



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 №5 по курсу "Операционные системы"

Тема Системный вызов open

Студент Калашков П. А.

Группа ИУ7-66Б

Оценка (баллы)

Преподаватели Рязанова Н. Ю.

ИСПОЛЬЗУЕМЫЕ СТРУКТУРЫ

Листинг 1 – Структура 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 – Структура filename

```
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 };
```

Листинг 3 – Структура nameidata

```
1 struct nameidata {
2     struct path path;
3     struct qstr last;
4     struct path root;
5     struct inode *inode; /* path.dentry.d_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;
```

СХЕМЫ АЛГОРИТМОВ

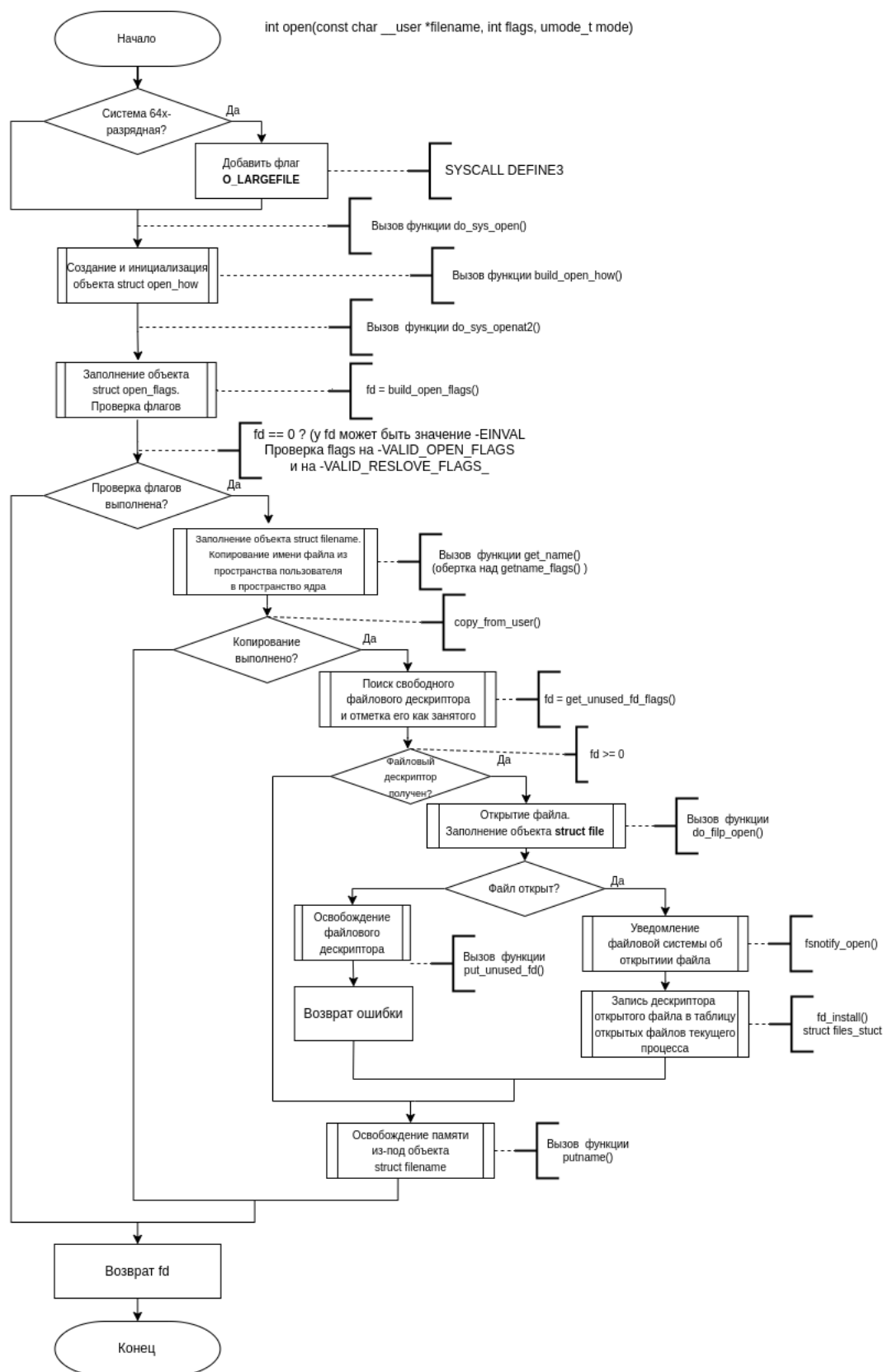


Рисунок 1 – Схема алгоритма работы системного вызова open()

