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»

на тему: «Системный вызов open»

Студент ИУ7-63Б _____ Лагутин Д. В.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Рязанова Н. Ю.

(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

1 Используемые структуры

Листинг 1 – struct open_how

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

Листинг 2 – struct open_flags

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

Листинг 3 – struct filename

```
struct filename {
        const char              *name;  /* pointer to actual string */
        const __user char       *uptr;  /* original userland pointer */
        int                     refcnt;
        struct audit_names      *aname;
        const char              iname[];
};
```

Листинг 4 – struct audit_names

```
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;
        kgid_t                  gid;
```

```
        dev_t                   rdev;
        u32                     osid;
        struct audit_cap_data   fcap;
        unsigned int            fcap_ver;
        unsigned char           type;           /* record type */
        /*
         * This was an allocated audit_names and not from the array of
         * names allocated in the task audit context.  Thus this name
         * should be freed on syscall exit.
         */
        bool                    should_free;
};
```

Листинг 5 – struct files_struct

```
struct files_struct {
  /*
   * read mostly part
   */
        atomic_t count;
        bool resize_in_progress;
        wait_queue_head_t resize_wait;

        struct fdtable __rcu *fdt;
        struct fdtable fdtab;
  /*
   * written part on a separate cache line in SMP
   */
        spinlock_t file_lock ____cacheline_aligned_in_smp;
        unsigned int next_fd;
        unsigned long close_on_exec_init[1];
        unsigned long open_fds_init[1];
        unsigned long full_fds_bits_init[1];
        struct file __rcu * fd_array[NR_OPEN_DEFAULT];
};
```

Листинг 6 – struct fdtable

```
struct fdtable {
        unsigned int max_fds;
        struct file __rcu **fd;      /* current fd array */
        unsigned long *close_on_exec;
        unsigned long *open_fds;
        unsigned long *full_fds_bits;
        struct rcu_head rcu;
};
```

Листинг 7 – struct file

```
struct file {
        union {
                struct llist_node       f_llist;
                struct rcu_head         f_rcuhead;
                unsigned int            f_iocb_flags;
        };
        struct path             f_path;
        struct inode            *f_inode;       /* cached value */
        const struct file_operations    *f_op;

        /*
         * Protects f_ep, f_flags.
         * Must not be taken from IRQ context.
         */
        spinlock_t              f_lock;
        atomic_long_t           f_count;
        unsigned int            f_flags;
        fmode_t                 f_mode;
        struct mutex            f_pos_lock;
        loff_t                  f_pos;
        struct fown_struct      f_owner;
        const struct cred       *f_cred;
        struct file_ra_state    f_ra;

        u64                     f_version;
#ifdef CONFIG_SECURITY
        void                    *f_security;
#endif
        /* needed for tty driver, and maybe others */
        void                    *private_data;

#ifdef CONFIG_EPOLL
        /* Used by fs/eventpoll.c to link all the hooks to this file */
        struct hlist_head       *f_ep;
#endif /* #ifdef CONFIG_EPOLL */
        struct address_space    *f_mapping;
        errseq_t                f_wb_err;
        errseq_t                f_sb_err; /* for syncfs */
};
```

Листинг 8 – struct nameidata

```
struct nameidata {
        struct path     path;
        struct qstr     last;
        struct path     root;
        struct inode    *inode; /* path.dentry.d_inode */
```

```
6         unsigned int    flags, state;
7         unsigned        seq, next_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        vfsuid_t        dir_vfsuid;
22        umode_t         dir_mode;
23 };
```

Листинг 9 – struct path

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

Листинг 10 – struct vfsmount

```
1 struct vfsmount {
2         struct dentry *mnt_root;        /* root of the mounted tree */
3         struct super_block *mnt_sb;     /* pointer to superblock */
4         int mnt_flags;
5         struct mnt_idmap *mnt_idmap;
6 };
```

Листинг 11 – struct dentry

```
1 struct dentry {
2         /* RCU lookup touched fields */
3         unsigned int d_flags;           /* protected by d_lock */
4         seqcount_spinlock_t d_seq;      /* per dentry seqlock */
5         struct hlist_bl_node d_hash;    /* lookup hash list */
6         struct dentry *d_parent;        /* parent directory */
7         struct qstr d_name;
8         struct inode *d_inode;          /* Where the name belongs to -
      NULL is
9                                          * negative */
10        unsigned char d_iname[DNAME_INLINE_LEN];        /* small names */
11
```

```
12          /* Ref lookup also touches following */
13          struct lockref d_lockref;       /* per-dentry lock and refcount */
14          const struct dentry_operations *d_op;
15          struct super_block *d_sb;       /* The root of the dentry tree */
16          unsigned long d_time;           /* used by d_revalidate */
17          void *d_fsdata;                 /* fs-specific data */
18
19          union {
20                  struct list_head d_lru;         /* LRU list */
21                  wait_queue_head_t *d_wait;      /* in-lookup ones only */
22          };
23          struct list_head d_child;       /* child of parent list */
24          struct list_head d_subdirs;     /* our children */
25          /*
26           * d_alias and d_rcu can share memory
27           */
28          union {
29                  struct hlist_node d_alias;      /* inode alias list */
30                  struct hlist_bl_node d_in_lookup_hash;  /* only for
    in-lookup ones */
31                  struct rcu_head d_rcu;
32          } d_u;
33  }
```

Листинг 12 – struct inode_operations

```
1  struct inode_operations {
2          struct dentry * (*lookup) (struct inode *,struct dentry *,
    unsigned int);
3          const char * (*get_link) (struct dentry *, struct inode *, struct
    delayed_call *);
4          int (*permission) (struct mnt_idmap *, struct inode *, int);
5          struct posix_acl * (*get_inode_acl)(struct inode *, int, bool);
6
7          int (*readlink) (struct dentry *, char __user *,int);
8
9          int (*create) (struct mnt_idmap *, struct inode *,struct dentry *,
10                        umode_t, bool);
11         int (*link) (struct dentry *,struct inode *,struct dentry *);
12         int (*unlink) (struct inode *,struct dentry *);
13         int (*symlink) (struct mnt_idmap *, struct inode *,struct dentry
    *,
14                         const char *);
15         int (*mkdir) (struct mnt_idmap *, struct inode *,struct dentry *,
16                       umode_t);
17         int (*rmdir) (struct inode *,struct dentry *);
18         int (*mknod) (struct mnt_idmap *, struct inode *,struct dentry *,
19                       umode_t,dev_t);
20         int (*rename) (struct mnt_idmap *, struct inode *, struct dentry
```

```
    *,
21                          struct inode *, struct dentry *, unsigned int);
22          int (*setattr) (struct mnt_idmap *, struct dentry *, struct iattr
    *);
23          int (*getattr) (struct mnt_idmap *, const struct path *,
24                          struct kstat *, u32, unsigned int);
25          ssize_t (*listxattr) (struct dentry *, char *, size_t);
26          int (*fiemap)(struct inode *, struct fiemap_extent_info *, u64
    start,
27                        u64 len);
28          int (*update_time)(struct inode *, struct timespec64 *, int);
29          int (*atomic_open)(struct inode *, struct dentry *,
30                             struct file *, unsigned open_flag,
31                             umode_t create_mode);
32          int (*tmpfile) (struct mnt_idmap *, struct inode *,
33                          struct file *, umode_t);
34          struct posix_acl *(*get_acl)(struct mnt_idmap *, struct dentry *,
35                                       int);
36          int (*set_acl)(struct mnt_idmap *, struct dentry *,
37                         struct posix_acl *, int);
38          int (*fileattr_set)(struct mnt_idmap *idmap,
39                              struct dentry *dentry, struct fileattr *fa);
40          int (*fileattr_get)(struct dentry *dentry, struct fileattr *fa);
41  }
```

Листинг 13 – struct dentry_operations

```
1  struct dentry_operations {
2          int (*d_revalidate)(struct dentry *, unsigned int);
3          int (*d_weak_revalidate)(struct dentry *, unsigned int);
4          int (*d_hash)(const struct dentry *, struct qstr *);
5          int (*d_compare)(const struct dentry *,
6                           unsigned int, const char *, const struct qstr *);
7          int (*d_delete)(const struct dentry *);
8          int (*d_init)(struct dentry *);
9          void (*d_release)(struct dentry *);
10         void (*d_prune)(struct dentry *);
11         void (*d_iput)(struct dentry *, struct inode *);
12         char *(*d_dname)(struct dentry *, char *, int);
13         struct vfsmount *(*d_automount)(struct path *);
14         int (*d_manage)(const struct path *, bool);
15         struct dentry *(*d_real)(struct dentry *, const struct inode *);
16 }
```

