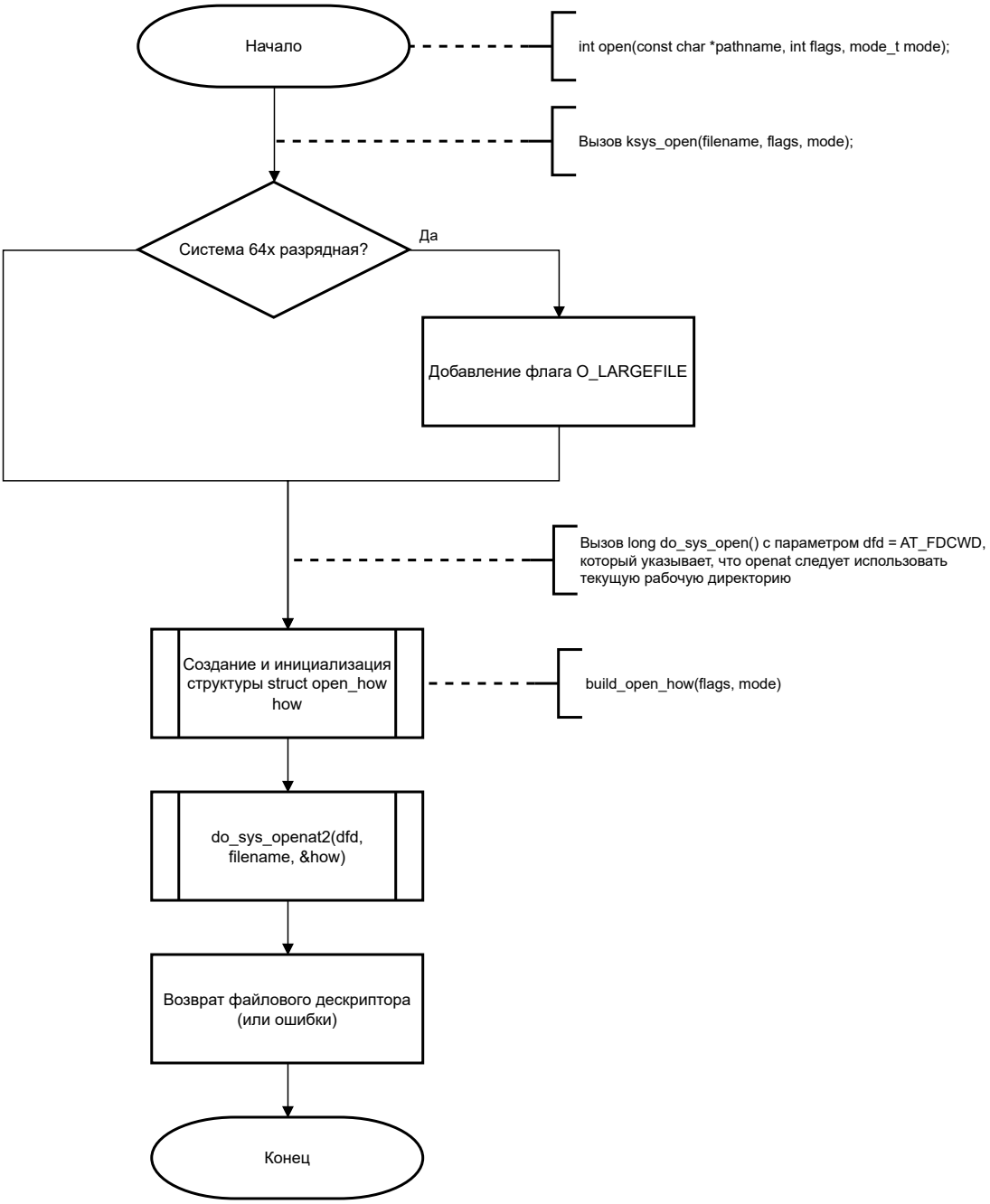

Рисунок 3.1 – Схема алгоритма работы системного вызова open()