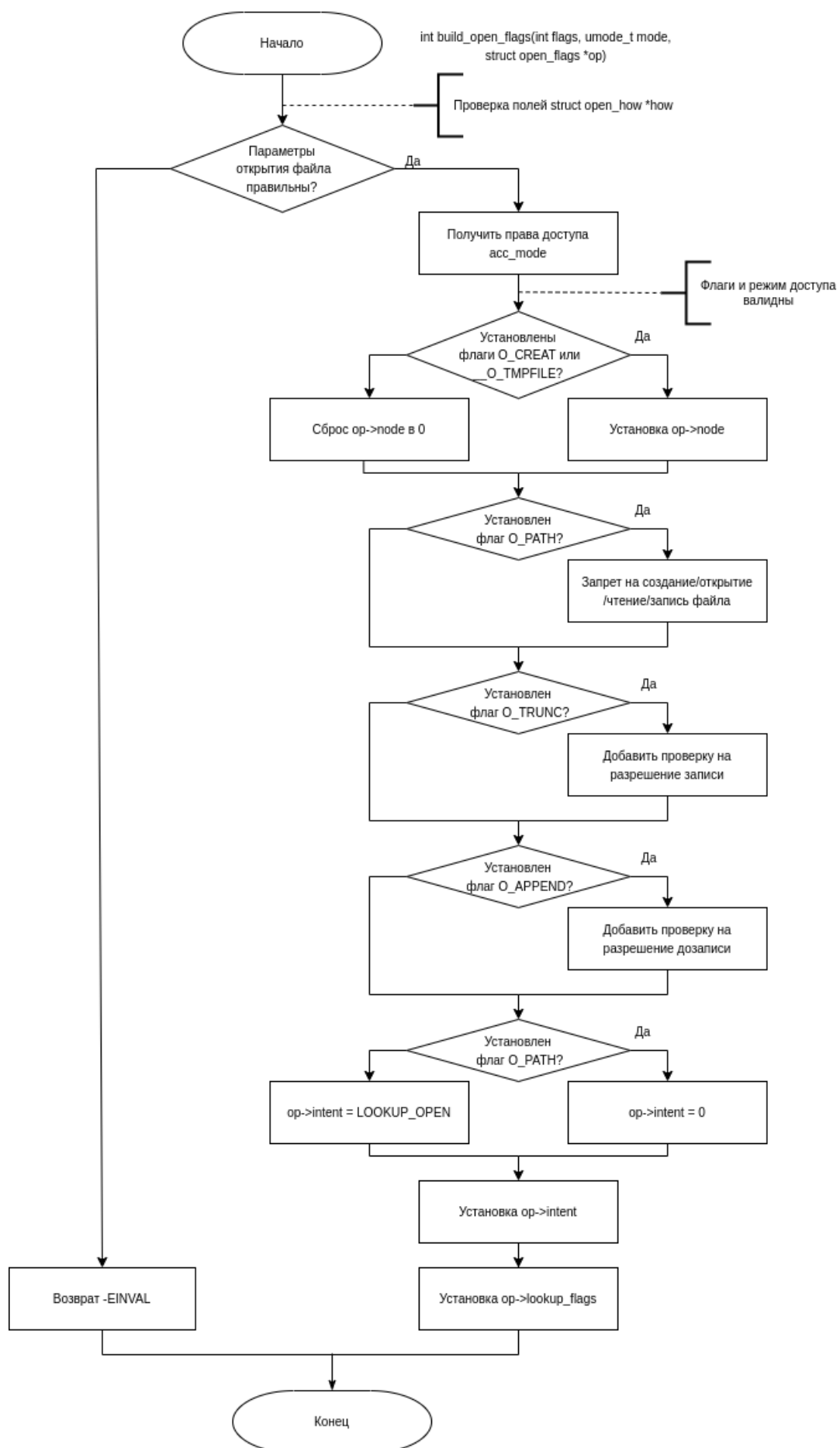


Рисунок 2 – Схема алгоритма работы функции `build_open_flags()`

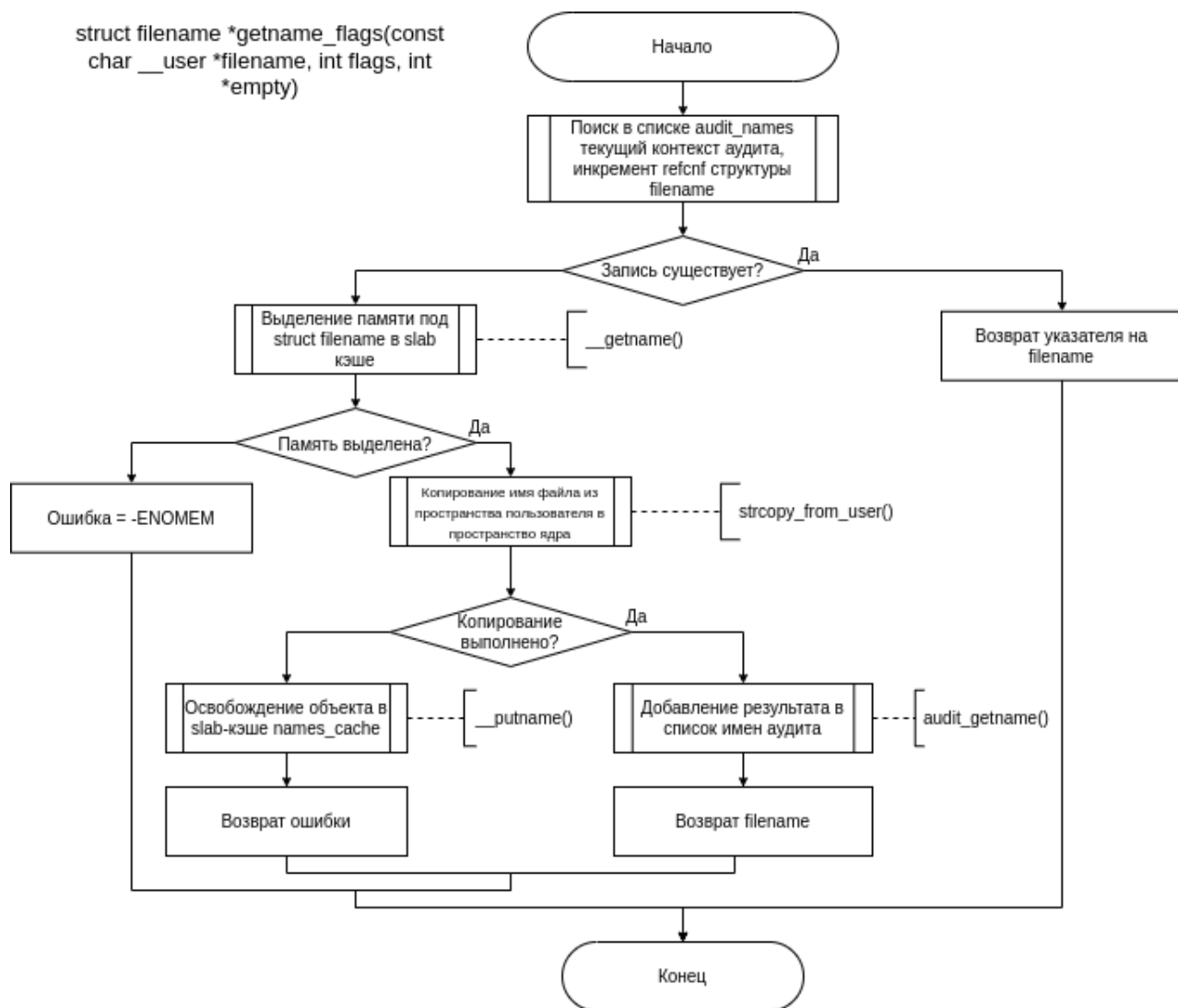