2 Схема алгоритма

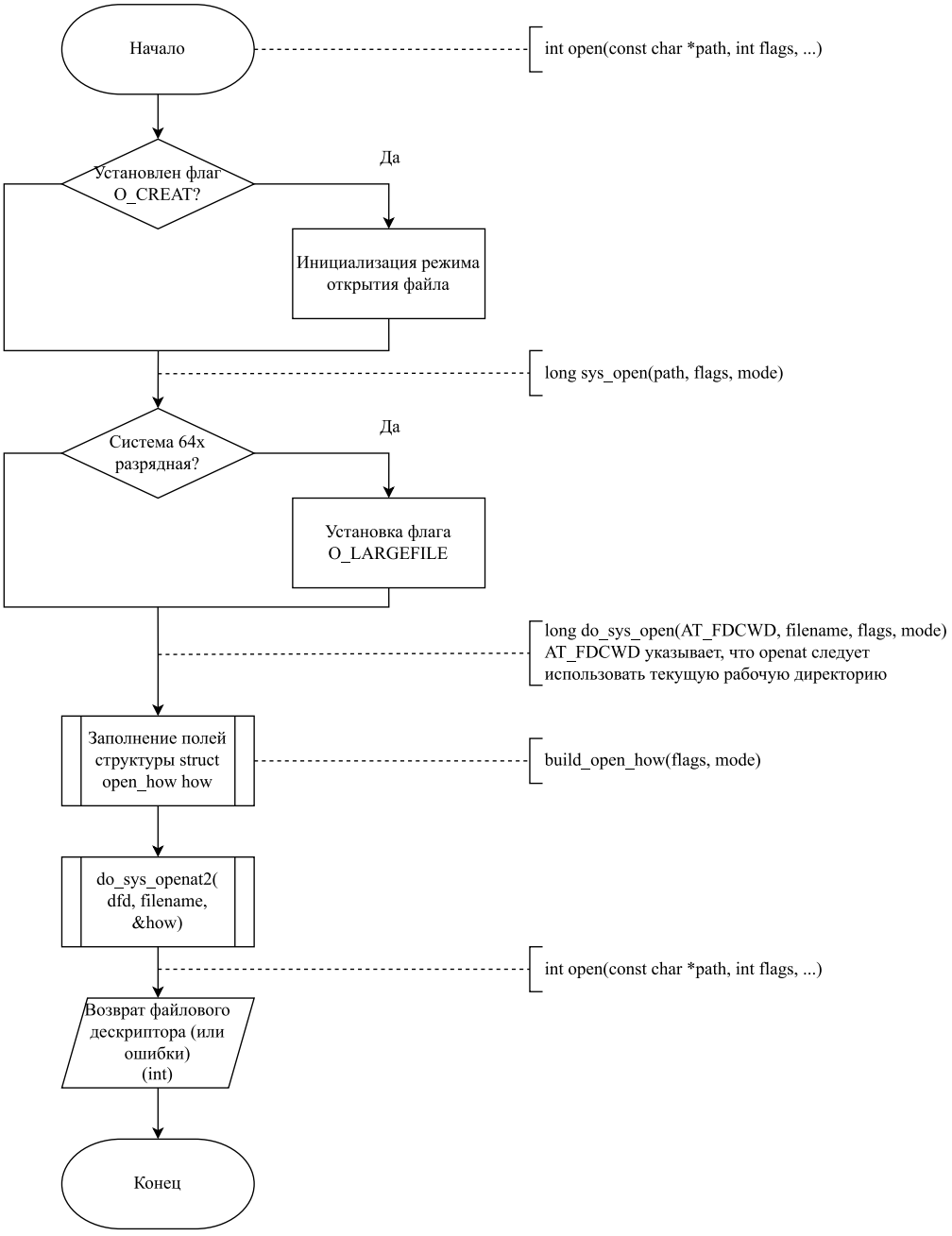

Рисунок 2.1 – Схема алгоритма функции open

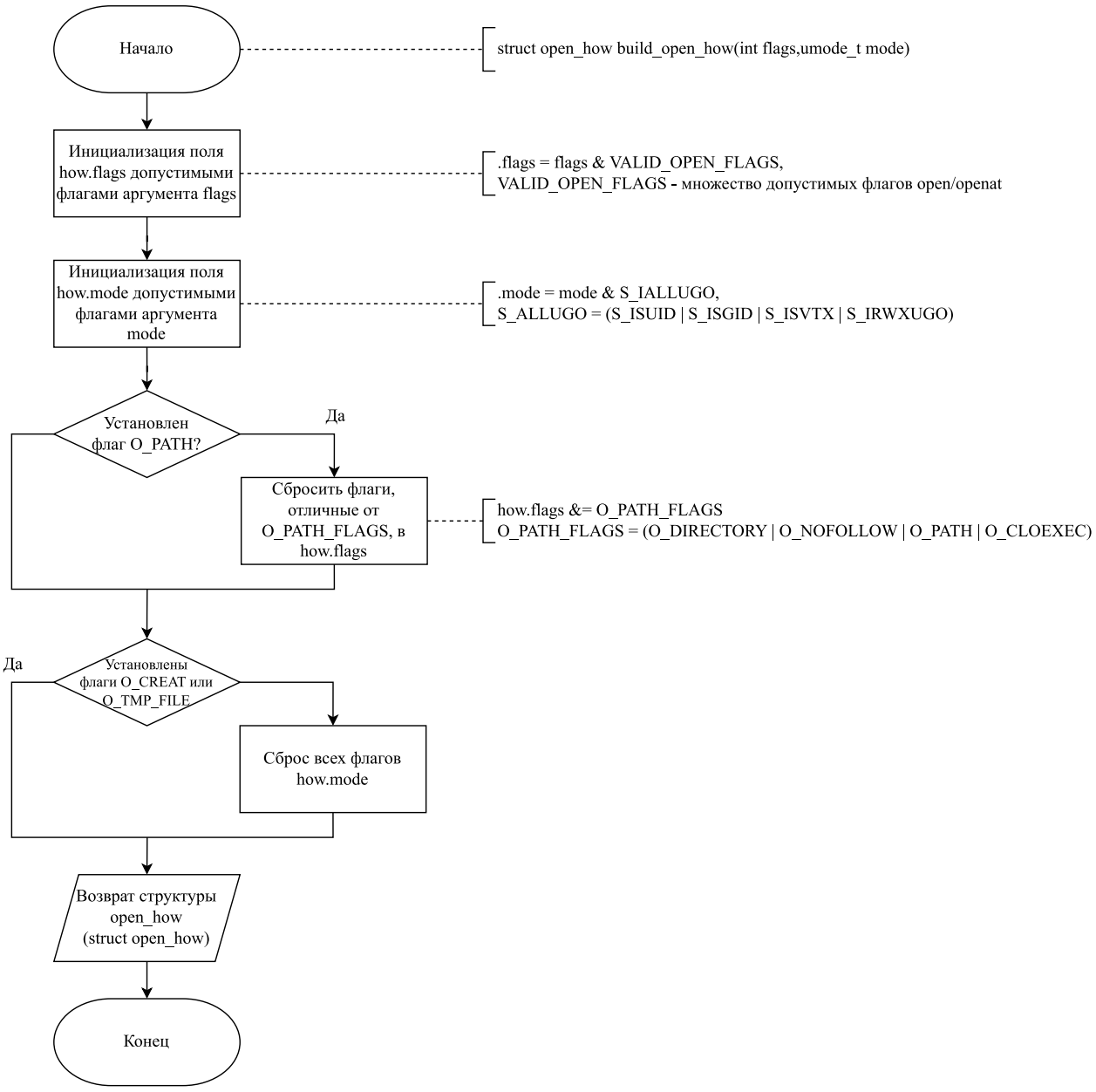

Рисунок 2.2 – Схема алгоритма функции build_open_how

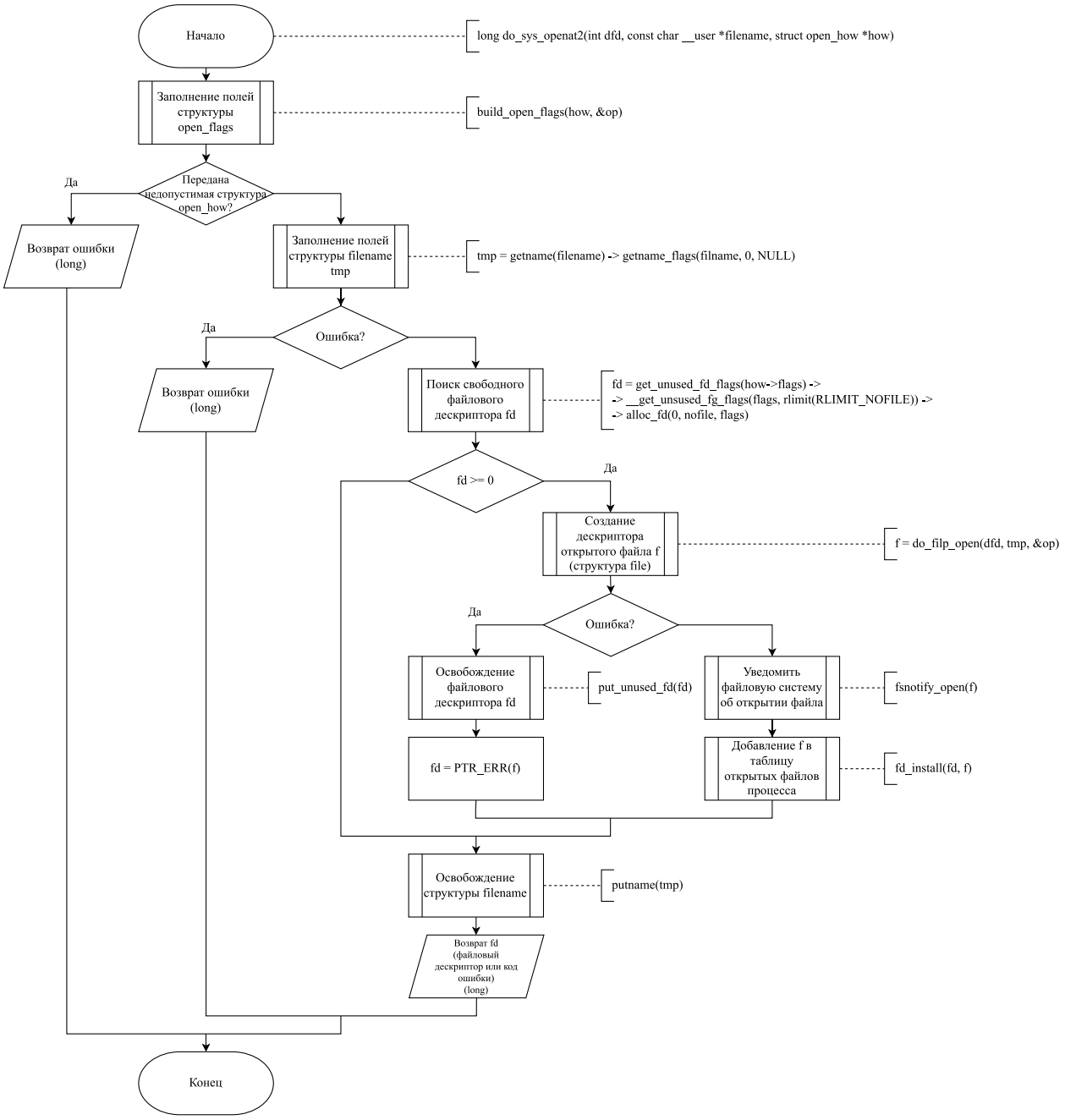

Рисунок 2.3 – Схема алгоритма функции do_sys_openat2

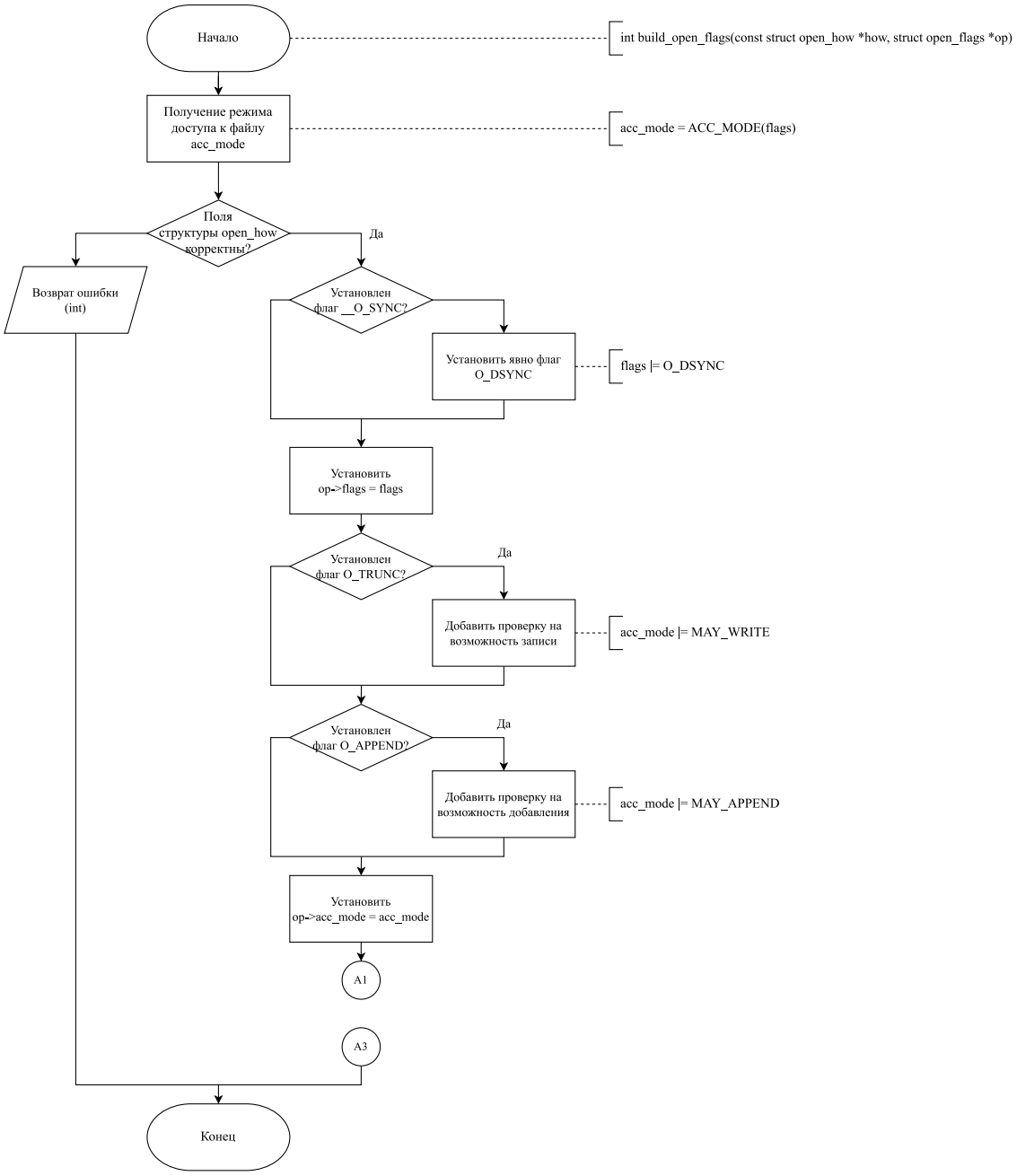

Рисунок 2.4 – Схема алгоритма функции build_open_flags (часть 1)

