СИСТЕМНЫЕ ВЫЗОВЫ ВВОДА И ВЫВОДА

Системные вызовы и библиотеки Unix SVR4

Иртегов Д.В.

ФФ/ФИТ НГУ

Электронный лекционный курс подготовлен в рамках реализации

Программы развития НИУ-НГУ на 2009-2018 г.г.

По завершении этого раздела вы сможете:

- описать характеристики файла
- открыть/закрыть файл
- читать и изменять данные в файле
- изменять позицию чтения/записи файла
- создавать копию файлового дескриптора
- управлять доступом к файлу

ЧТО ТАКОЕ ФАЙЛ ?

- последовательность байтов
- операционная система не накладывает никакого формата
- адресация с точностью до байта
- дисковый файл автоматически расширяется при записи
- метка конца файла не входит в данные файла
- файл также является универсальным интерфейсом с внешним устройством

open(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <fcntl.h>
#include <sys/stat.h>
int open (const char *path,
  int oflag, ... /* mode_t mode */);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - неотрицательный дескриптор файла
неуспех - -1 и errno установлена

Флаги open

- O_RDONLY Открывает файл для чтения.
- O_WRONLY Открывает файл для записи.
- O_RDWR Открывает файл для чтения и для записи.
- O_APPEND Перед каждой записью помещает указатель файла в конец файла. Иными словами, все операции записи будут происходить в конец файла.
- O_CREAT Создает файл, если он не существует.
- O_TRUNC Стирает данные файла, устанавливая размер файла равным нулю.
- O_EXCL Используется совместно с O_CREAT. Вызывает неуспех open(2), если файл уже существует.

Флаги open (продолжение)

- O_SYNC Заставляет write(2) ожидать окончания физической записи на диск.
- O_NDELAY,O_NONBLOCK Открытие специального байт-ориентированного файла или именованного программного канала часто вызывает блокировку. Любой из этих флагов предотвращает блокировку open(2). Если установлены оба флага, O_NONBLOCK получает приоритет.
- O_NOCTTY Не открывает файл терминала как управляющий терминал.

open - примеры

- открытие файла для чтения

```
acctfd = open(account, O_RDONLY);
```

- открытие файла для записи

```
file = TMPFILE;
fd = open(file, O_WRONLY | O_CREAT |
  O_TRUNC, 0600);
```

- открытие файла для дописывания

```
logfd = open("/sys/log", O_WRONLY |
  O_APPEND | O_CREAT, 0600);
```

open – примеры (продолжение)