Рисунок 3 – Схема алгоритма работы функции getname_flags()

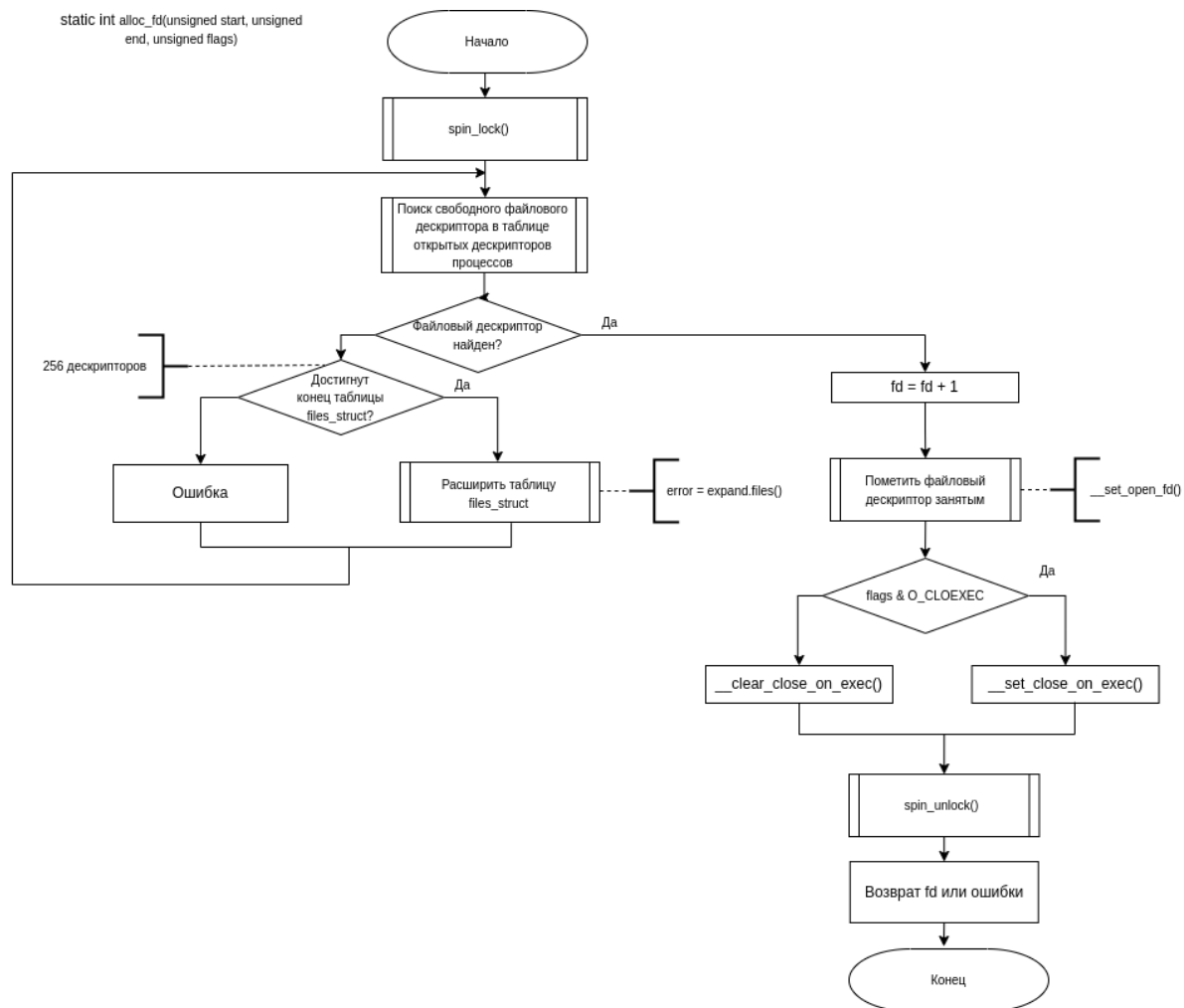


Рисунок 4 – Схема алгоритма работы функции alloc_fd()