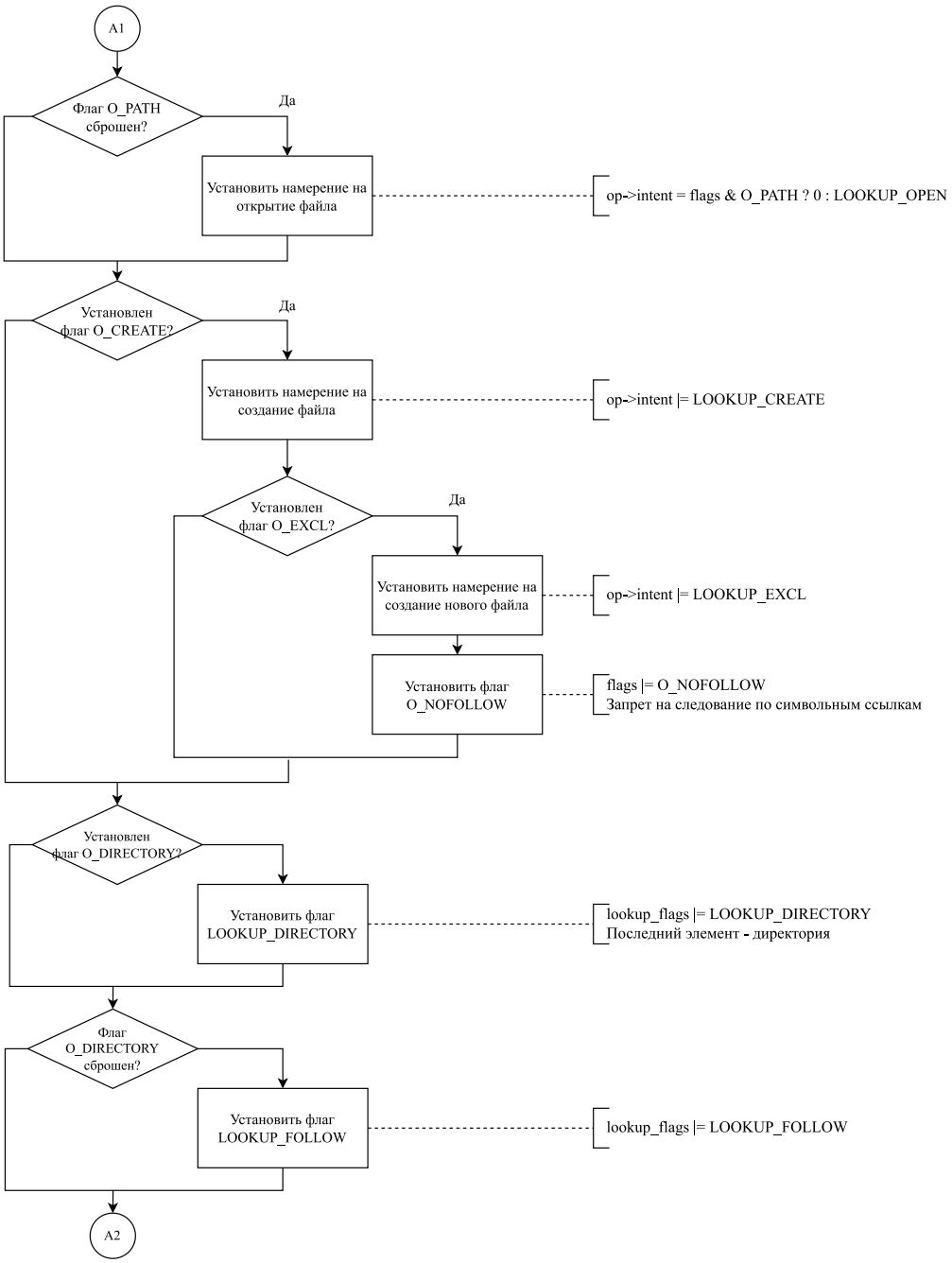

Рисунок 2.5 – Схема алгоритма функции build_open_flags (часть 2)

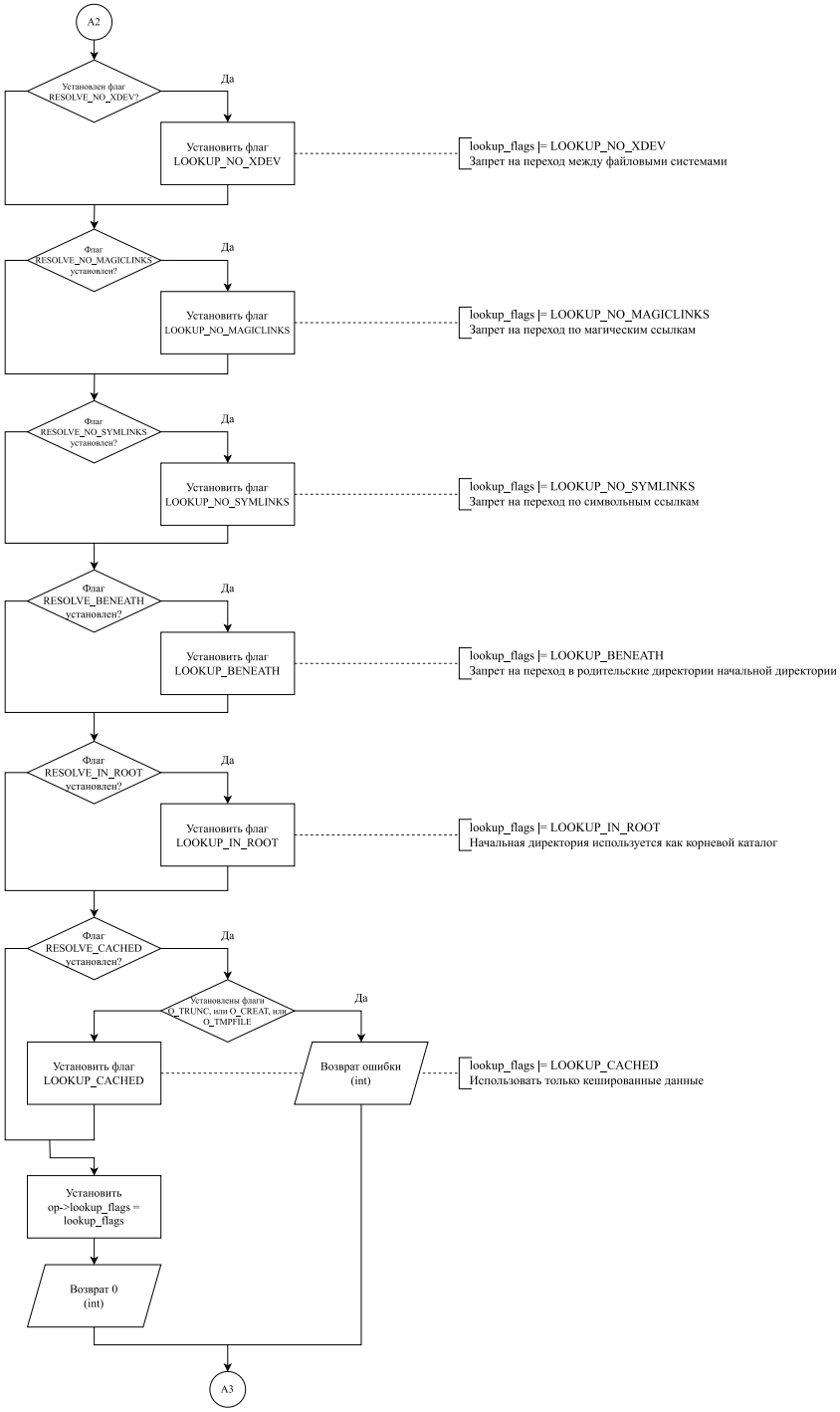

Рисунок 2.6 – Схема алгоритма функции build_open_flags (часть 3)

