Дисциплины
"ПРОГРАММИРОВАНИЕ"

Файловый ввод-вывод

Преподаватель:

Перышкова Евгения Николаевна

Использование файлов

- Информация, размещенная на жестких дисках сохраняется после перезагрузки компьютера.
- Для хранения информации на жестких дисках используются файлы.
- Программа не может работать с данными, расположенными в файле непосредственно. Для обработки необходимо сначала считать их в оперативную память (в переменную или массив).
- В программе файл описывается с помощью специальной структуры данных (FILE *), которая хранит информацию, необходимую для корректного доступа к нему.
- Программа может работать с большим количеством файлов одновременно, вбор нужного файла определяется переменной, хранящей указатель на соответствующую структуру (FILE *).

Количество элементов

Количество элементов в типах данных, рассмотренных к настоящему моменту является конечным:

```
int i;                  <-- 1 элемент типа int
float mas[10];          <-- 10 элементов типа int
struct my{
    int age;
    char name[20];
    struct my *next;
} table[10];            <-- 10 элементов типа struct my
```

Большинство *усложненных* структур , например:

1) последовательности;
2) деревья;
3) графы;

характеризуются тем, что количество элементов в них заранее не определено и может стремиться к бесконечности.

Последовательность

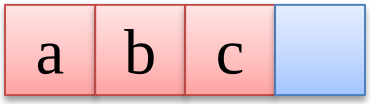
Последовательность с базовым типом T_0 это либо пустая последовательность, либо конкатенация последовательности (с базовым типом T_0) и значения типа T_0.

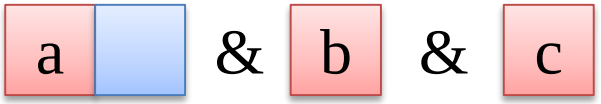
Например, пусть базовый тип – char (символ), тогда:

1) S_1 = ‹ › - пустая последовательность

S_1 

2) S_2 = ‹ a, b, c › = ‹ a, b › & c = (‹ a › & b) & c = ((‹ › &a) & b) & c, где "&" – операция конкатенации (сцепления, склеивания).

S_2 

S_2 

S_2 

S_2 

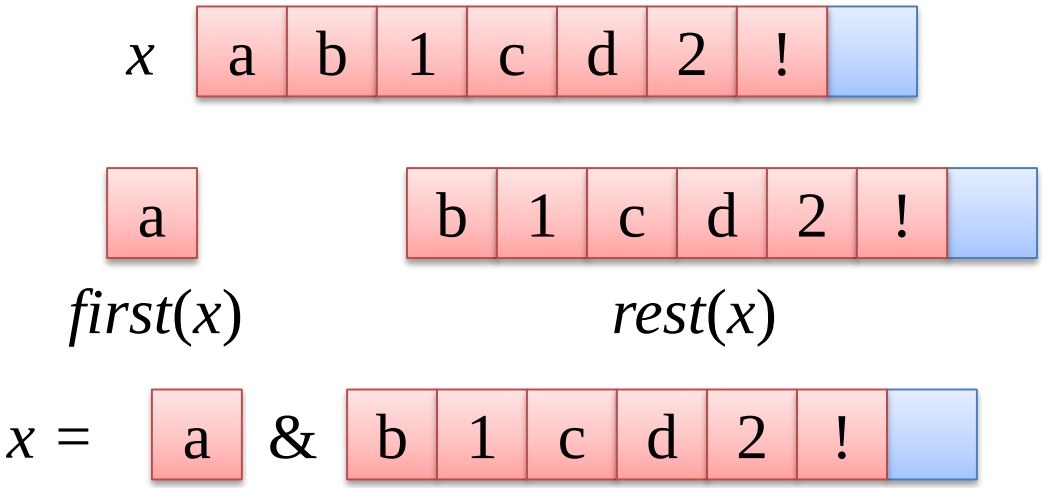
Последовательность

Если $x = \langle x_1, x_2, \ldots, x_n \rangle$ - непустая последовательность, то $first(x) = x_1$ обозначает ее первый элемент.

Если $x = \langle x_1, x_2, \ldots, x_n \rangle$ - непустая последовательность, то $rest(x) = \langle x_2, \ldots, x_n \rangle$ обозначает последовательность без ее первой компоненты.

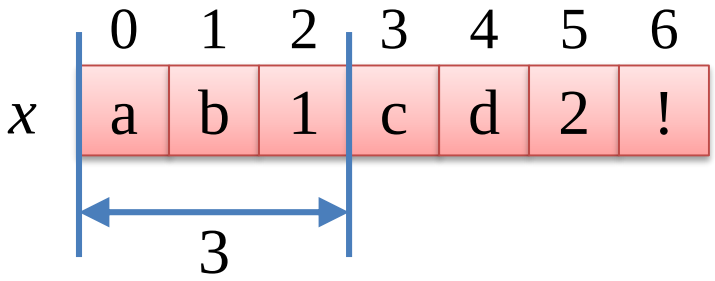
Справедливо: $x \equiv first(x)$ & $rest(x)$.