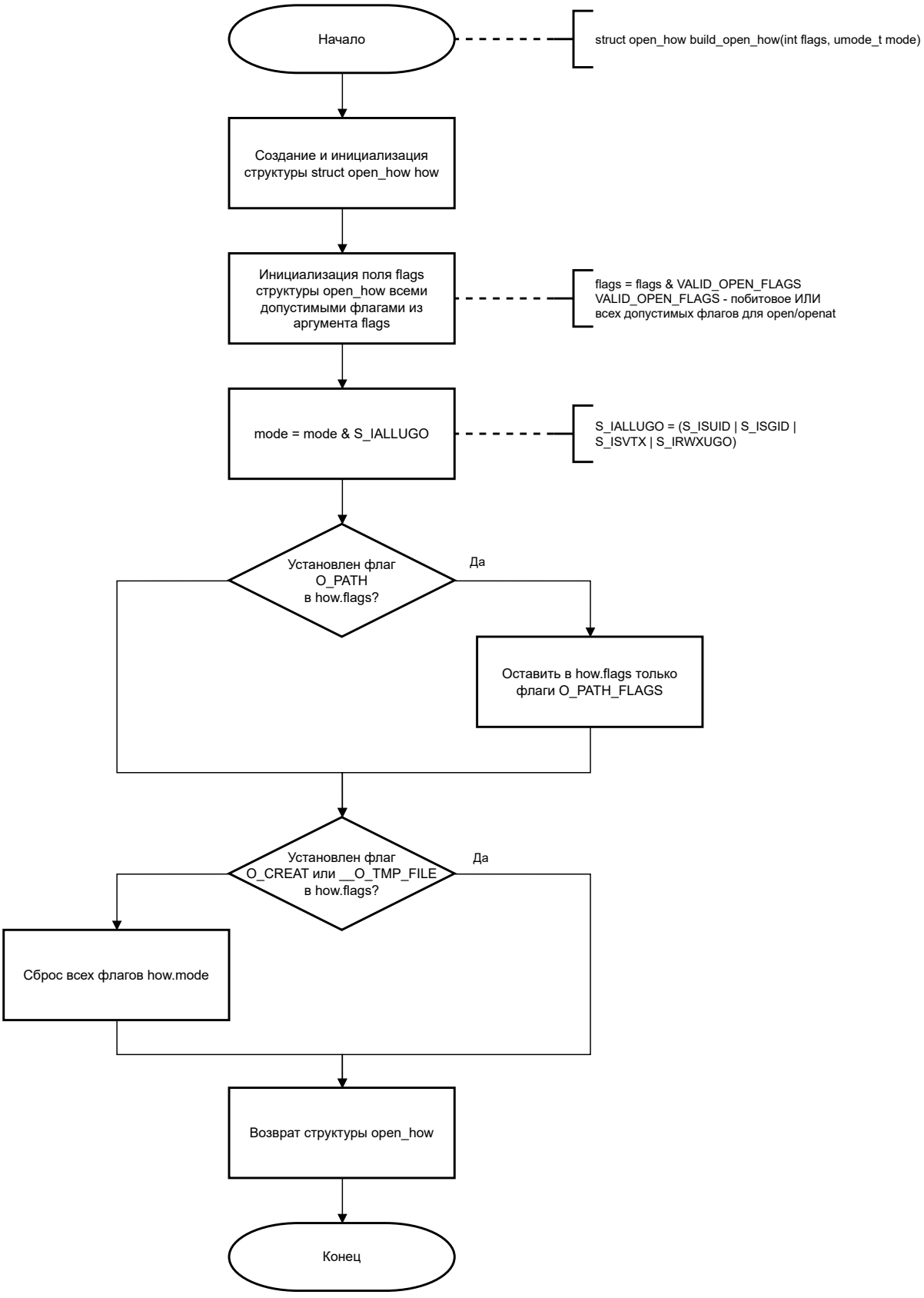

Рисунок 3.2 – Схема алгоритма работы функции build_open_how()

