Содержание

errno.h

заголовочник, предоставляющий механизм сигнализации и диагностики ошибок через глобальную переменную и набор макросов-кодов ошибок. Используется совместно с системными и библиотечными вызовами для понимания причин неуспешных операций.

определения:

```
#include <errno.h>

extern int errno;                   /* глобальная переменная с кодом последней ошибки */

#define EDOM    /* аргумент вне области определения (матем. функции) */
#define ERANGE  /* результат вне диапазона представимых значений */
#define EILSEQ  /* некорректная последовательность байтов (напр., при преобразовании кодировок) */

#define EACCES  /* отказ в доступе (файлы/ресурсы) */
#define ENOENT  /* файл или каталог не найден */
#define EEXIST  /* файл уже существует */
#define ENOMEM  /* недостаточно памяти */
#define EBADF   /* неверный файловый дескриптор */
#define EAGAIN  /* ресурс временно недоступен (повторите позже) */
#define EINTR   /* операция прервана сигналом */
#define EINVAL  /* невалидный аргумент */
#define EIO     /* ошибка ввода-вывода */
#define ENOSPC  /* недостаточно места на устройстве */
#define EPIPE   /* запись в pipe без читателя */

char *strerror(int errnum);      /* получить строковое описание кода ошибки */
int strerror_r(int errnum, char *buf, size_t buflen); /* потокобезопасная версия (две реализации: POSIX/GNU) */
void perror(const char *s);      /* вывести сообщение об ошибке на stderr (s: префикс) */
```

понятия:

- errno — глобальный код ошибки, устанавливается функциями при неудаче.
- Макросы E* — символические кодовые имена ошибок (платформо-зависимы в наборе и значениях).
- strerror/strerror_r — преобразование кода ошибки в человекочитаемую строку.
- perror — простой вывод сообщения об ошибке с использованием errno.
- Виды ошибок — ошибки файловой системы, памяти, аргументов, ввода/вывода, прерываний и др.
- Потокобезопасность — errno зачастую реализуется как макрос, дающий поток-локальную переменную; strerror не потокобезопасна, strerror_r — предпочтительна.

особенности:

- errno имеет значение только после вызова функции, которая явно его устанавливает; при успехе значение не обнуляется автоматически.
- Проверяйте возвращаемое значение функции (например, -1 или NULL) перед чтением errno.
- Значения и набор кодов ошибок могут различаться между системами; полагаться на конкретные численные значения не рекомендуется — используйте макросы.
- strerror_r имеет две несовместимые реализации: POSIX (возвращает 0 при успехе и заполняет буфер) и GNU (возвращает указатель на строку — возможно внутр. статическая строка).
- Не записывайте прямо в errno; присваивание может иметь смысл только при реализации или для тестирования, но обычно errno устанавливается системой/библиотекой.
- При написании многопоточных программ используйте потоко-локальные errno и потокобезопасные функции описания ошибок.

пример использования:

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main(int argc, const char * argv[])
{
    // Generate unique filename.
    char *file_name = tmpnam((char[L_tmpnam]){0});

    errno = 0;
    FILE *file = fopen(file_name, "rb");

    if (file) {
        // Do something useful.
        fclose(file);
    } else {
        perror("fopen() ");
    }

    return EXIT_SUCCESS;
}
```