Число элементов *последовательности* **не ограничено**.



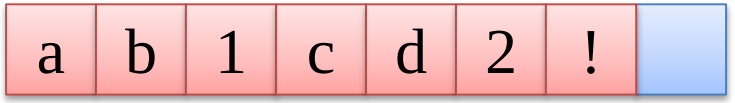
Доступ к элементам

Массив

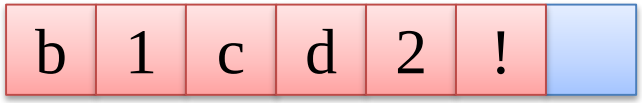



```
x[3] = *(ptr(x) + 3) = 'c'
```

Последовательность




```
x = rest(x)
```




```
x = rest(x)
```


```
x = rest(x)
```

x_3 = first(x)

Области применения

- Массивы и структуры (записи) допускают произвольный доступ к своим элементам. Их используют, размещая в оперативной памяти.

- Последовательности используют для работы с данными на внешних устройствах хранения, допускающих только последовательный доступ (жесткие диски, магнитные ленты и т.д.)

Применение последовательностей

Особенность – возможность прямого доступа
только к элементу *first*(x).

Последовательный доступ, имеющий такое ограничение, позволяет обеспечить:
1)простое управление памятью;
2)применение эффективных методов *буферизации*.

Буферизация – накопление данных из потока в *буфере* и последующую пересылку целиком содержимого буфера, как только он заполнится. Это обеспечивает эффективность управления памятью.

Последовательности обычно используют в случаях, когда данные пересылаются с одного устройства хранения на другое, например, с **жесткого диска в оперативную память** и обратно.

Файловая переменная

Файлы – динамические структуры данных, хранящиеся на внешних запоминающих устройствах.

Файловая переменная – структура данных, связывающая программу с некоторым файлом на диске.

Дисциплина последовательного доступа – ограниченный **набор специальных операций**.

В математическом описании алгоритма выражение s_i для последовательности s имеет значение.

В программе получение значения элемента s_i из файла, содержащего последовательность, **не является элементарной операцией** (требует вычитывания значений с 1 по $i - 1$)

Операции последовательного доступа

Опер.	Описание
f = *opennew*(*n*) *f* = *openold*(*n*)	связывание программы и последовательности *x*, расположенной в файле с именем *n*
close(*f*)	закрытие файлового соединения
write(*f* , *c*)	Запись нового элемента *c* в текущую позицию последовательности *x*, описываемую файловой переменной *f*.
c = *read*(*f*)	Чтение элемента, расположенного в текущей позиции последовательности *x*, описываемой файловой переменной *f*, в ячейку *c*.
set(*f*, *p*)	Установка текущей позиции последовательности *x*, описываемой файловой переменной *f*, в значение смещения *p*.

Операция open

$x_L = < >$
$x_R = < >$

$f = opennew(n_1)$

$f = openold(n_2)$

$x_L = < >$
$x_R = x$

Программа

HDD (жесткий диск)

Файл n_1

x

Файл n_2

x | a | b | 1 | c | d | 2 | A | B | C | D | E | F | G | H |

Операция *write*

$x_L = <>$
$x_R = <>$
$x = x_L \;\&\; x_R$

$f = opennew(n_1)$

Программа

HDD (жесткий диск)

Операция *write*

$x_L = <>$
$x_L = <>$
$x = x_L \,\&\, x_R$

$f = opennew(n_1)$

write(f , 'a')

$x_L = <'a'>$
$x_R = <>$
$x = x_L \,\&\, x_R$

Программа

HDD (жесткий диск)

Операция *write*

$x_L = <>$
$x_R = <>$
$x = x_L \,\&\, x_R$

$f = opennew(n_1)$

write(f , 'a')

$x_L = <'a'>$
$x_R = <>$
$x = x_L \,\&\, x_R$

$x_L = <'ab'>$
$x_R = <>$
$x = x_L \,\&\, x_R$

write(f , 'b')

Программа

HDD (жесткий диск)

Операция *write*

$x_L = \langle\,\rangle$
$x_R = \langle\,\rangle$
$x = x_L \,\&\, x_R$

$f = opennew(n_1)$

***write*(*f* , 'a')**

$x_L = \langle$ 'a' $\rangle$
$x_R = \langle\,\rangle$
$x = x_L \,\&\, x_R$

***write*(*f* , 'b')**

$x_L = \langle$ 'ab' $\rangle$
$x_R = \langle\,\rangle$
$x = x_L \,\&\, x_R$

***write*(*f* , 'c')**

$x_L = \langle$ 'abc' $\rangle$
$x_R = \langle\,\rangle$
$x = x_L \,\&\, x_R$

Программа

HDD (жесткий диск)