- открытие файла для чтения и записи

```
fdin = open( argv[1], O_RDWR );
```

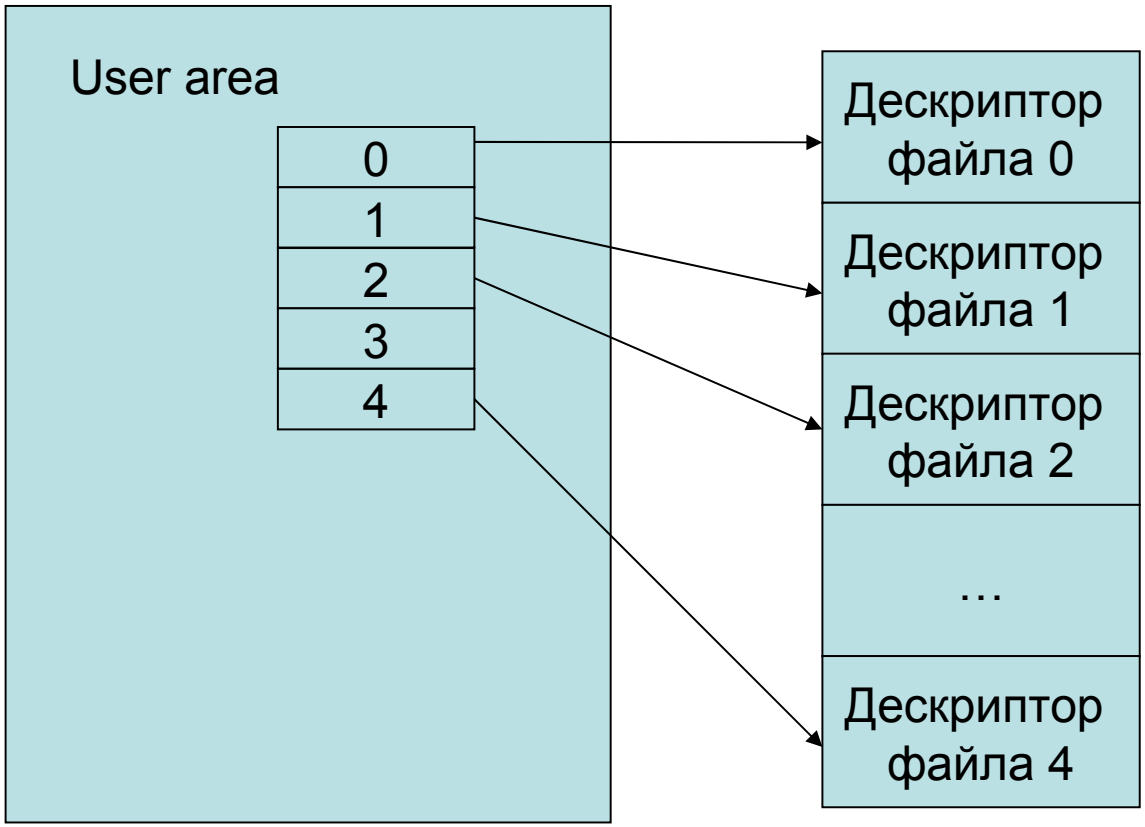
- создание нового файла для записи

```
if ((fdout = open(TMPFILE,
  O_WRONLY | O_CREAT | O_EXCL,
  0666)) == -1)
  perror(TMPFILE);
```

Что делает open

- файл ищется в иерархии директорий для получения inode-номера.
- проверяются права доступа файла
- в таблице дескрипторов размещается новый дескриптор.
- проверяются системные структуры файлов и, если необходимо, размещается новое поле.
- если необходимо, размещается новая структура информации о файле.
- соединяется с подходящим драйвером устройства.
- возвращается файловый дескриптор (индекс в таблице файловых

Что делает open(2)



close (2)

ИСПОЛЬЗОВАНИЕ

```
#include <unistd.h>
int close(int fildes);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1, errno присвоен код ошибки

read (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <unistd.h>
ssize_t read( int fildes, void *buf,
          size_t nbyte);
#include <sys/uio.h>
ssize_t readv(int fildes,
          struct iovec *iov, int iovcnt);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - количество прочитанных байт

неуспех - -1 и errno установлена

write (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types/h>
#include <unistd.h>
ssize_t write( int fildes, const void *buf,
  size_t nbyte);
#include <sys/uio.h>
ssize_t writev(int fildes,
  const struct iovec *iov, int iovcnt);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - количество записанных байт

неуспех - -1 и errno установлена

КОПИРОВАНИЕ ВВОДА В ВЫВОД - ПРИМЕР

```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  main()
6  {
7    char buf[BUFSIZ];
8    int n;
9
10   while ((n = read(0, buf, BUFSIZ)) > 0)
11   write(1, buf, n);
12   exit(0);
13 }
```

fsync (2)

ИСПОЛЬЗОВАНИЕ

```
#include <unistd.h>
int fsync(int fildes);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1 и errno установлена

lseek (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek( int fildes,
        off_t offset, int whence);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - расстояние в байтах от начала файла

неуспех - -1 и errno установлена

Параметр whence

- SEEK_CUR — от текущей позиции
- SEEK_SET — от начала файла
- SEEK_END — от конца файла

Разреженные файлы

- «Дырка» - участок файла, в который никогда не было записи
- Считается в длину файла
- При чтении считываются нули
- Дисковое пространство не выделяется

dup (2)