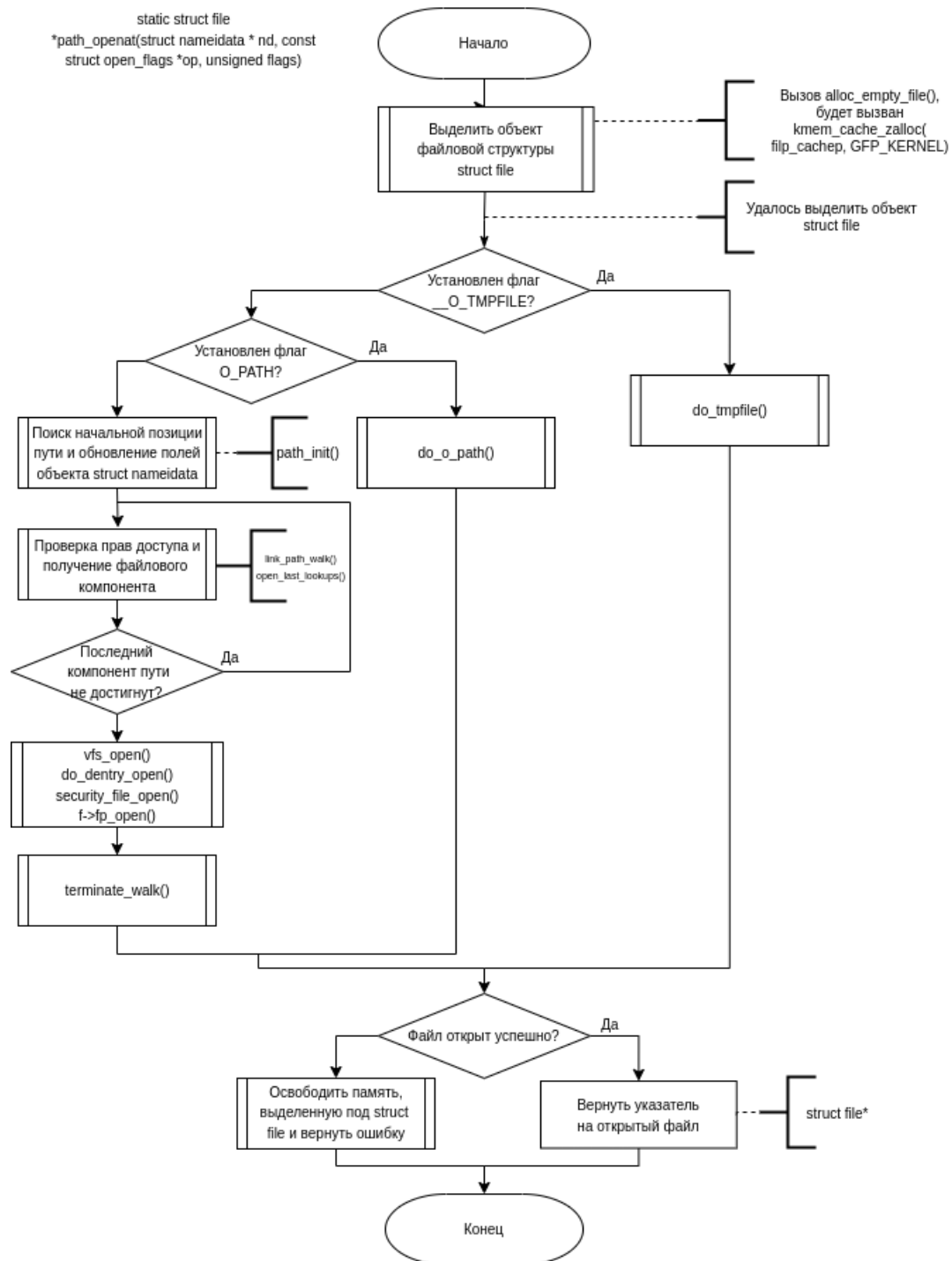


Рисунок 5 – Схема алгоритма функции path_openat()

```
struct file *do_filp_open(int dfd, struct filename
*pathname, const struct open_flags *op)
```