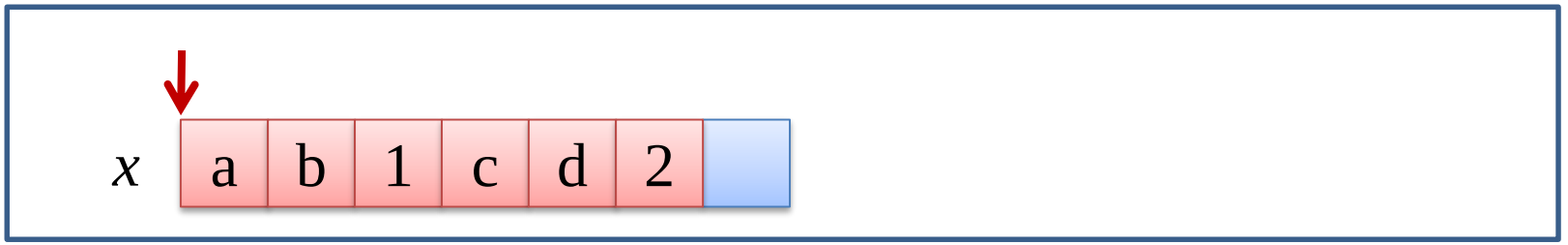
Операция *read*

x_L = < >
x_R = < 'ab1cd2' >
$x = x_L$ & x_R

$f = openold(n_2)$

Программа

HDD (жесткий диск)



Операция *read*

x_L = < >
x_R = < 'ab1cd2' >
$x = x_L$ & x_R

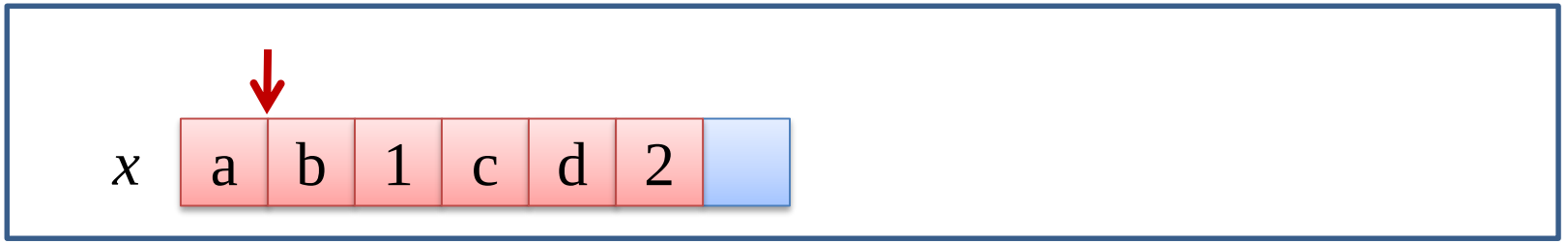
$f = openold(n_2)$

x_L = < 'a' >
x_R = < 'b1cd2' >
$x = x_L$ & x_R

$c = read(f)$

Программа

HDD (жесткий диск)



Операция *read*

x_L = < >
x_R = < 'ab1cd2' >
$x = x_L$ & x_R

$f = openold(n_2)$

x_L = < 'a' >
x_R = < 'b1cd2' >
$x = x_L$ & x_R

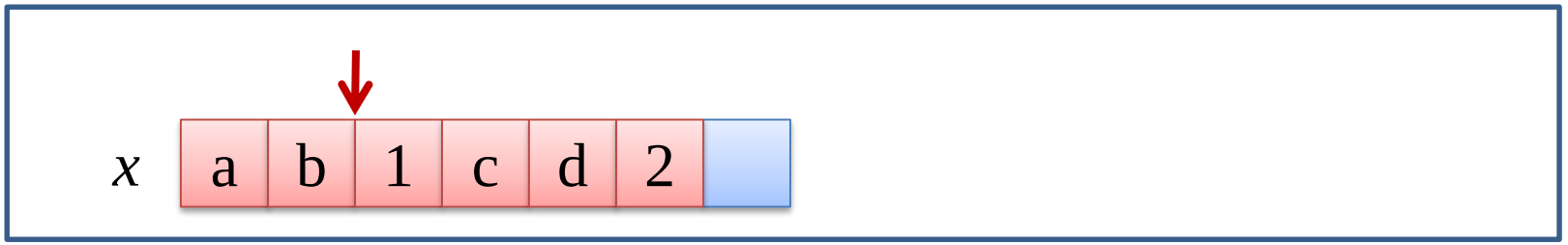
$c = read(f)$

x_L = < 'ab' >
x_R = < '1cd2' >
$x = x_L$ & x_R

$c = read(f)$

Программа

HDD (жесткий диск)