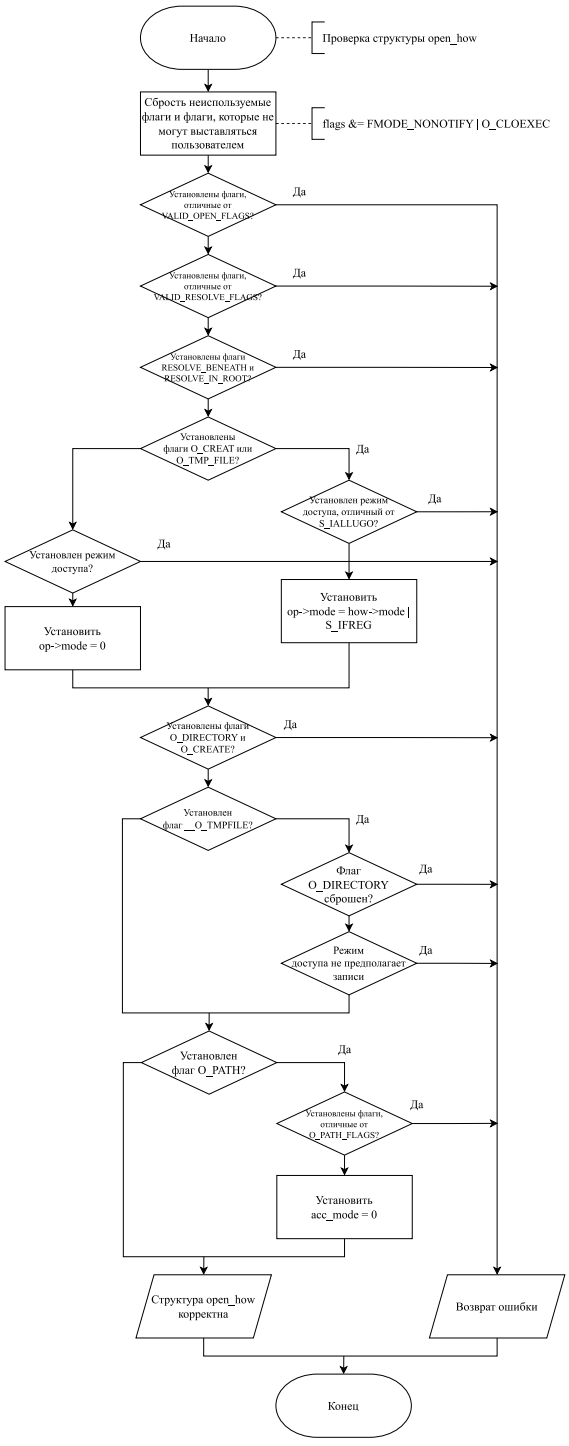

Рисунок 2.7 – Схема алгоритма проверки полей структуры open_how

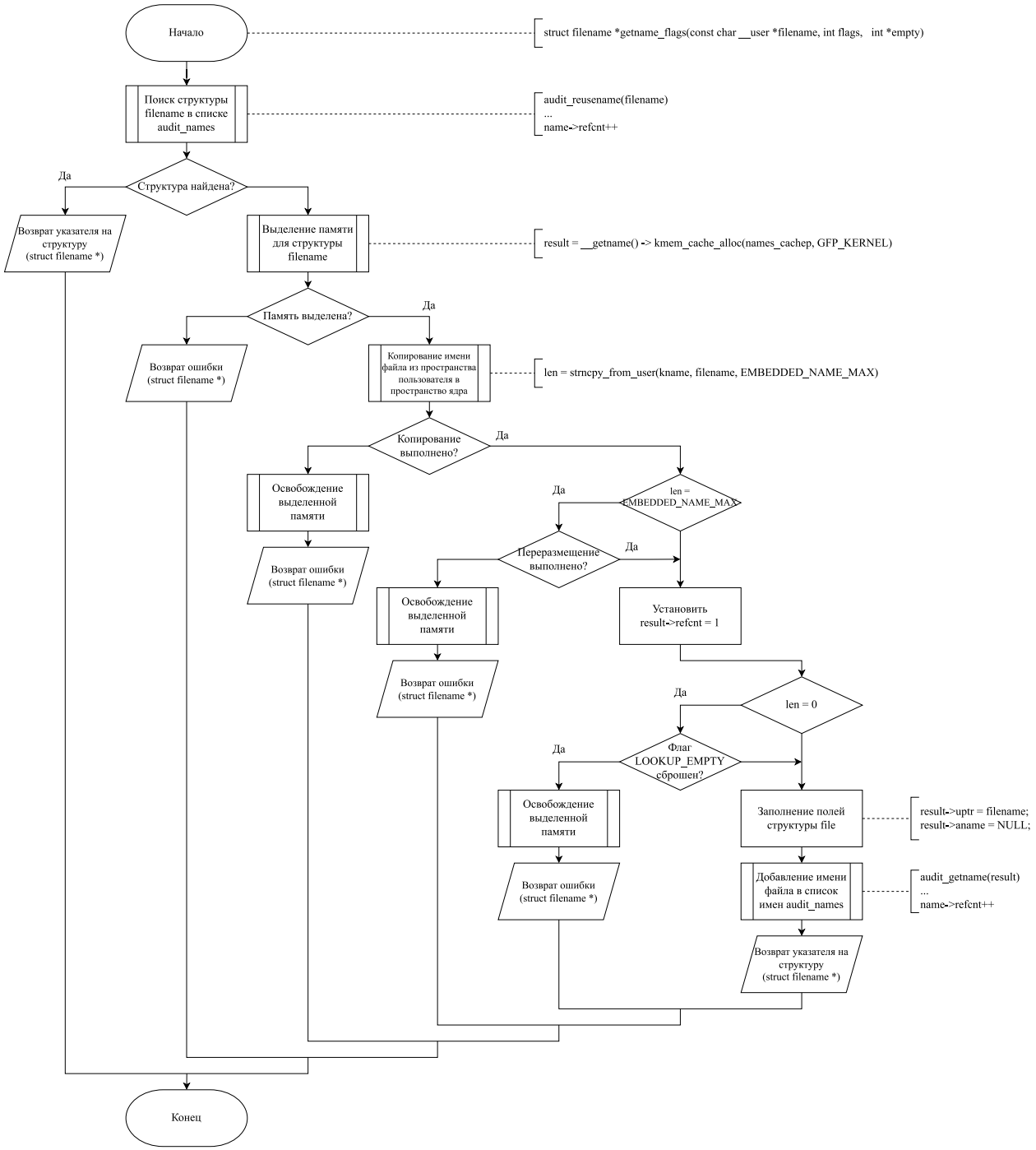

Рисунок 2.8 – Схема алгоритма функции getname_flags

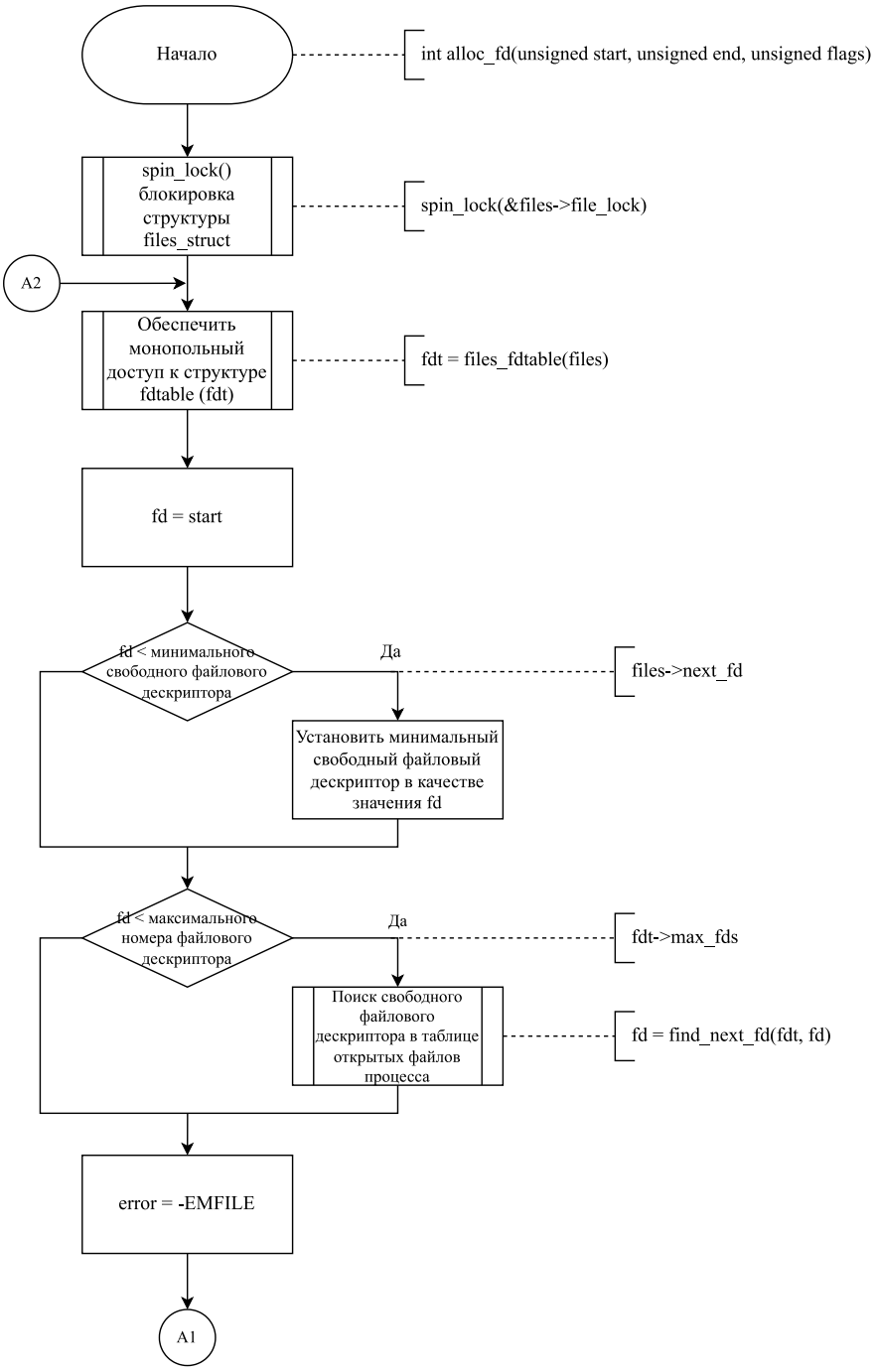

Рисунок 2.9 – Схема алгоритма функции alloc_fd (часть 1)

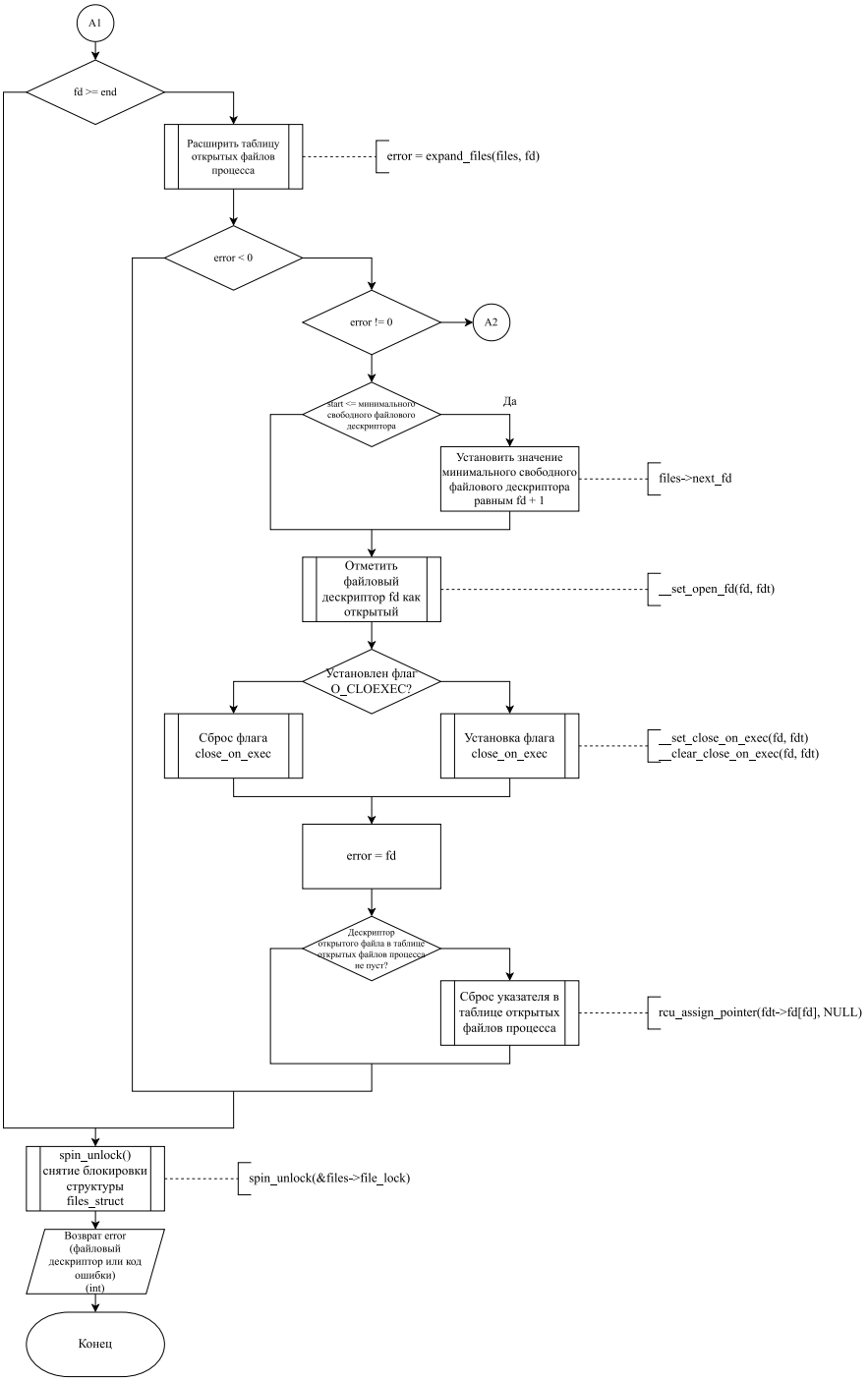

Рисунок 2.10 – Схема алгоритма функции alloc_fd (часть 2)