ИСПОЛЬЗОВАНИЕ

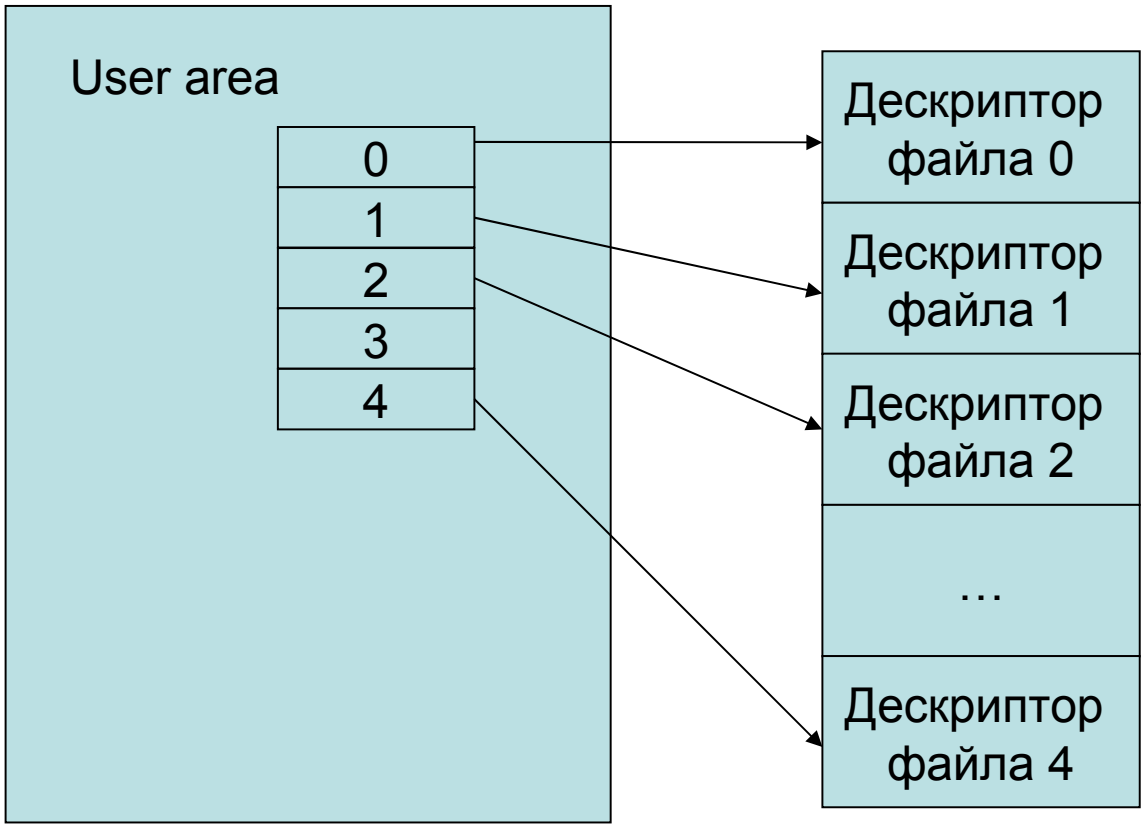
```
#include <unistd.h>
int dup (int fildes);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - неотрицательный файловый дескриптор

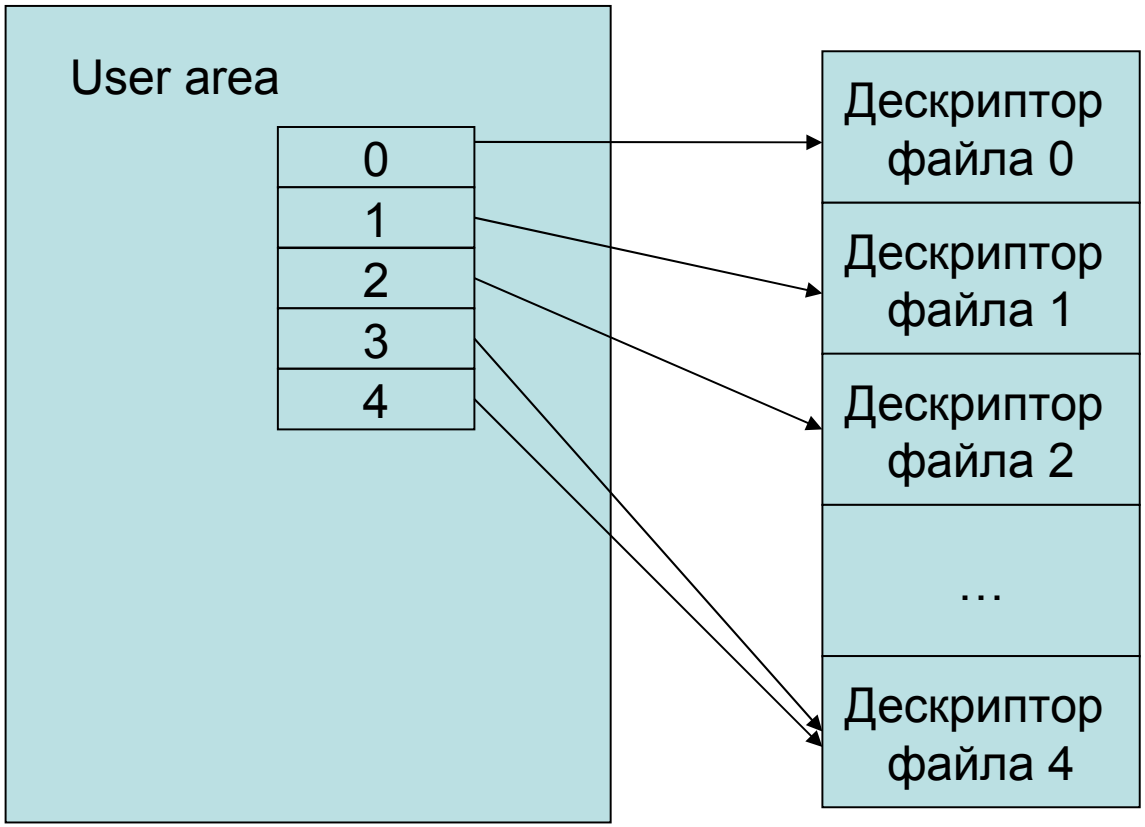
неуспех - -1 и errno установлена

Что делает dup (2)



Что делает dup (2)

- `i=dup(4);`



fcntl (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
int fcntl(int fildes, int cmd,
  ... /* arg */ );
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - зависит от команды cmd

неуспех - -1 и errno установлена

Команды fcntl(2)

- без arg
  - F_GETFD - получить состояние флага закрытия-по-exec
  - F_GETFL - получить флаги файла (NDELAY, NONBLOCK, SYNCH, APPEND)
- int arg
  - F_DUPFD - скопировать файловый дескриптор
  - F_SETFD - установить флаг закрытия-по-exec
  - F_SETFL - установить флаги файла
- struct flock *arg
  - F_FREESP - освободить физический носитель
  - F_GETLK - получить информацию о захватах записи
  - F_SETLK - захватить запись
  - F_SETLKW- захватить запись (с блокировкой)

struct flock