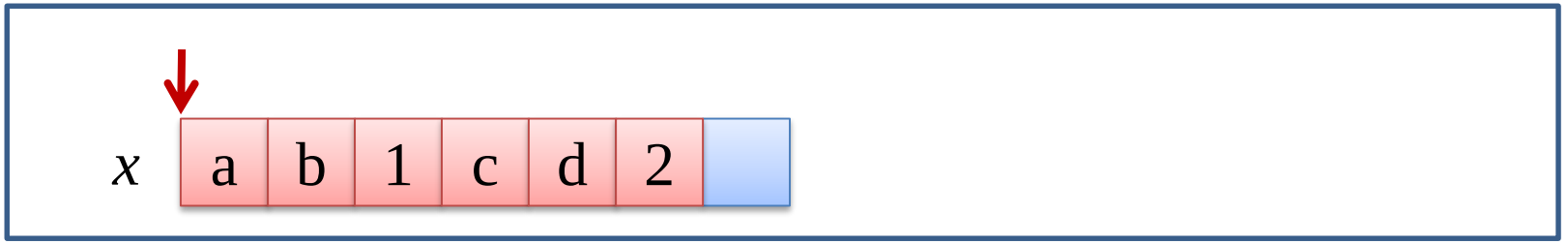
Операция *set*

x_L = < >
x_R = < 'ab1cd2' >
$x = x_L$ & x_R

$f = openold(n_2)$

Программа

HDD (жесткий диск)



Операция *set*

x_L = < >
x_R = < 'ab1cd2' >
$x = x_L$ & x_R

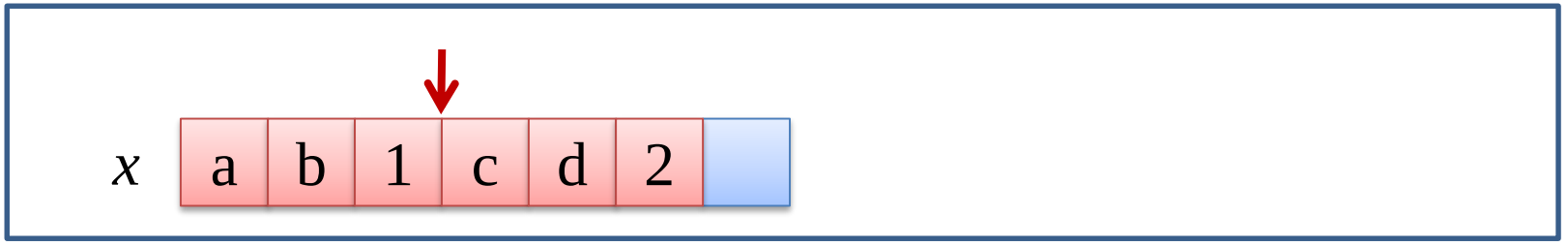
x_L = < 'ab1' >
x_R = < 'cd2' >
$x = x_L$ & x_R

$f = openold(n_2)$

$c = set(f, 3)$

Программа

HDD (жесткий диск)



Средства низкоуровневого ввода-вывода языка СИ

```
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
```

O_RDONLY
O_WRONLY
O_RDWR

```
#include <unistd.h>
int close(int fd);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
```