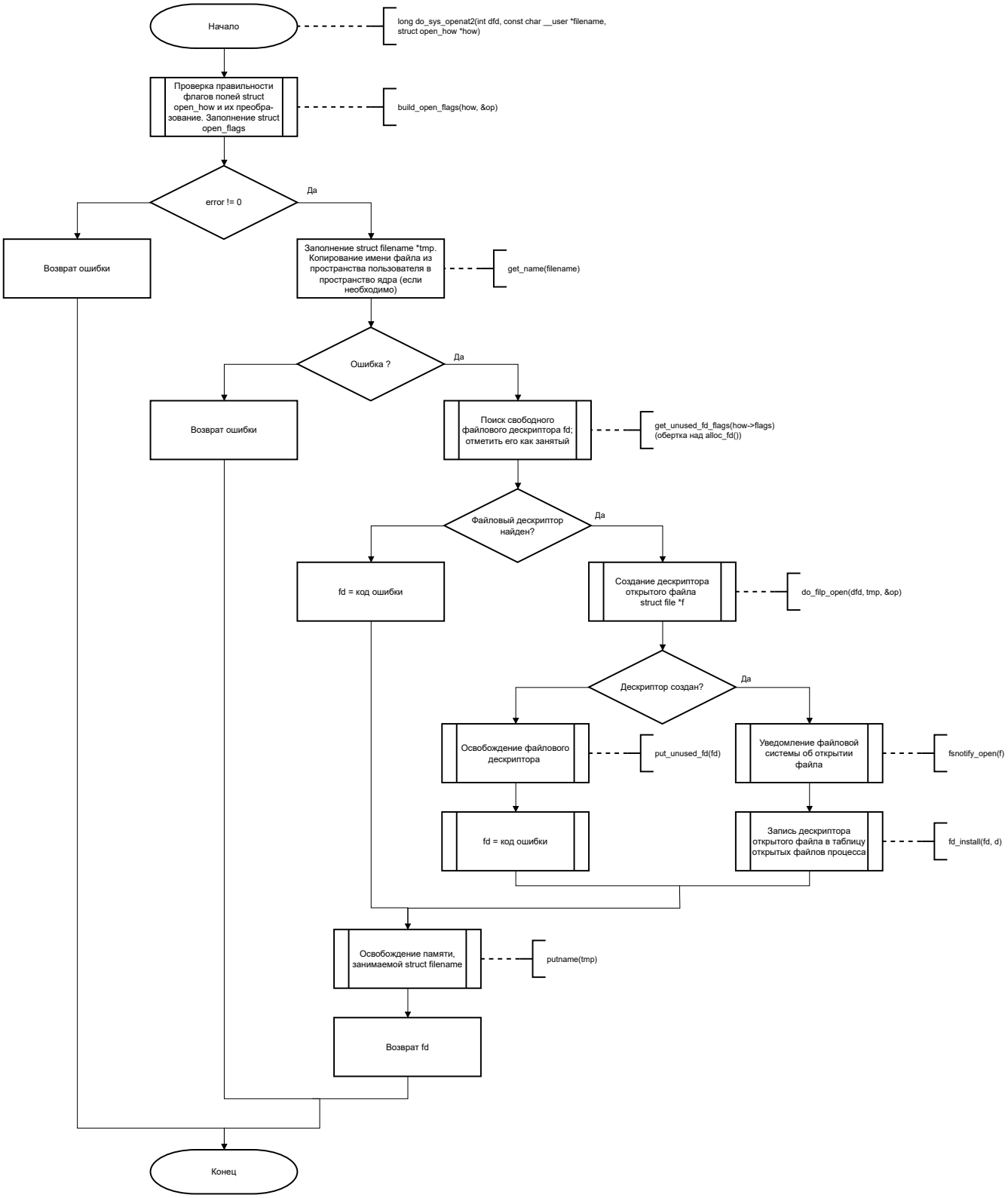

Рисунок 3.3 – Схема алгоритма работы функции do_sys_openat2()

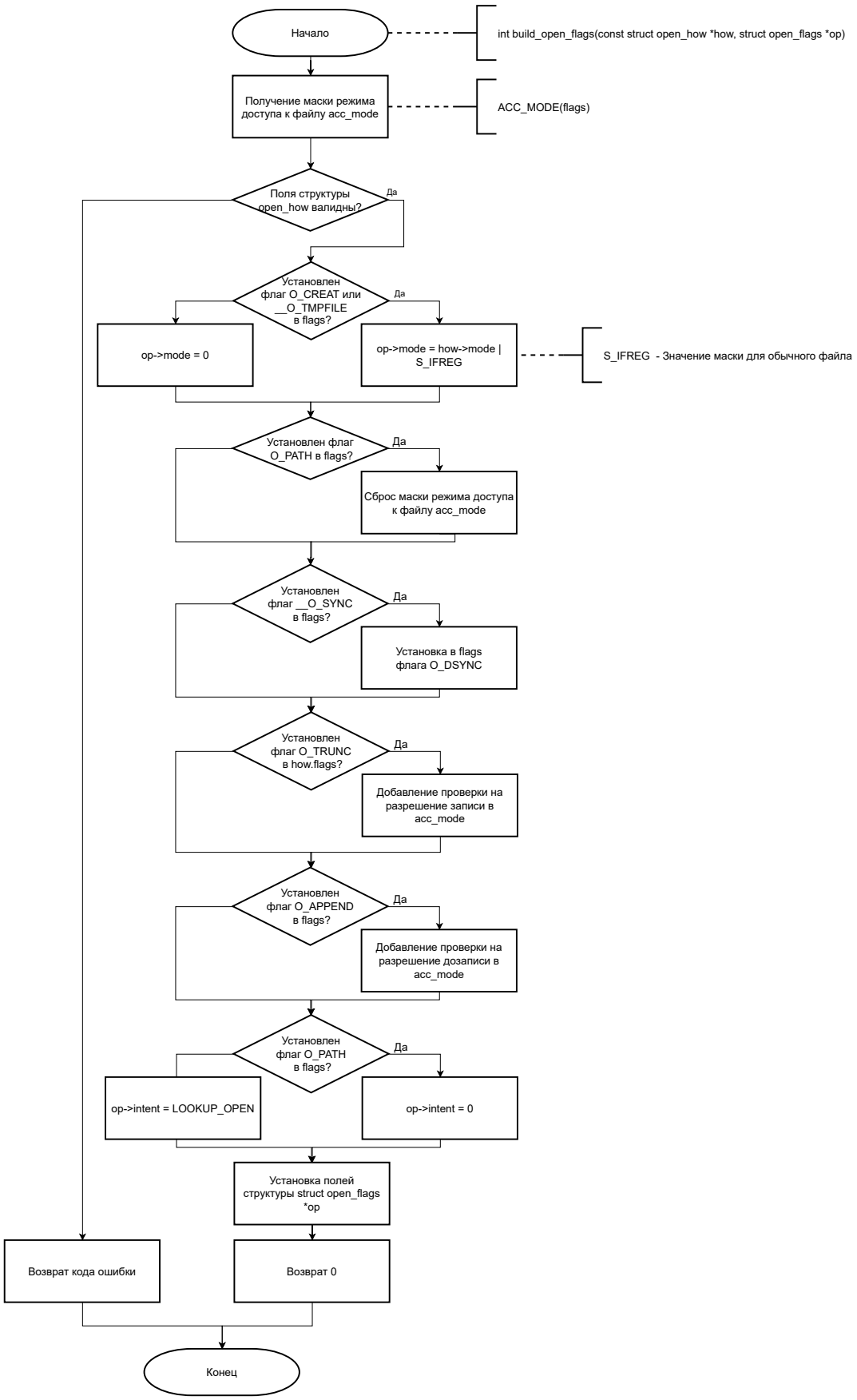

Рисунок 3.4 – Схема алгоритма работы функции build_open_flags()