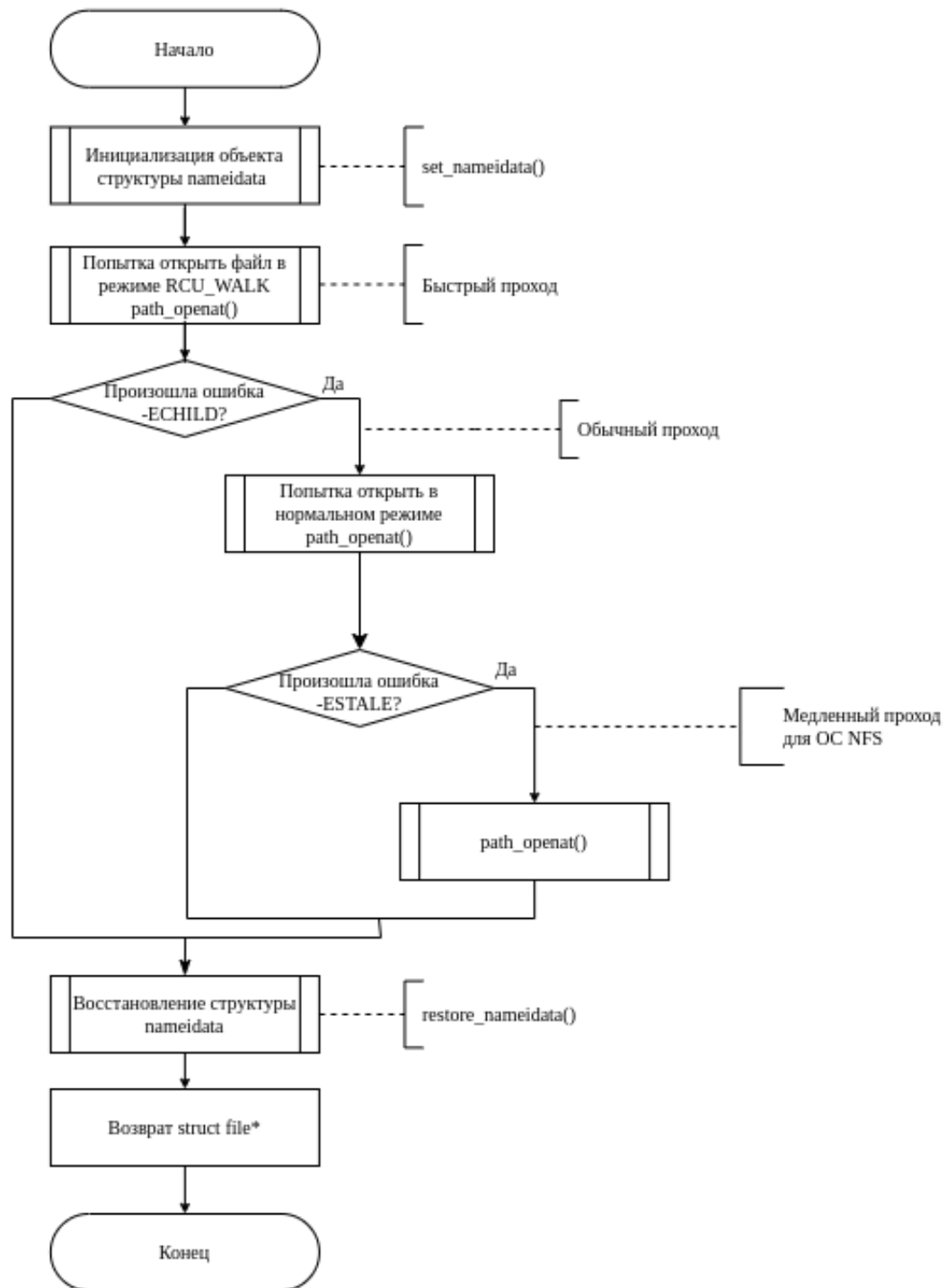


Рисунок 6 – Схема алгоритмов функций, работающих с nameidata (do filp open)