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

Пример использования низкоуровневого ввода-вывода

Код программы

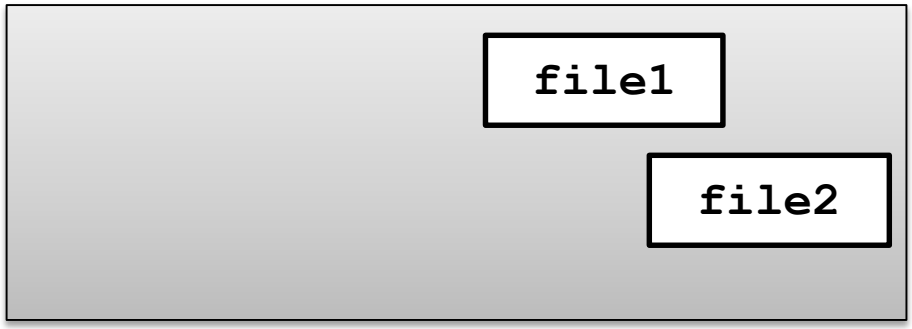
```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```

Память программы

Память ОС



Файловая система

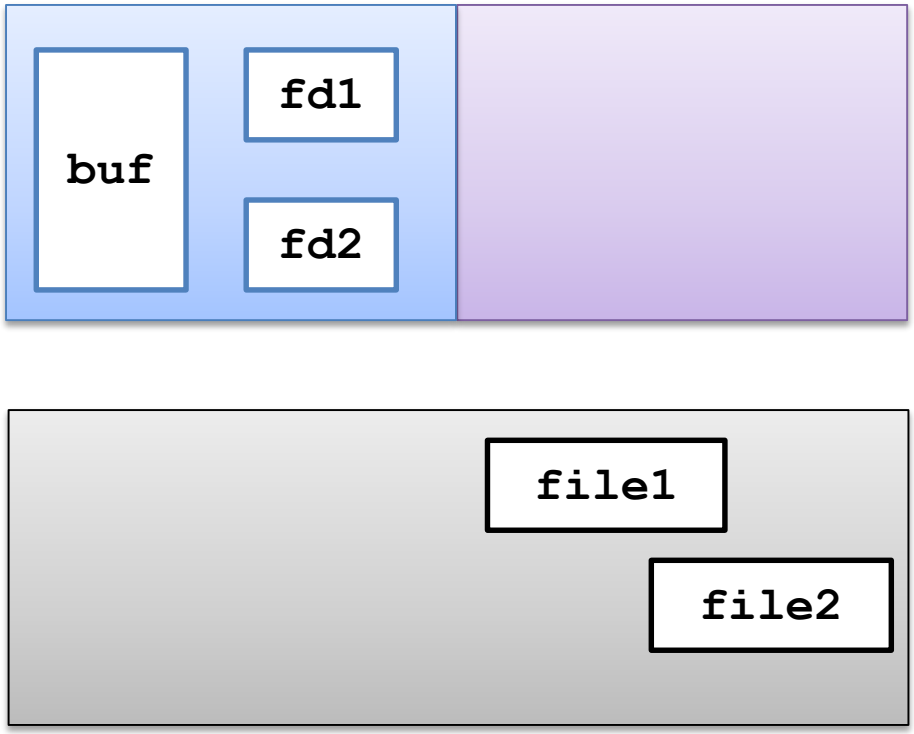
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



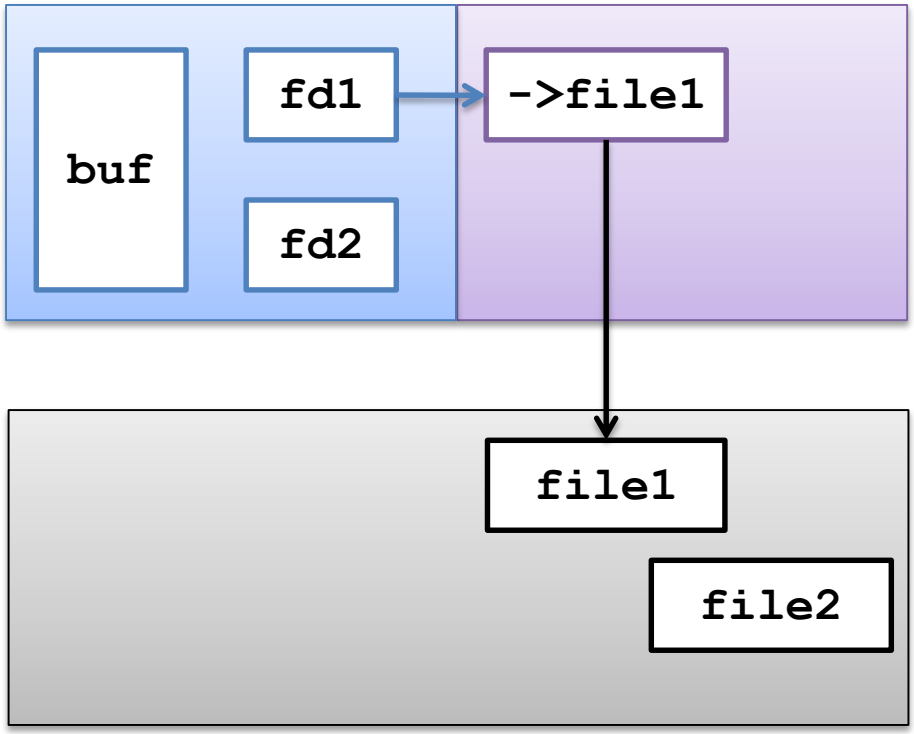
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