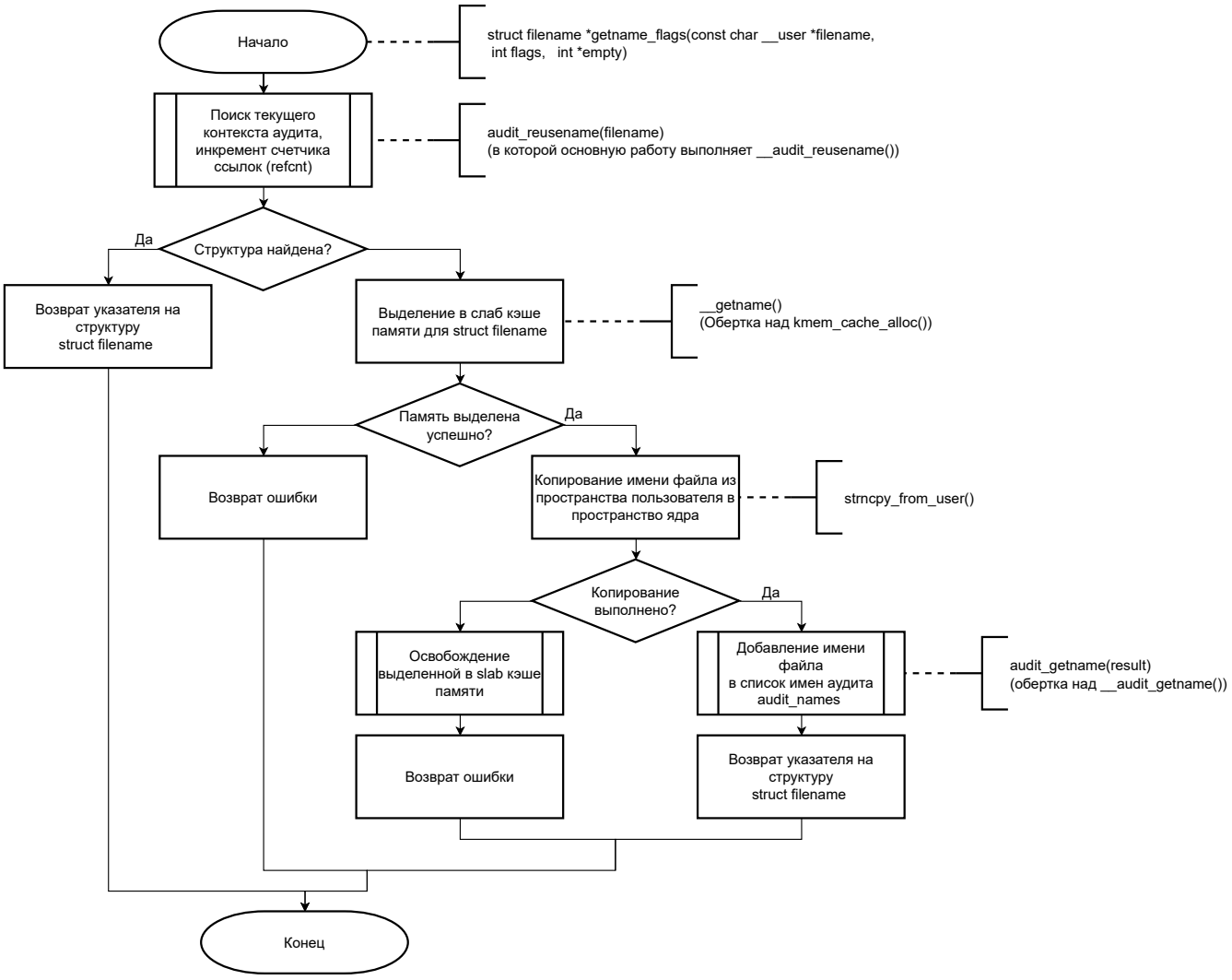

Рисунок 3.5 – Схема алгоритма работы функции getname_flags()