```
typedef struct flock
{
  short l_type;
  short l_whence;
   /* SEEK_SET, SEEK_CUR, SEEK_END */
  off_t l_start;
  off_t l_len;
   /* len == 0 means until end of file */
  long l_sysid;
  pid_t l_pid;
  long pad[4]; /* reserve area */
} flock_t;
```

mmap (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/mman.h>
caddr_t mmap( caddr_t addr,
  size_t len, int prot,
  int flags, int fd, off_t off);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - адрес

неуспех - NULL и errno установлена

mmap - параметры

prot	PROT_READ можно читать
	PROT_WRITE можно изменять
	PROT_EXEC можно исполнять

flags	MAP_SHARED разделяемые изменения
	MAP_PRIVATE частные изменения
	MAP_ANON эквивалент mmap /dev/zero

munmap (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/mman.h>
int munmap(caddr_t addr,
           size_t len);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1 и errno установлена

msync (2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types/h>
#include <sys/mman.h>
int msync(caddr_t addr,
          size_t len, int flags);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1 и errno установлена

Флаги msync

- MS_ASYNC немедленно вернуться, как только спланированы все операции записи
- MS_SYNC вернуться, только когда завершатся все операции записи
- MS_INVALIDATE помечает страницы памяти как недействительные. После этого любое обращение к этим адресам вызывает чтение с диска.