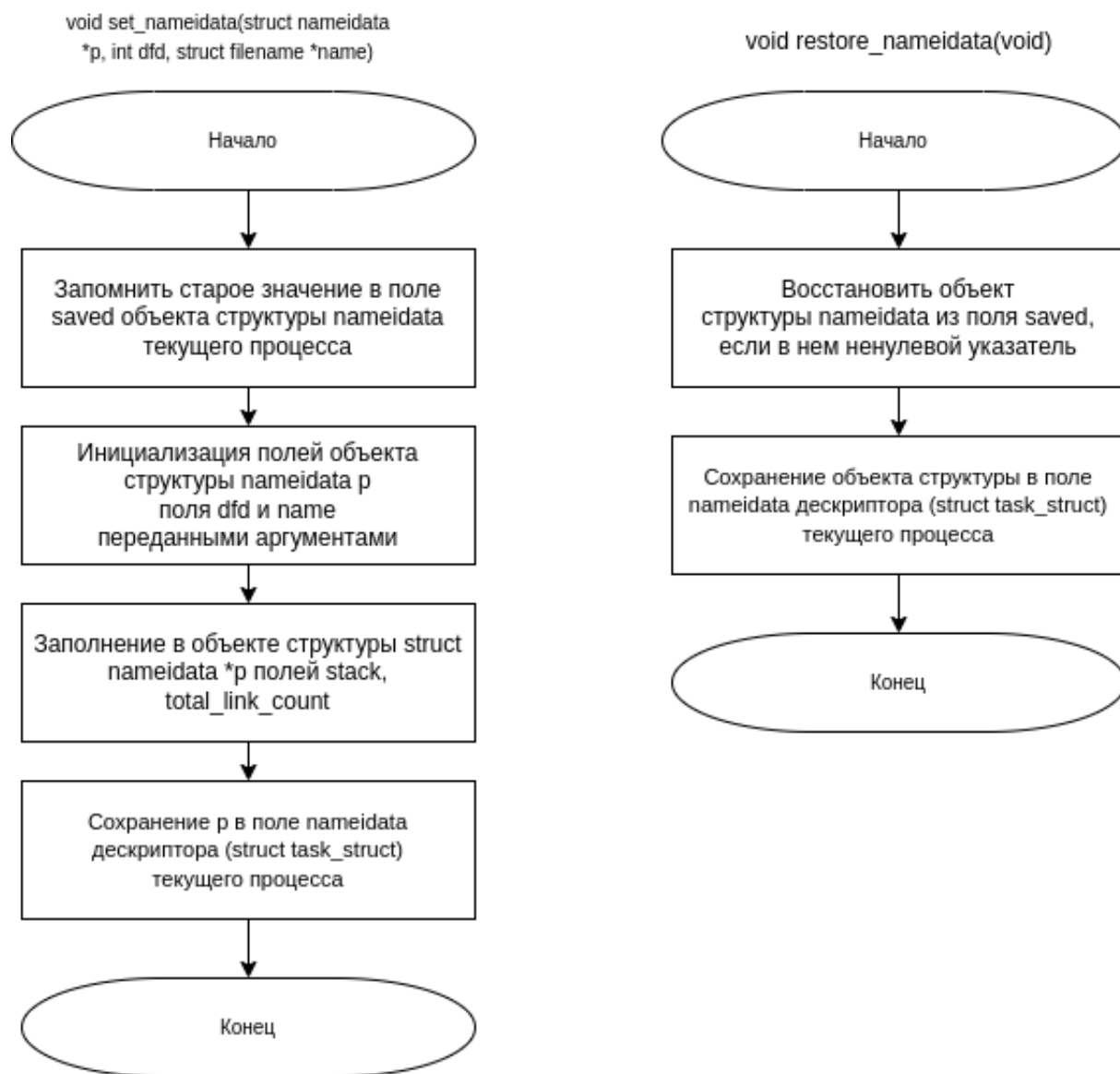


Рисунок 7 – Схема алгоритмов функций, работающих с nameidata

LOOKUP_RCU — флаг для открытия файла в режиме RCU_walk (Допускает возможность одновременного доступа).

LOOKUP_REVAL — флаг для ФС NFS O_APPEND.

O_APPEND может проводить к потере данных файлов в ФС NFS, если одновременно добавл. данные нескольких процессов. Нельзя избежать ускорение гонки.

NFS не поддерживает добавление в файл, потому клиентское ядро имитирует такое поведение.

```
static struct dentry *lookup_open( struct
nameidata *ndm struct file *file, const
struct open_flags *op, bool got_write)
```