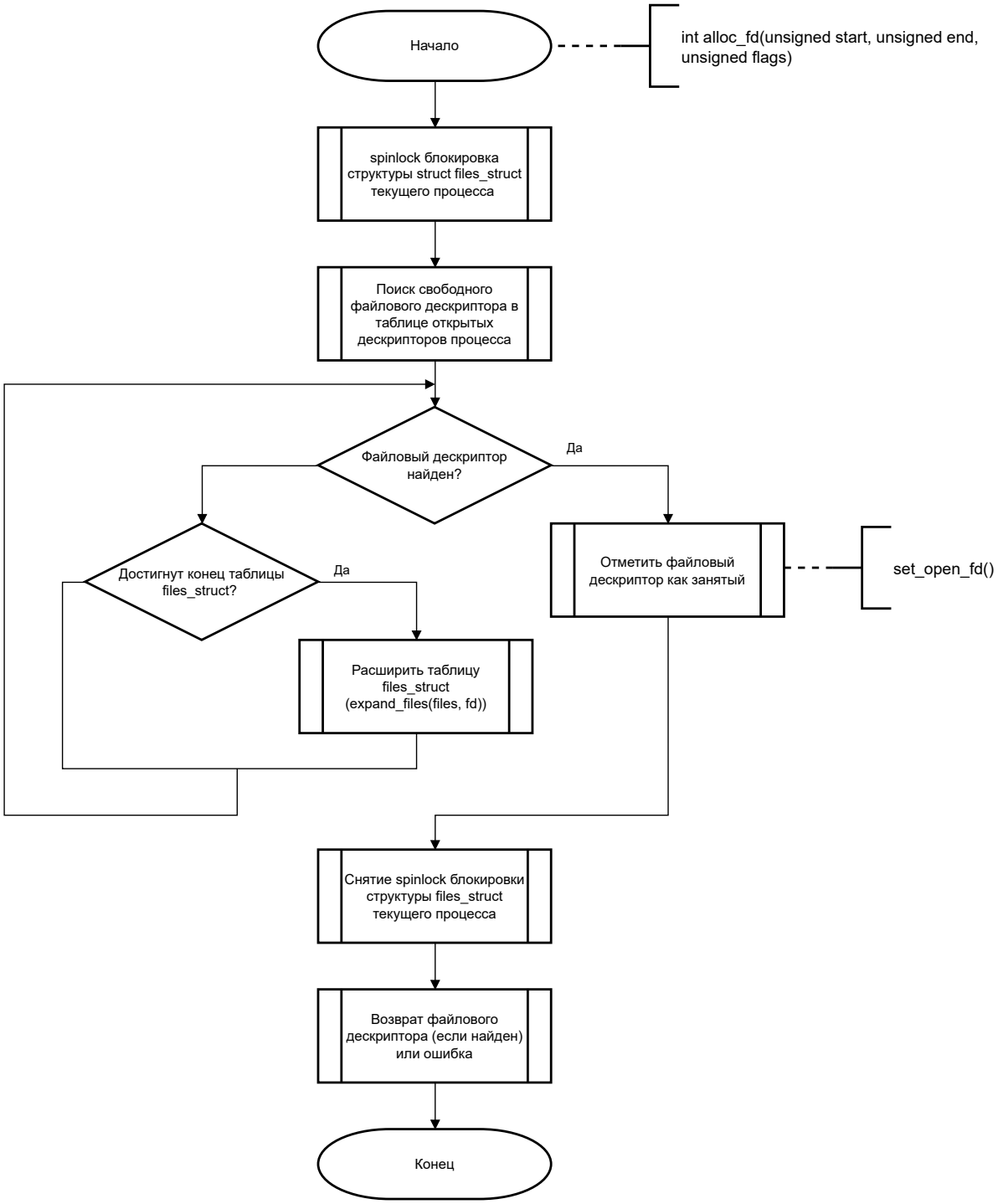

Рисунок 3.6 – Схема алгоритма работы функции alloc_fd()

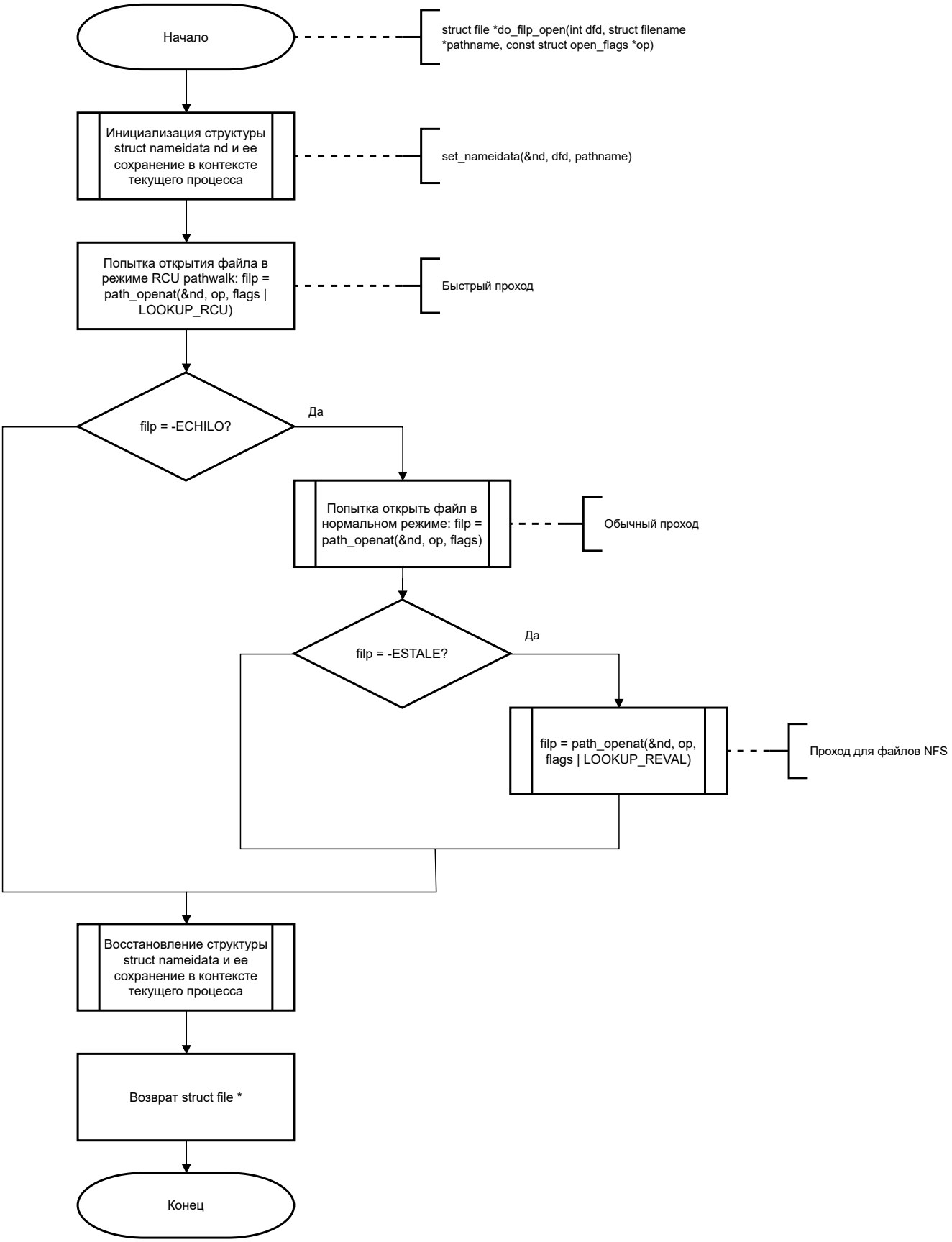

Рисунок 3.7 – Схема алгоритма работы функции do_filp_open()