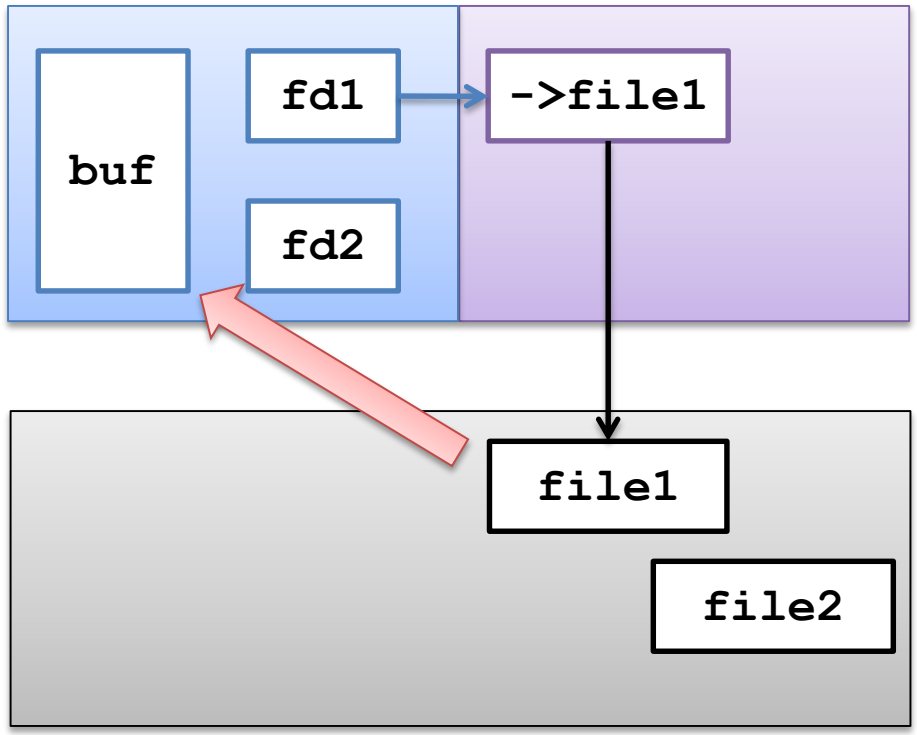
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
                ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



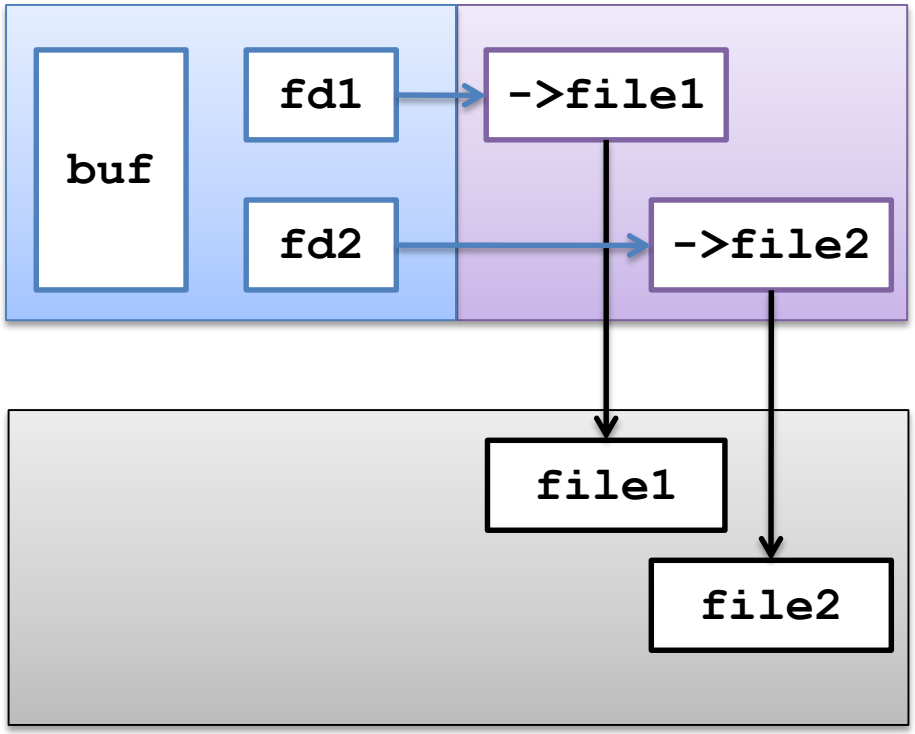
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
                ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



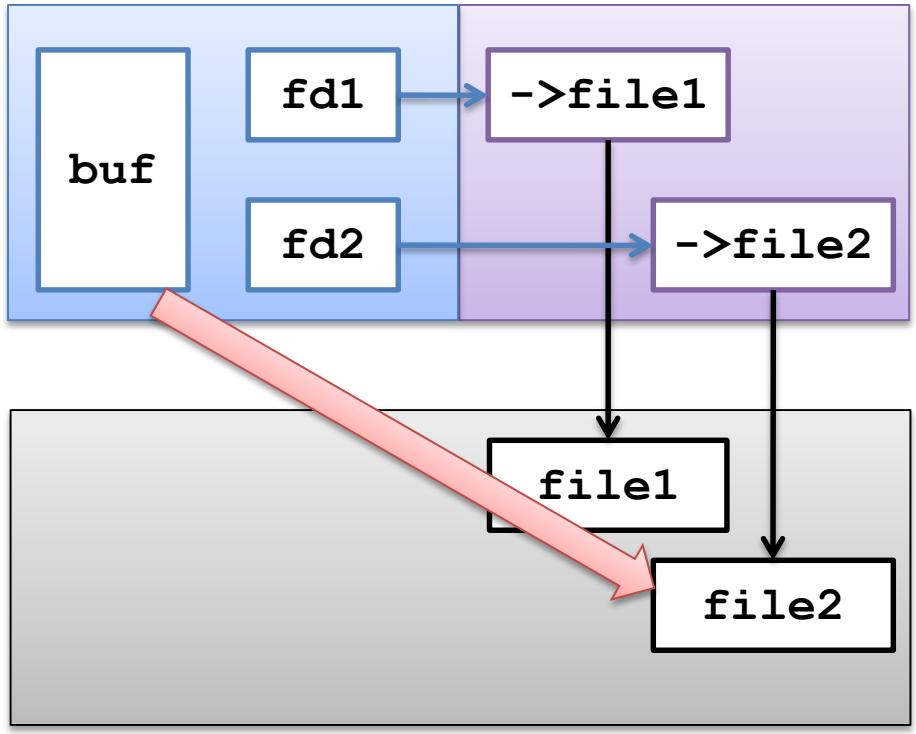
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
                ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