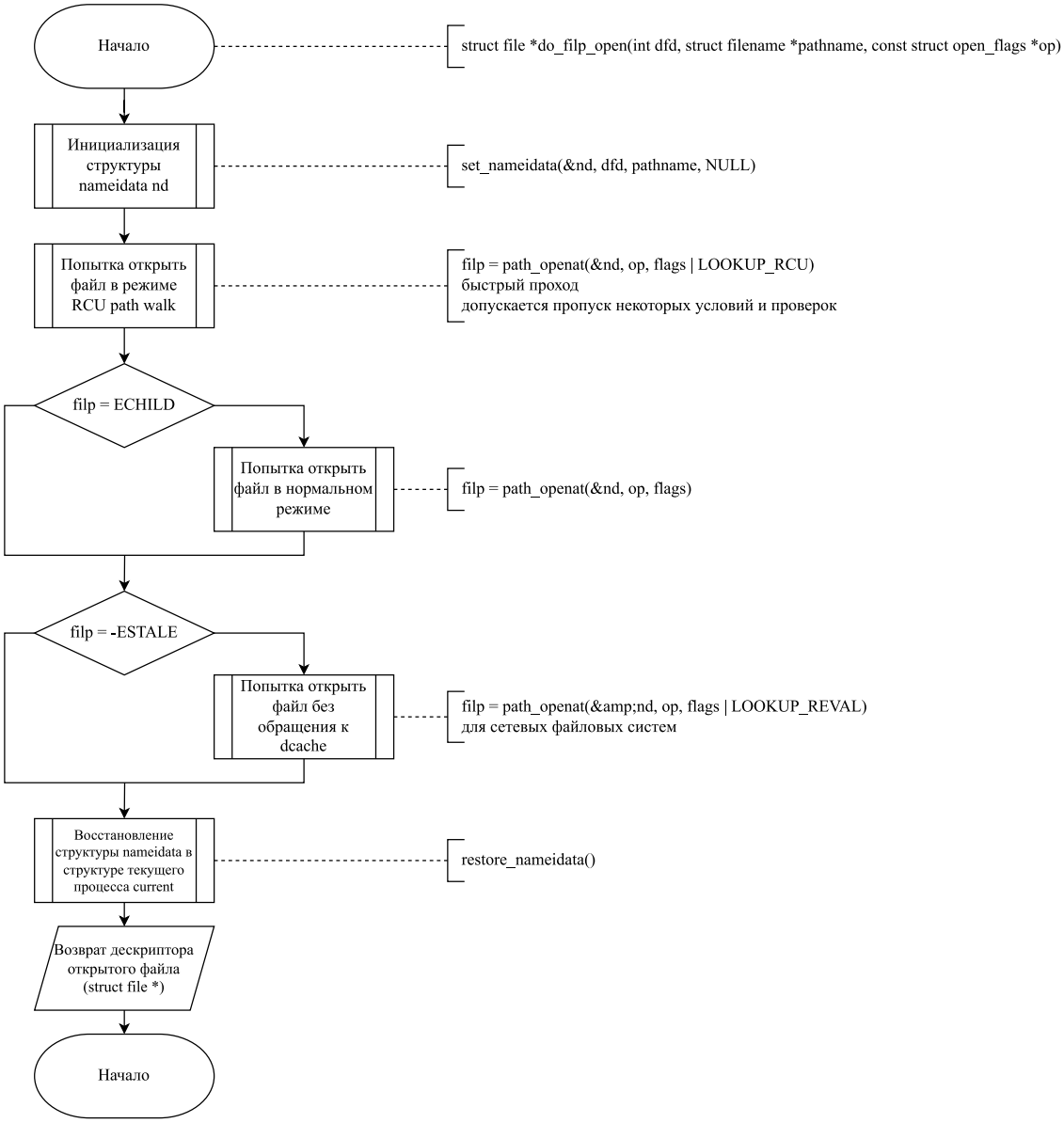

Рисунок 2.11 – Схема алгоритма функции do_filp_open

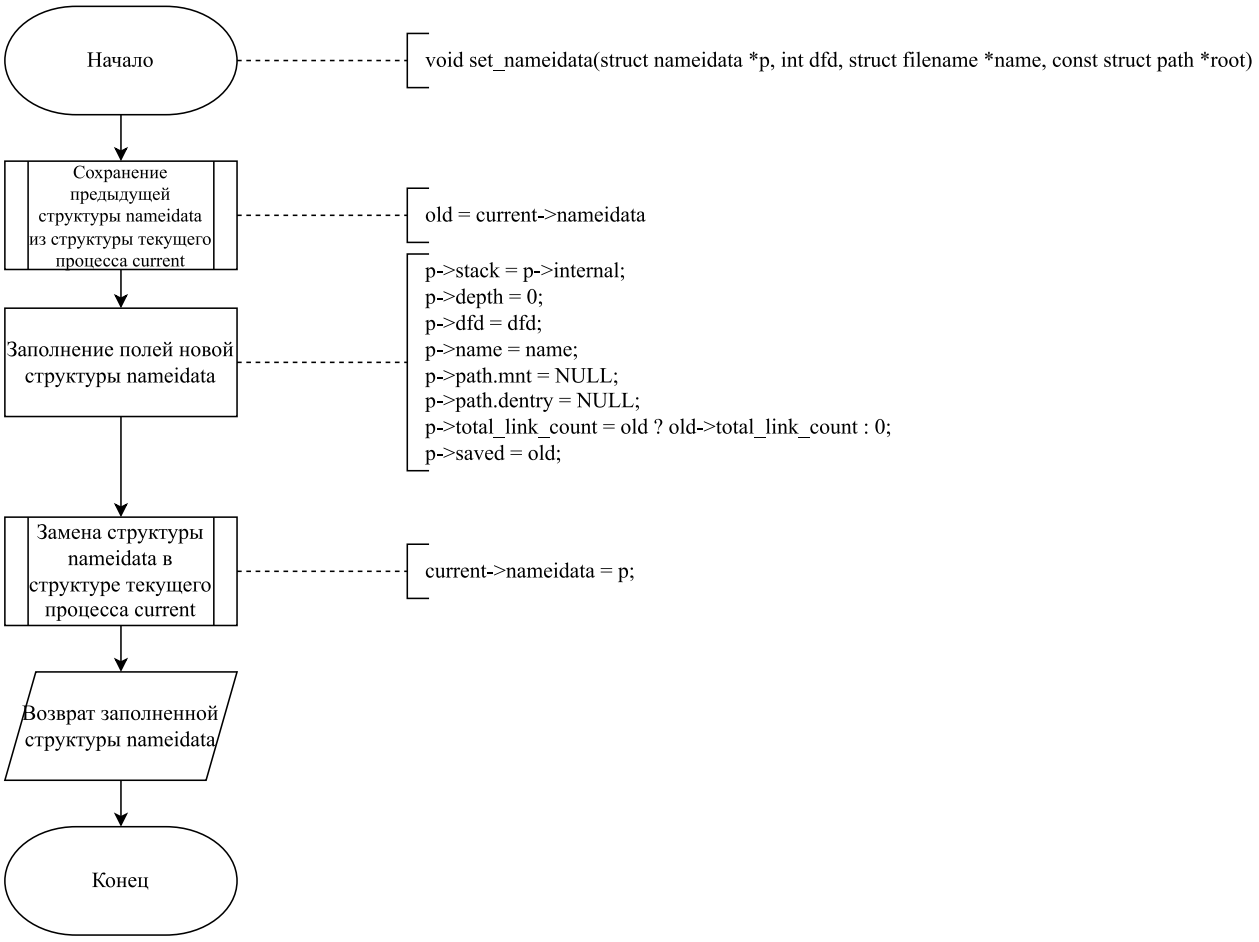

Рисунок 2.12 – Схема алгоритма функции set_nameidata

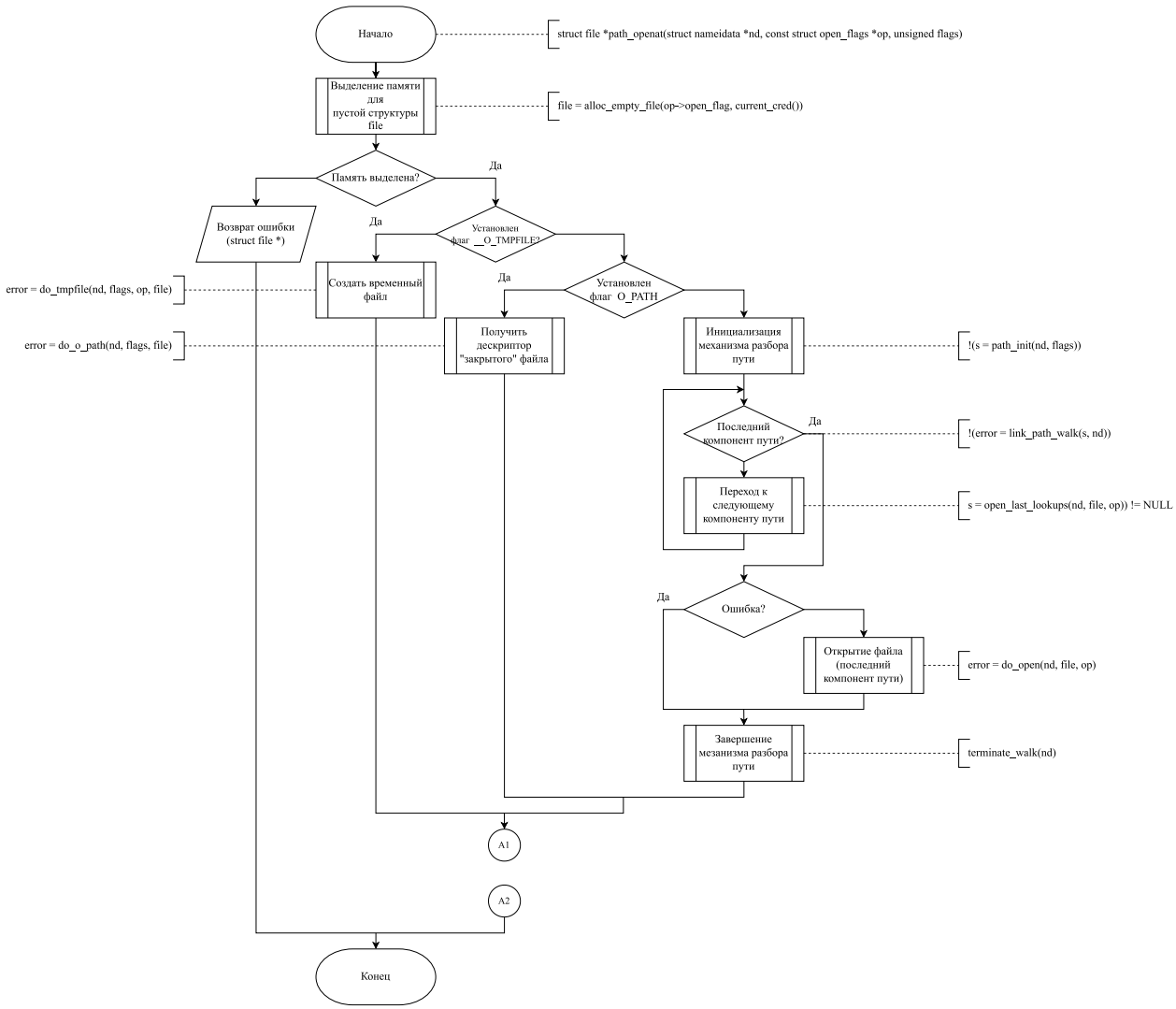

Рисунок 2.13 – Схема алгоритма функции path_openat (часть 1)