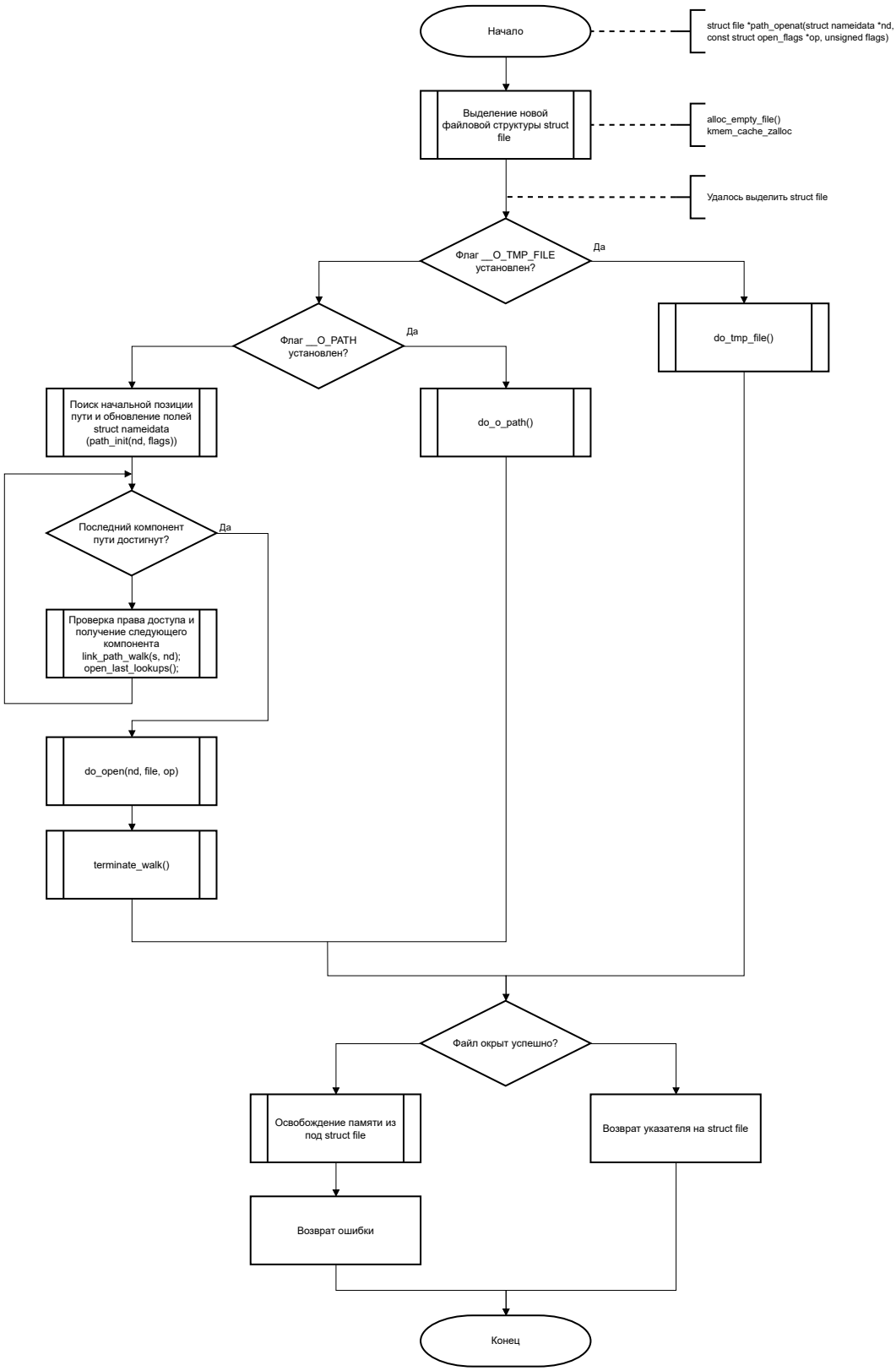

Рисунок 3.8 – Схема алгоритма работы функции path_openat()