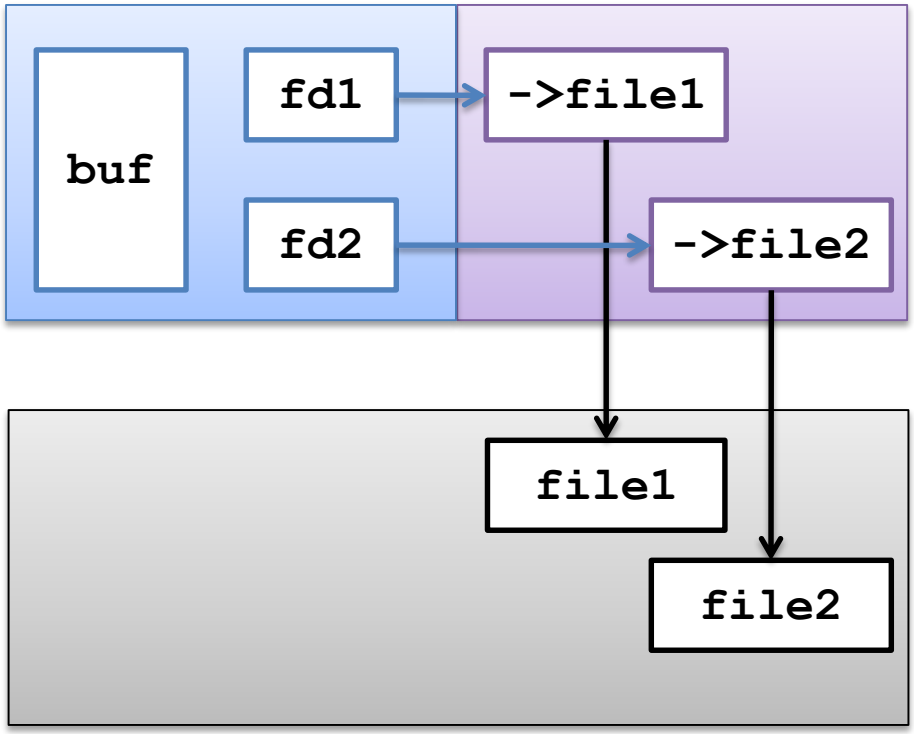
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
                ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



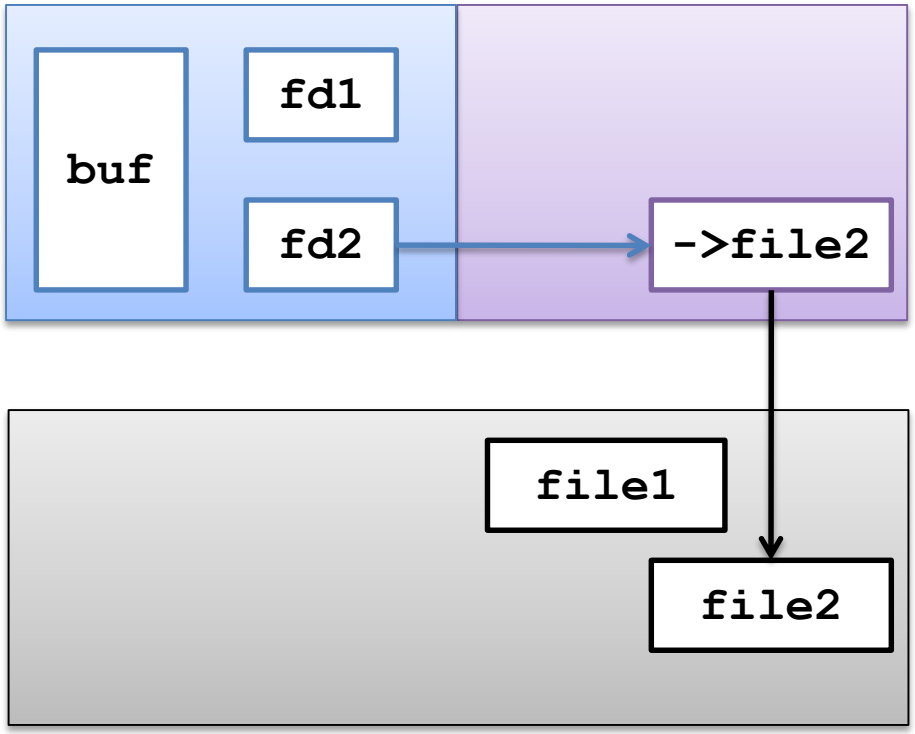
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



Файловая система

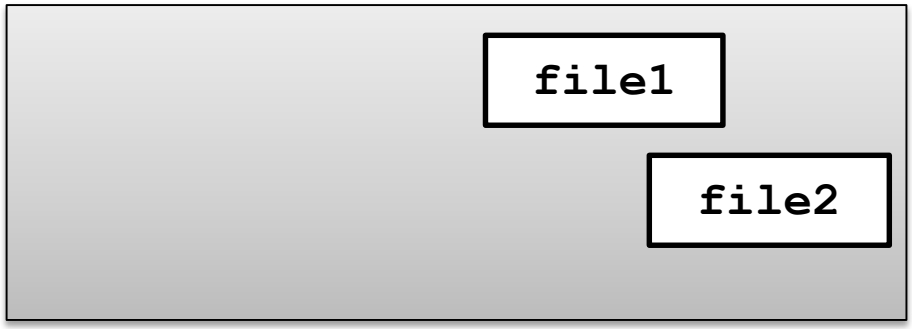
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);

fd2=open("file2",O_WRONLY);
ret2=write(fd2,buf,ret1);

close(fd1);
close(fd2);
```