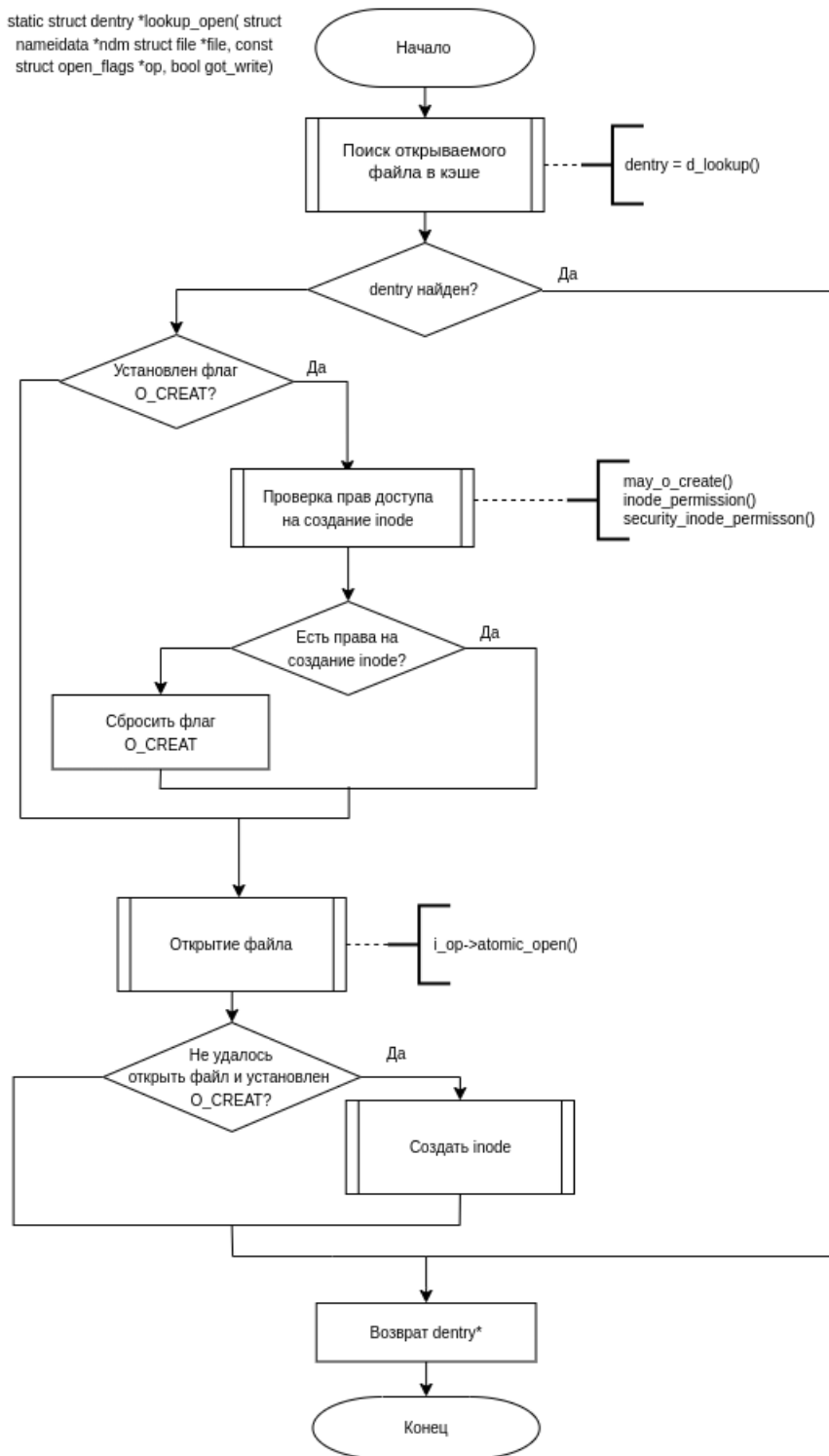


Рисунок 8 – Схема алгоритма функции `open_last_lookups()`

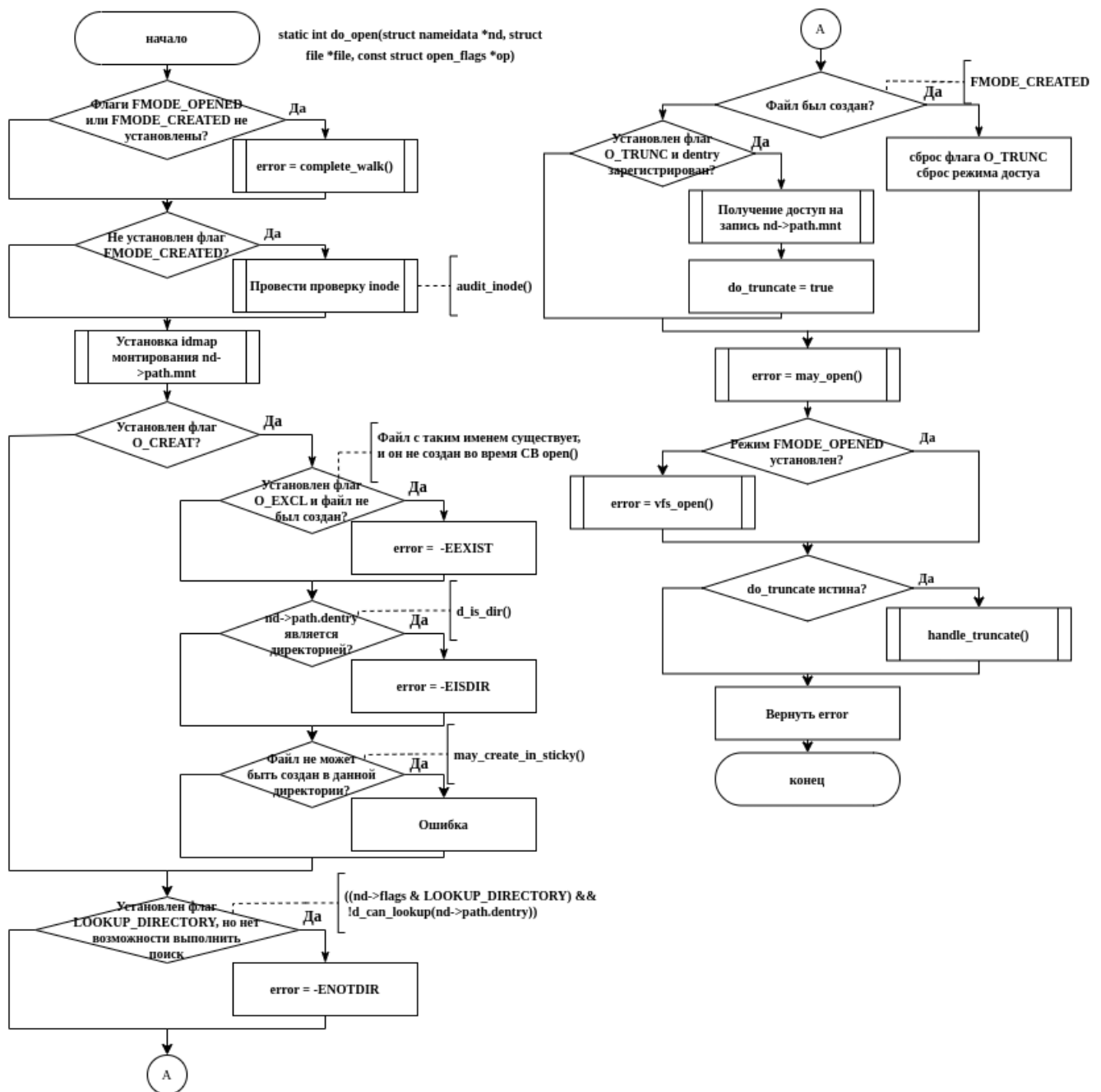


Рисунок 9 – Схема алгоритма функции do_open()

```
static struct dentry *lookup_open( struct
nameidata *ndm, struct file *file, const
struct open_flags *op, bool got_write)
```