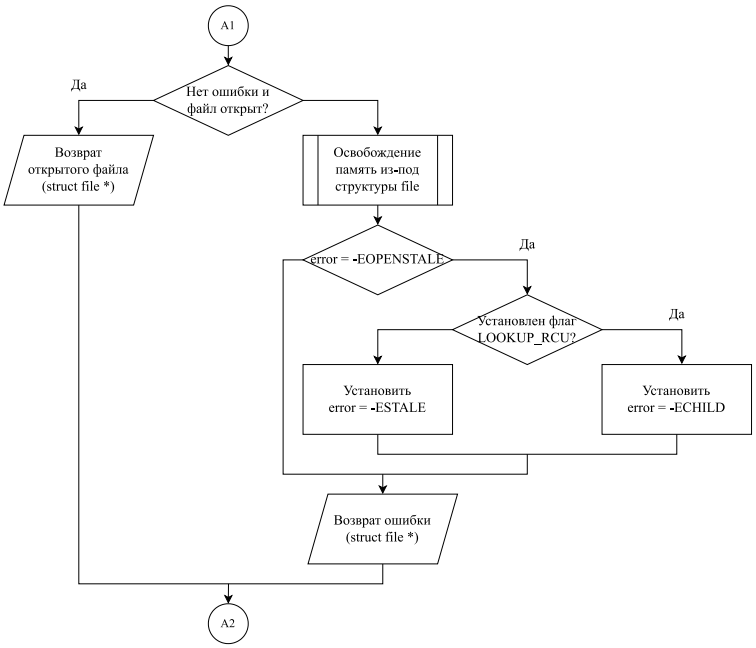

Рисунок 2.14 – Схема алгоритма функции path_openat (часть 2)

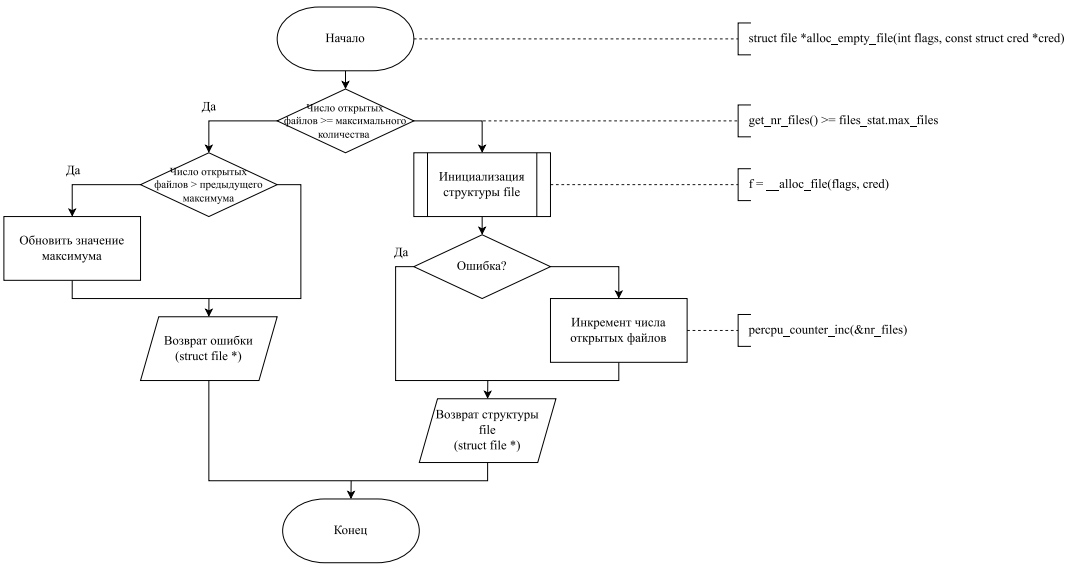

Рисунок 2.15 – Схема алгоритма функции alloc_empty_file

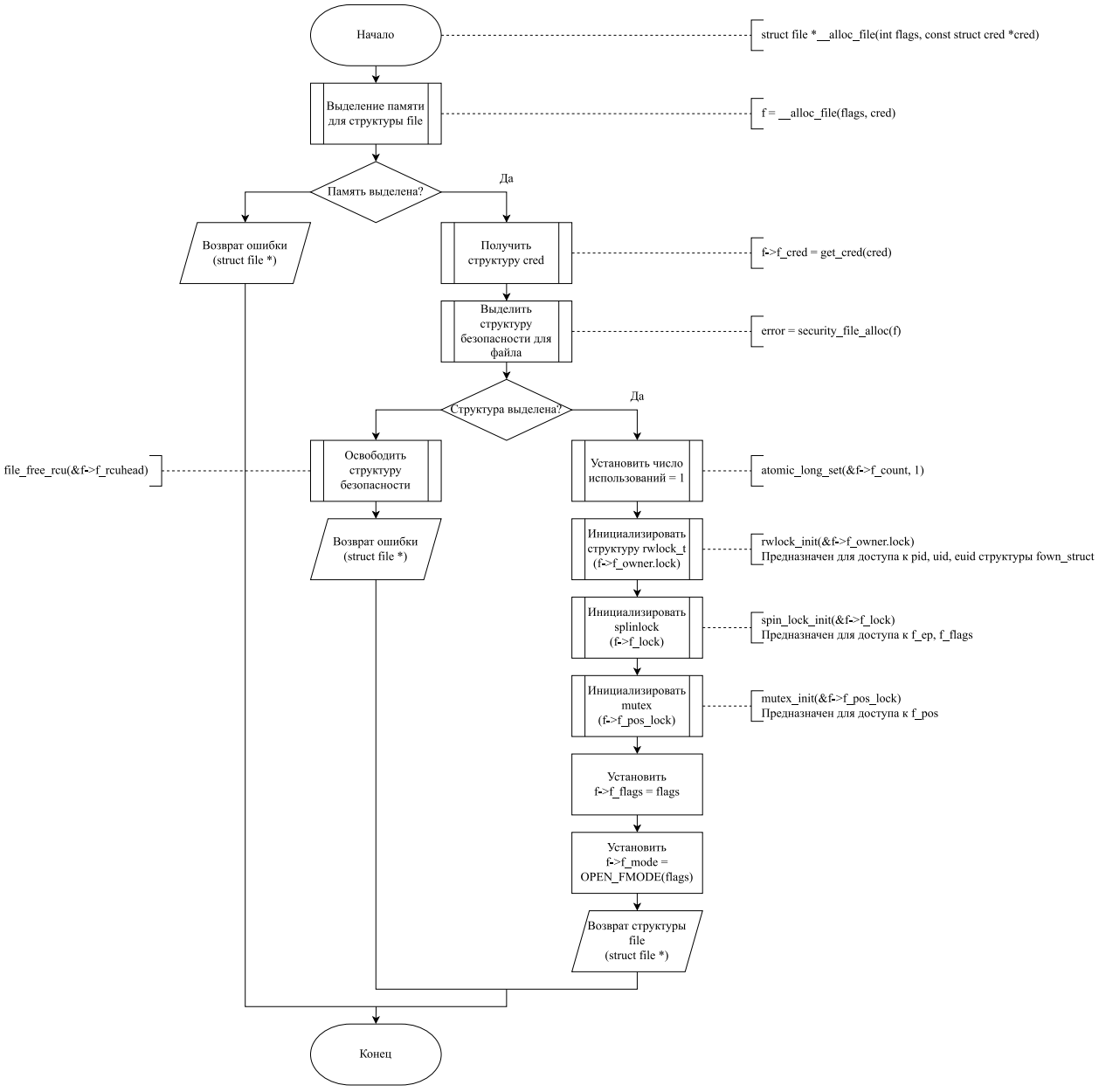

Рисунок 2.16 – Схема алгоритма функции __alloc_file

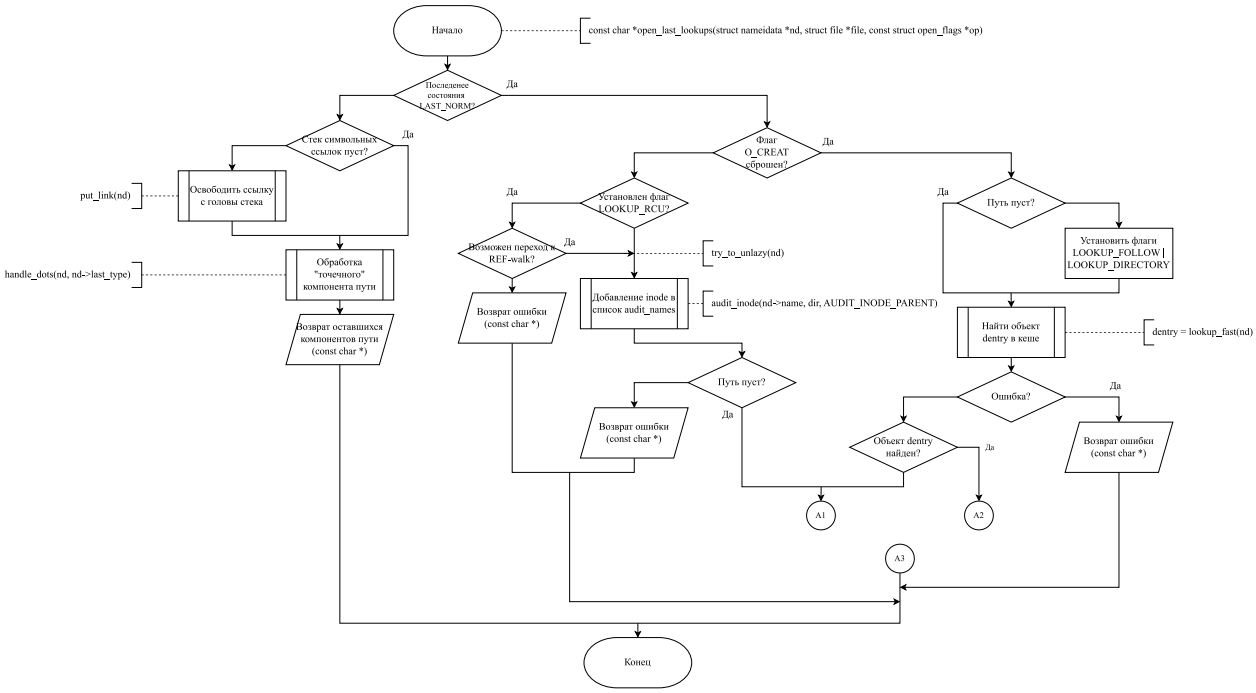

Рисунок 2.17 – Схема алгоритма функции open_last_lookups (часть 1)