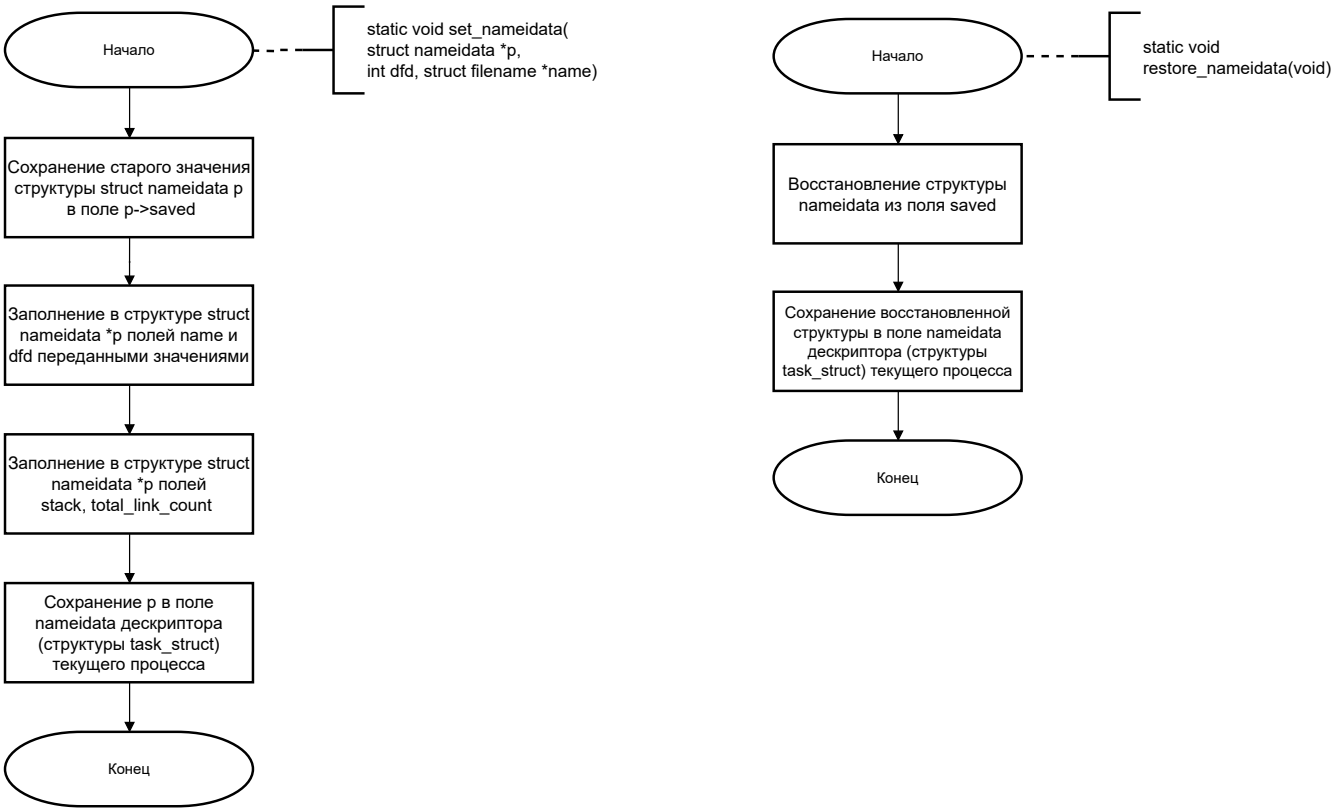

Рисунок 3.9 – Схема алгоритмов функций, работающих с nameidata

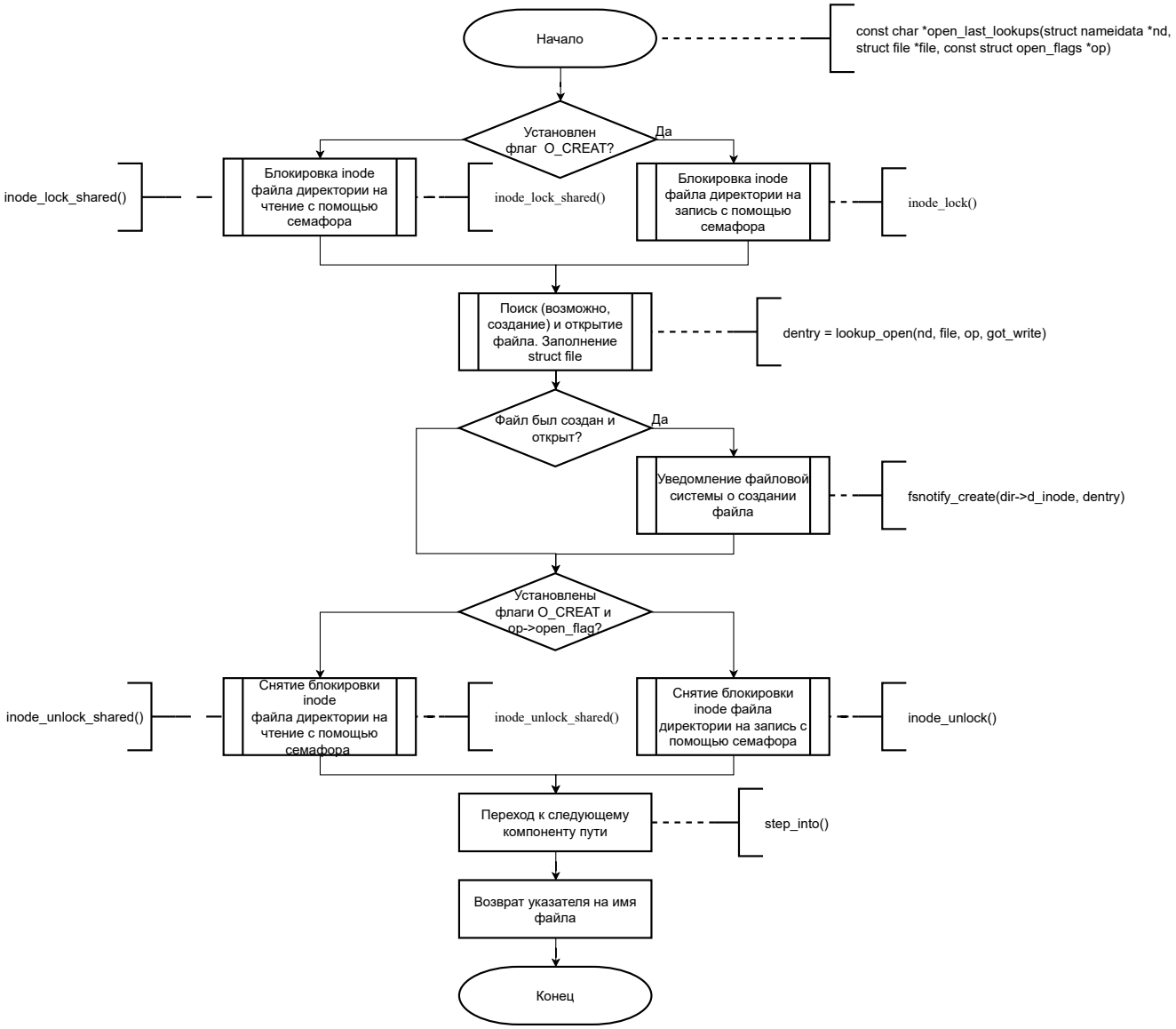