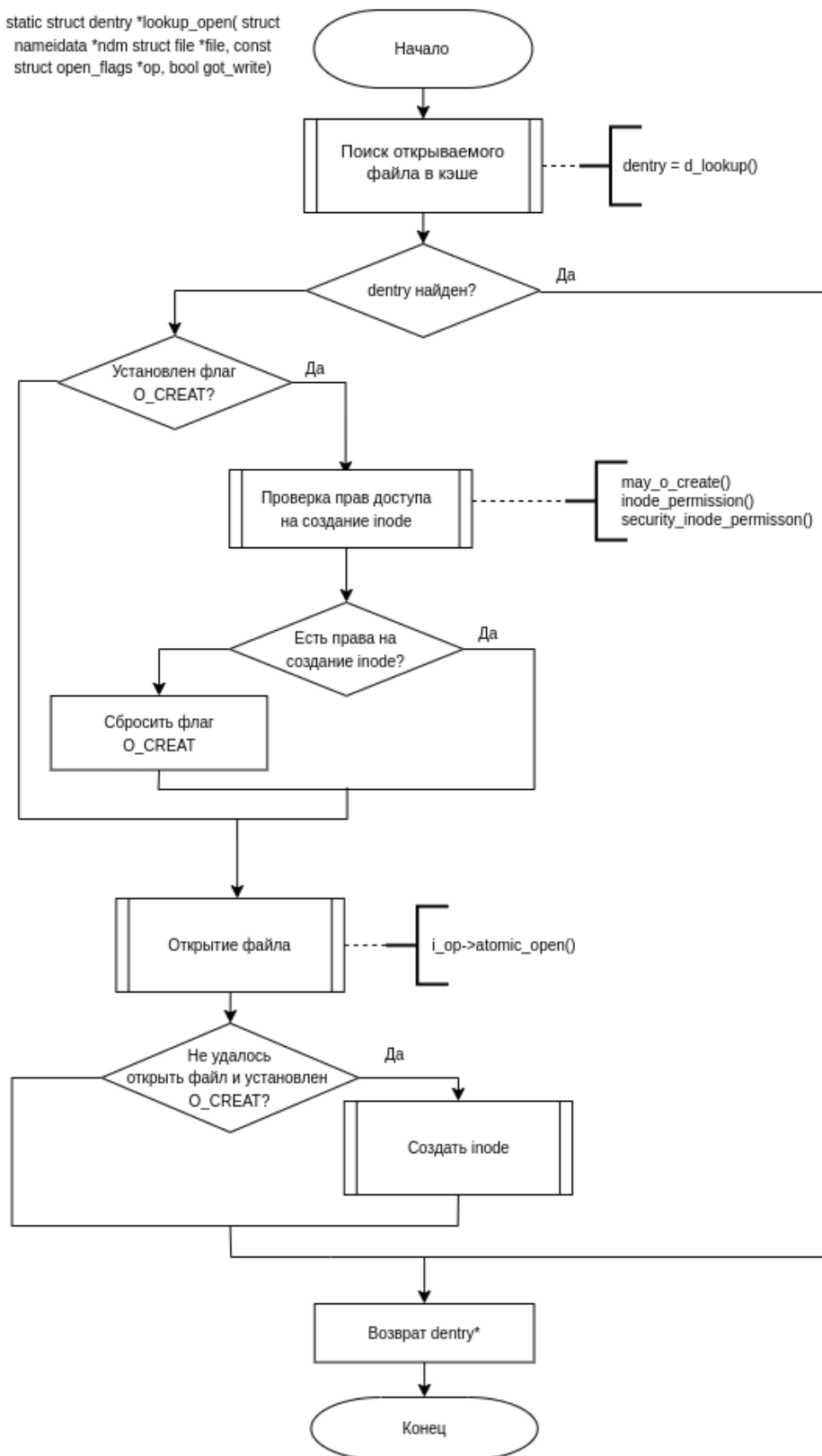


Рисунок 10 – Схема алгоритма функции open_lookup()

```

graph TD
    subgraph "Left Flowchart (Initial Steps)"
        Start([Начало]) --> IsFile{Заключительный компонент пути - файл?}
        IsFile -- Да --> CallHandle[Вызов и возврат handle_dots()]
        CallHandle --> IsOCreate{Флаг O_CREATE установлен?}
        IsOCreate -- Да --> Audit[audit_inode()]
        IsOCreate -- Нет --> LookupFast[lookup_fast()]
        Audit --> SearchOpen[Поиск и открытие файла возможно с его созданием]
        LookupFast --> SearchOpen
        SearchOpen -.-> LookupOpen[lookup_open()]
        SearchOpen --> CheckFlags{Установлен хотя бы один из O_CREAT, O_TRUNC, O_WRONLY, O_RDWR?}
        CheckFlags -- Да --> GetWrite[Получение доступа на запись]
        GetWrite -.-> GotWrite[got_write = !mnt_want_write()]
        CheckFlags -- Нет --> IsOCreate2{Флаг O_CREATE установлен?}
        IsOCreate2 -- Да --> LockInodeWrite[Блокировка inode файла директории на запись с помощью семафора]
        LockInodeWrite -.-> InodeLock[inode_lock()]
        IsOCreate2 -- Нет --> LockInodeShared[Блокировка inode файла директории на чтение с помощью семафора]
        LockInodeShared -.-> InodeLockShared[inode_lock_shared()]
        InodeLock --> SearchOpen2[Поиск и открытие файла возможно с его созданием]
        InodeLockShared --> SearchOpen2
        SearchOpen2 --> A((A))
    end

    subgraph "Right Flowchart (Final Steps)"
        A2((A)) --> IsCreated{Файл создан / открыт?}
        IsCreated -- Да --> Notify[Уведомление файловой системы о создании файла]
        Notify -.-> Fnotify_create[fsnotify_create()]
        IsCreated -- Нет --> IsOCreate3{Флаг O_CREATE установлен?}
        IsOCreate3 -- Да --> UnlockShared[Снятие блокировки с inode файла директории с помощью семафора]
        UnlockShared -.-> InodeUnlock[inode_unlock()]
        IsOCreate3 -- Нет --> InodeUnlocked[Блокировка inode файла директории на чтение с помощью семафора]
        InodeUnlocked -.-> InodeUnlockShared[inode_unlock_shared()]
        InodeUnlocked --> CanWrite{Нет доступа на запись (got_write)?}
        CanWrite -- Да --> DropWrite[mnt_drop_write()]
        CanWrite -- Нет --> NeedSoftlink{Нужно переходить по softlink?}
        NeedSoftlink -- Да --> PutLink[Переход по символической ссылке]
        PutLink -.-> PutLinkLink[put_link()]
        NeedSoftlink -- Нет --> StepInto[Переход к следующему компоненту пути]
        StepInto -.-> StepIntoLink[step_into()]
        StepIntoLink --> ReturnFile[Возврат указателя на имя файла]
        ReturnFile -.-> StructFilename[struct filename]
        ReturnFile --> End([Конец])
    end

```

Рисунок 11 – Схема алгоритма функции last_lookup()