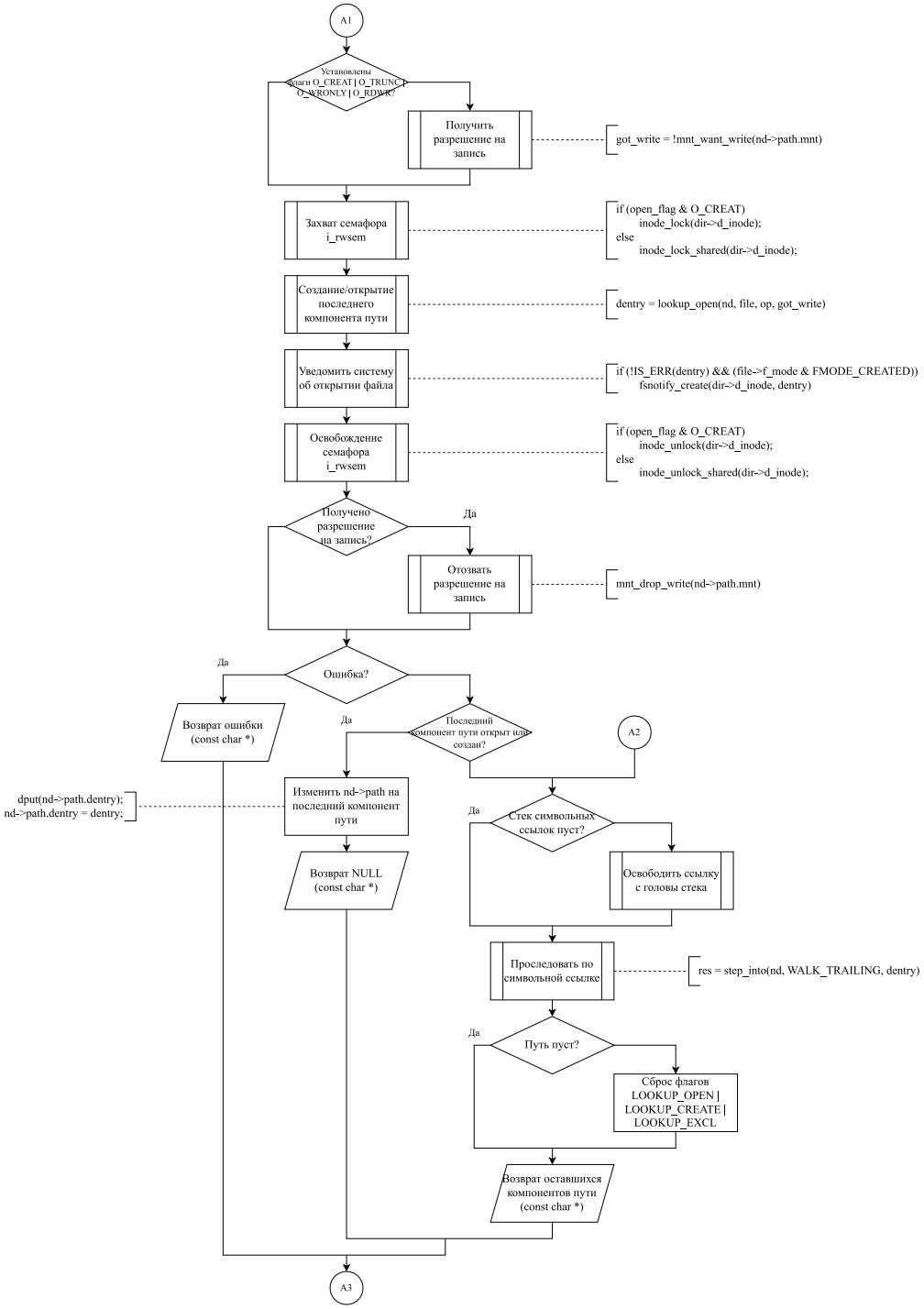

Рисунок 2.18 – Схема алгоритма функции open_last_lookups (часть 2)

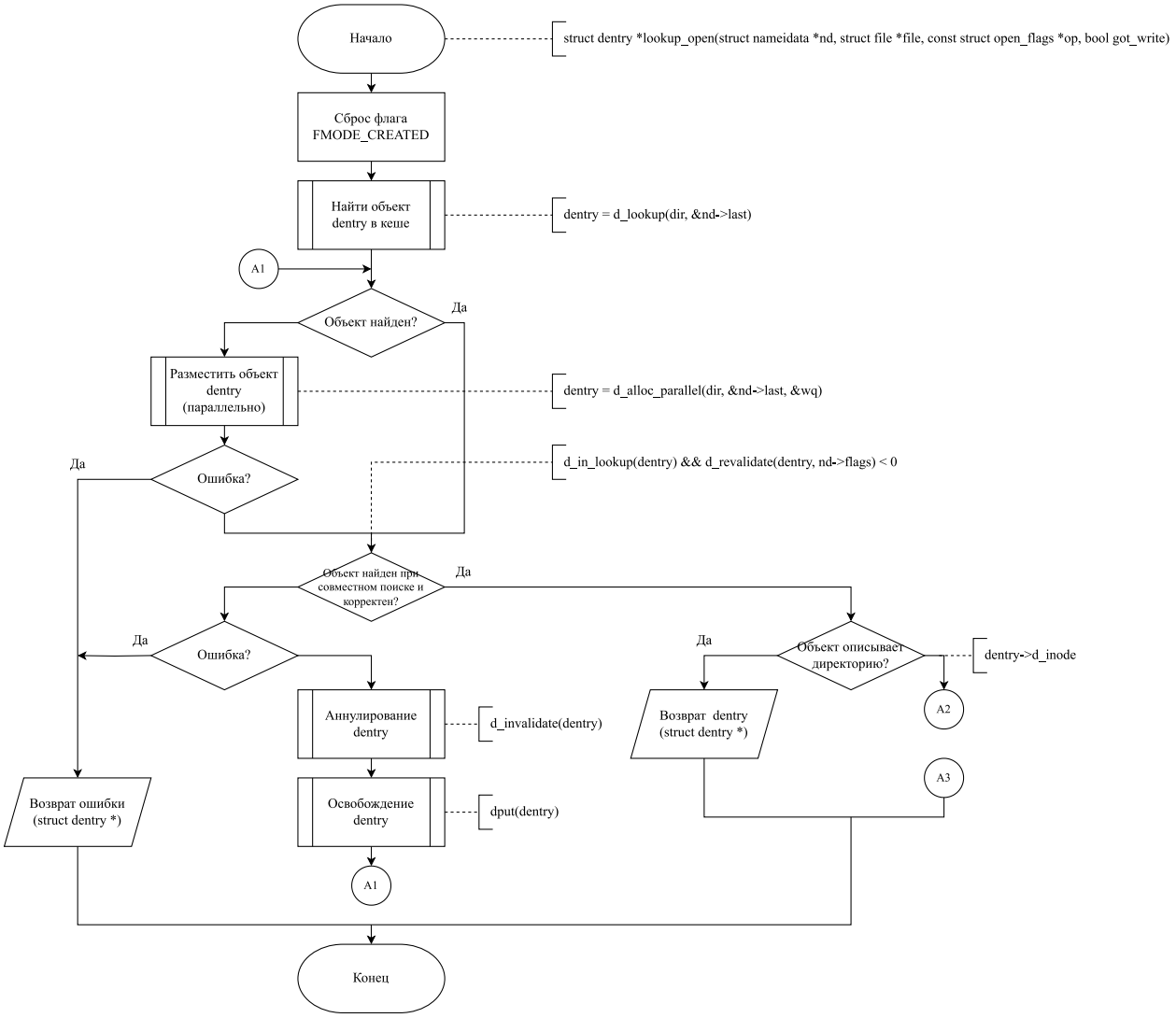

Рисунок 2.19 – Схема алгоритма функции lookup_open (часть 1)

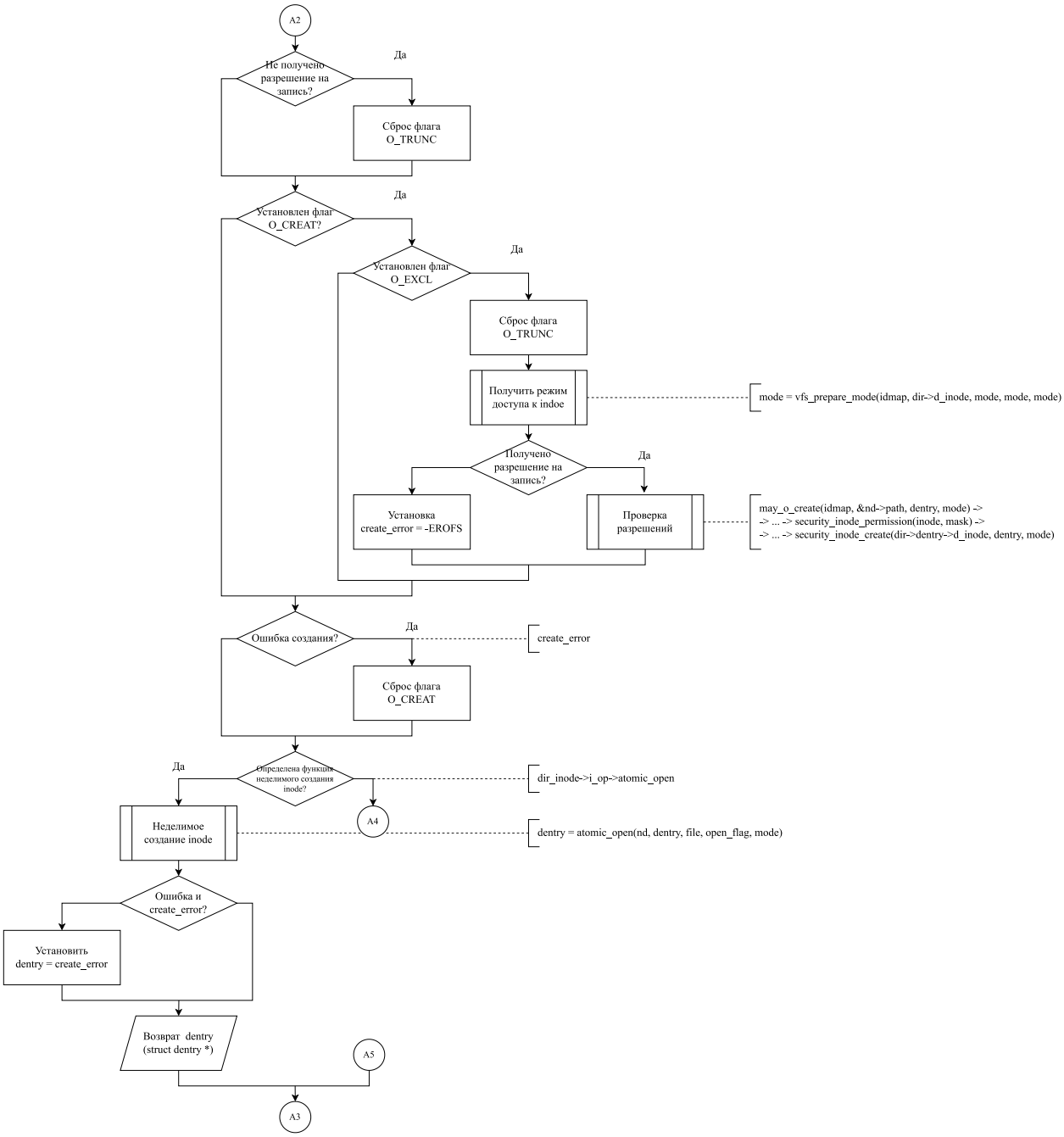

Рисунок 2.20 – Схема алгоритма функции lookup_open (часть 2)