Рисунок 3.10 – Схема алгоритма работы функции open_last_lookups()

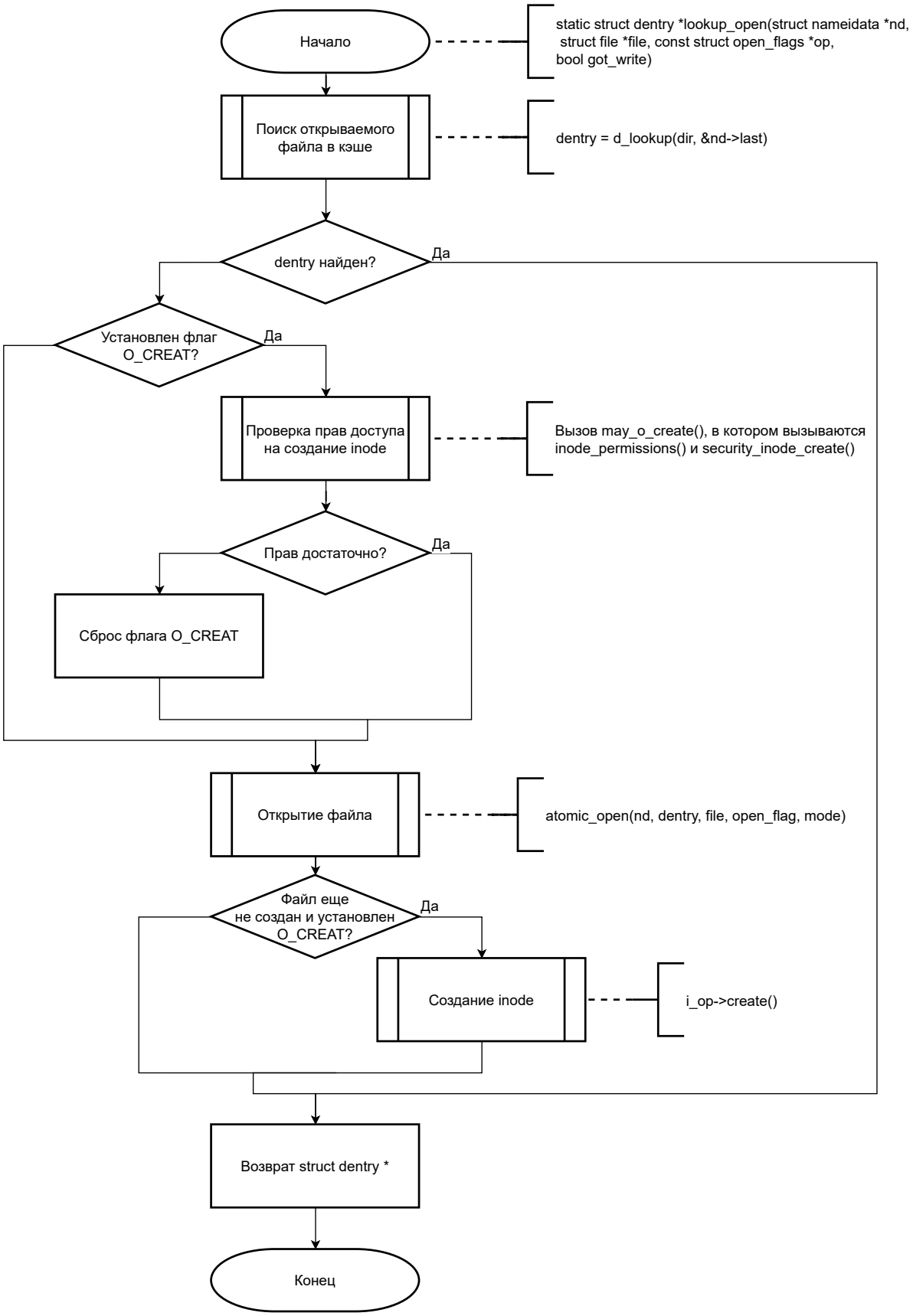

Рисунок 3.11 – Схема алгоритма работы функции lookup_open()