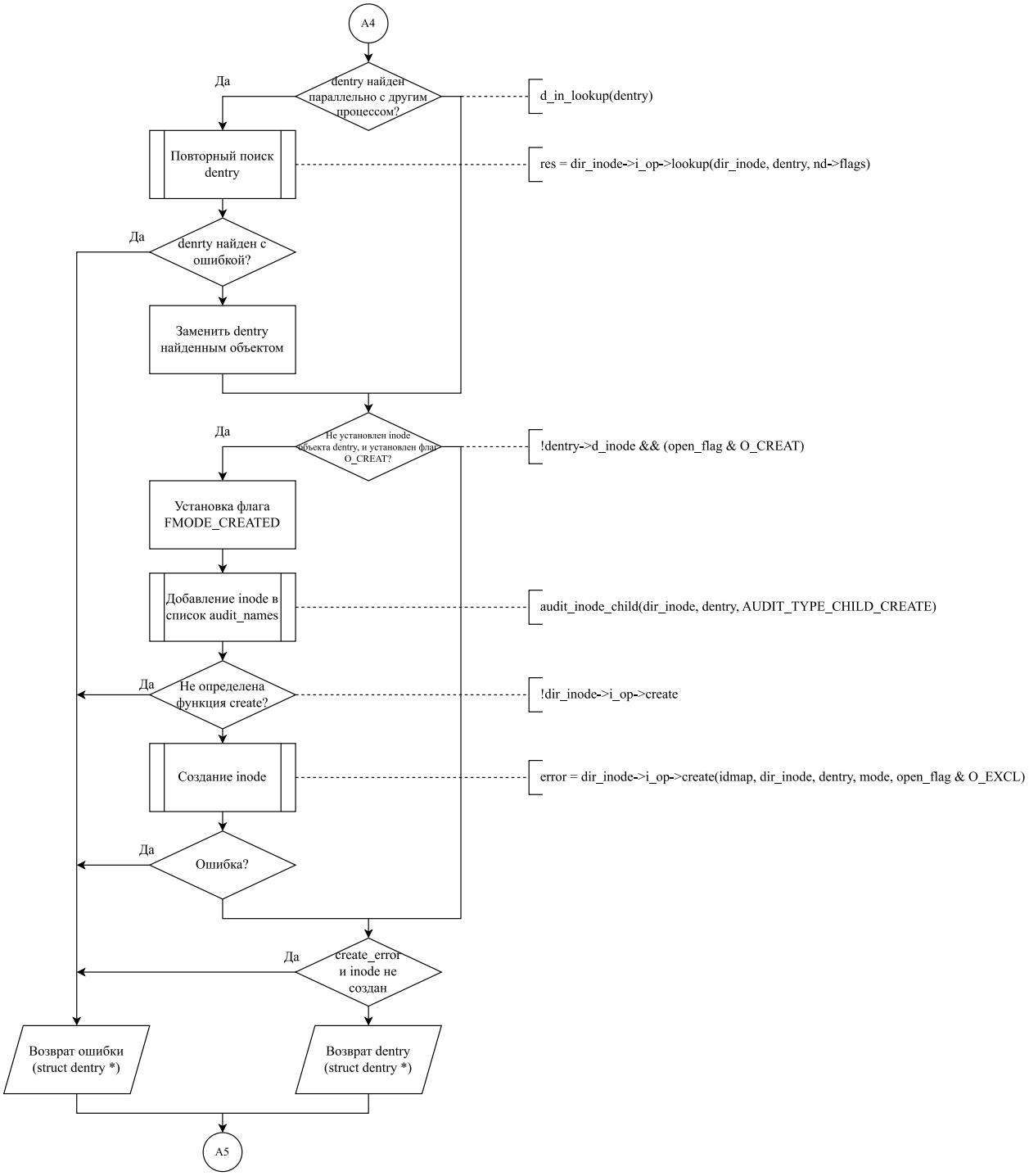

Рисунок 2.21 – Схема алгоритма функции lookup_open (часть 3)

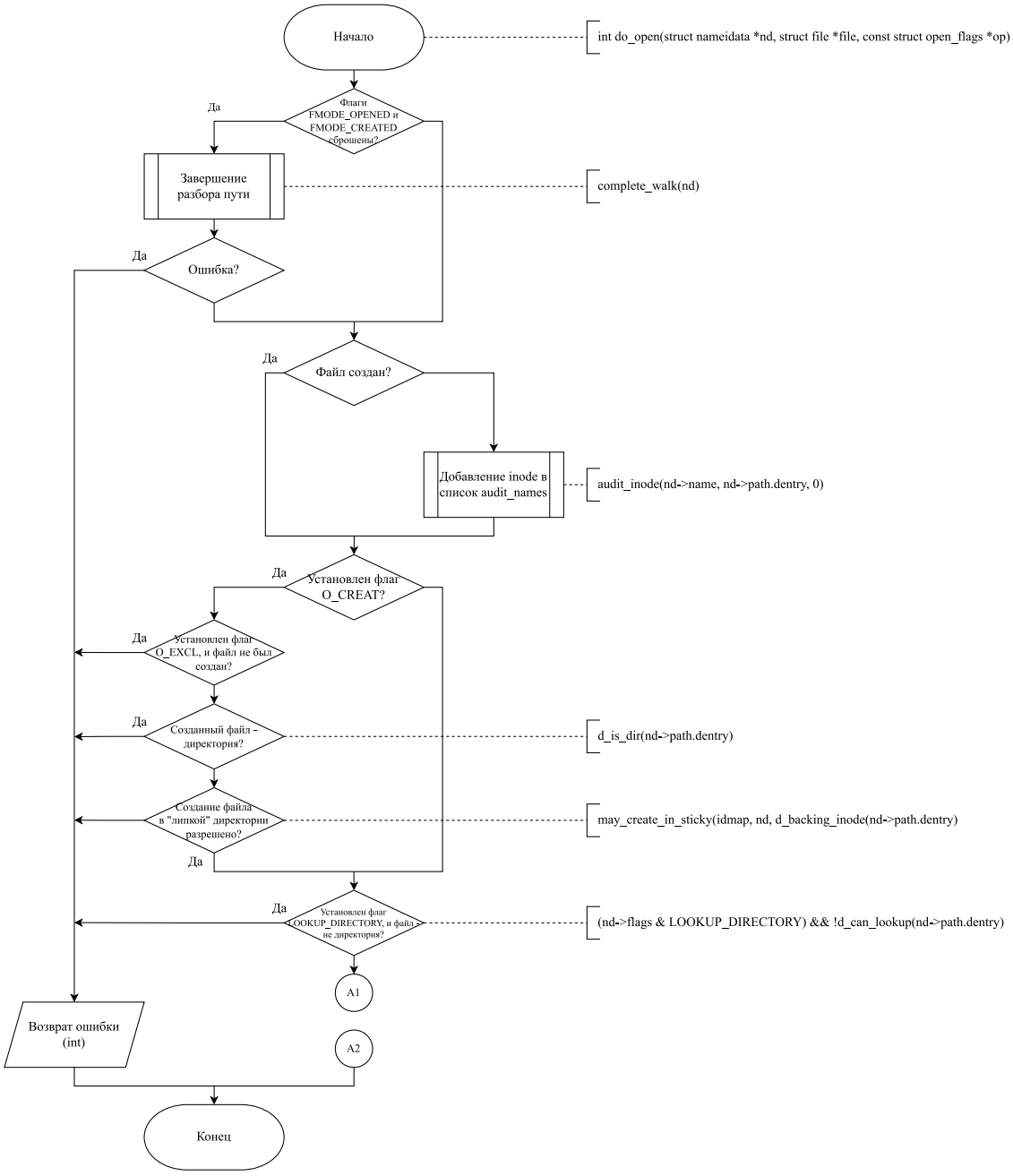

Рисунок 2.22 – Схема алгоритма функции do_open (часть 1)

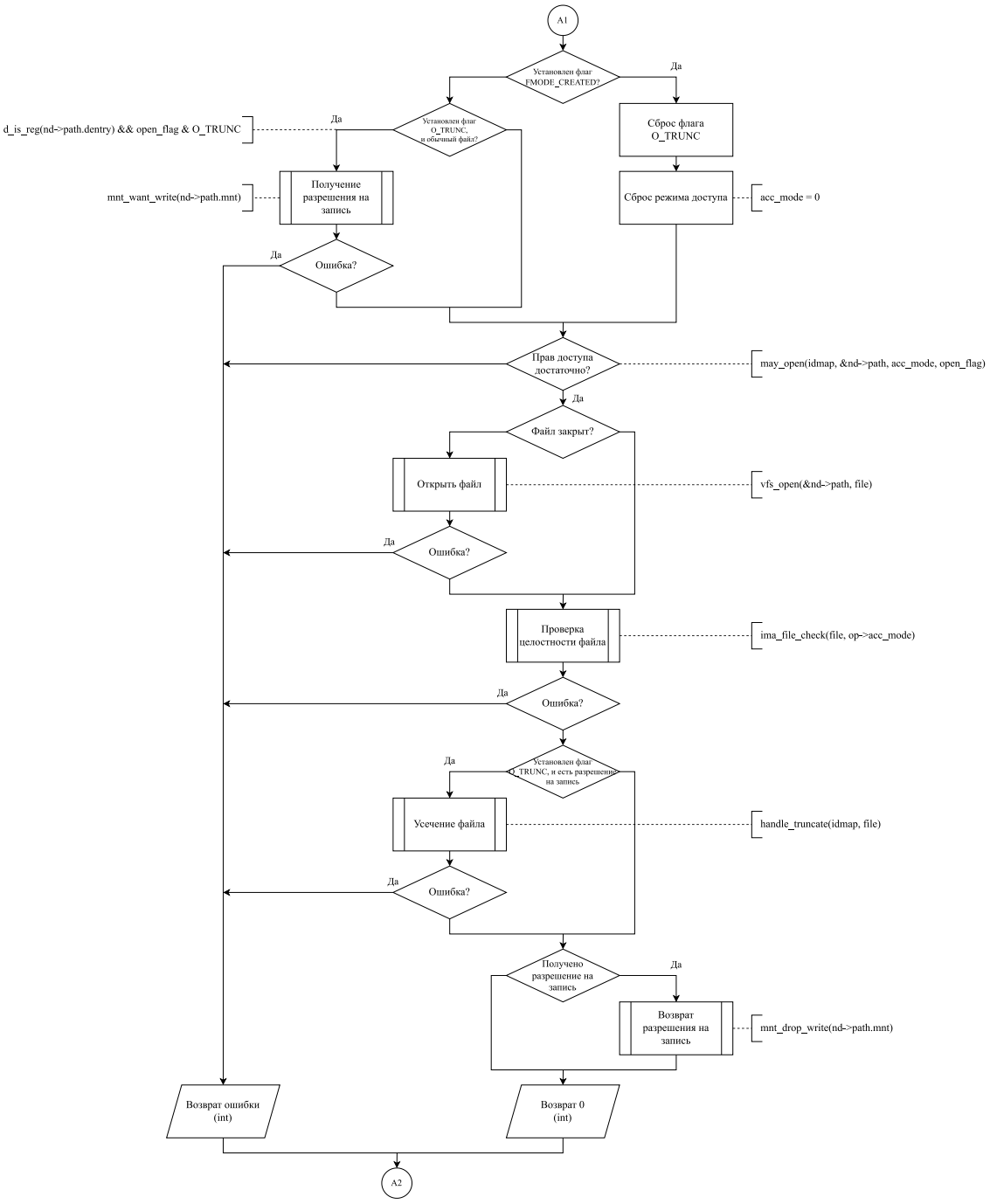

Рисунок 2.23 – Схема алгоритма функции do_open (часть 2)