Файловая система

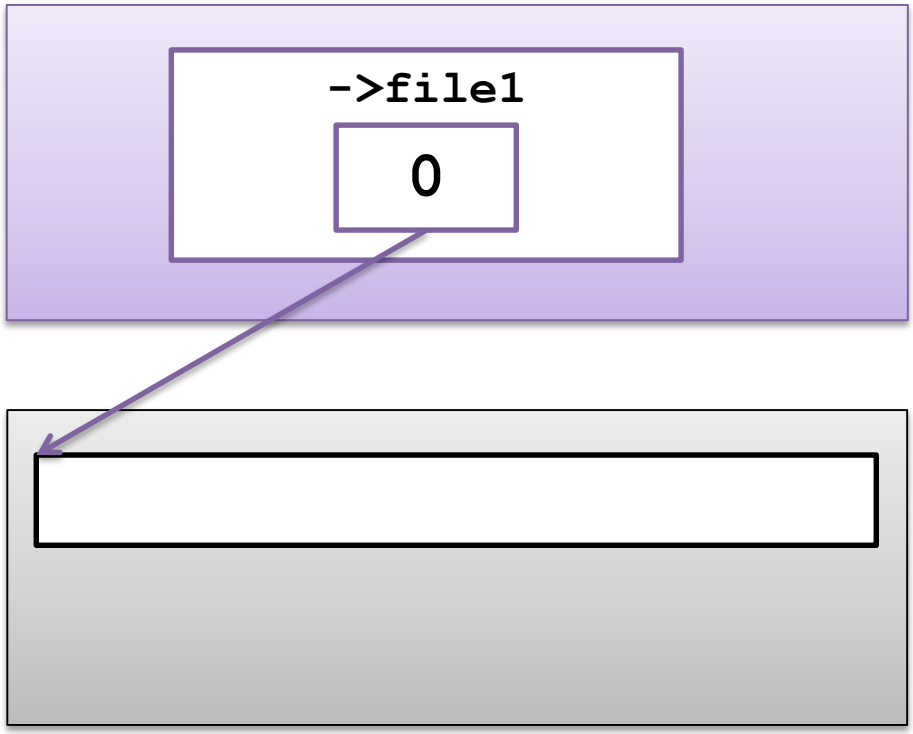
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```



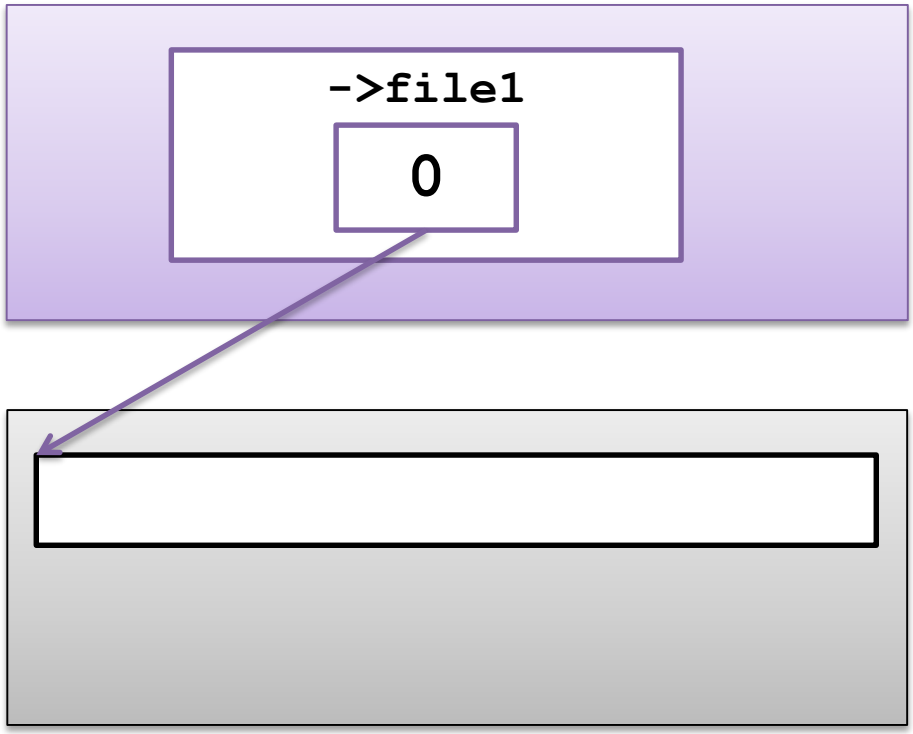
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```



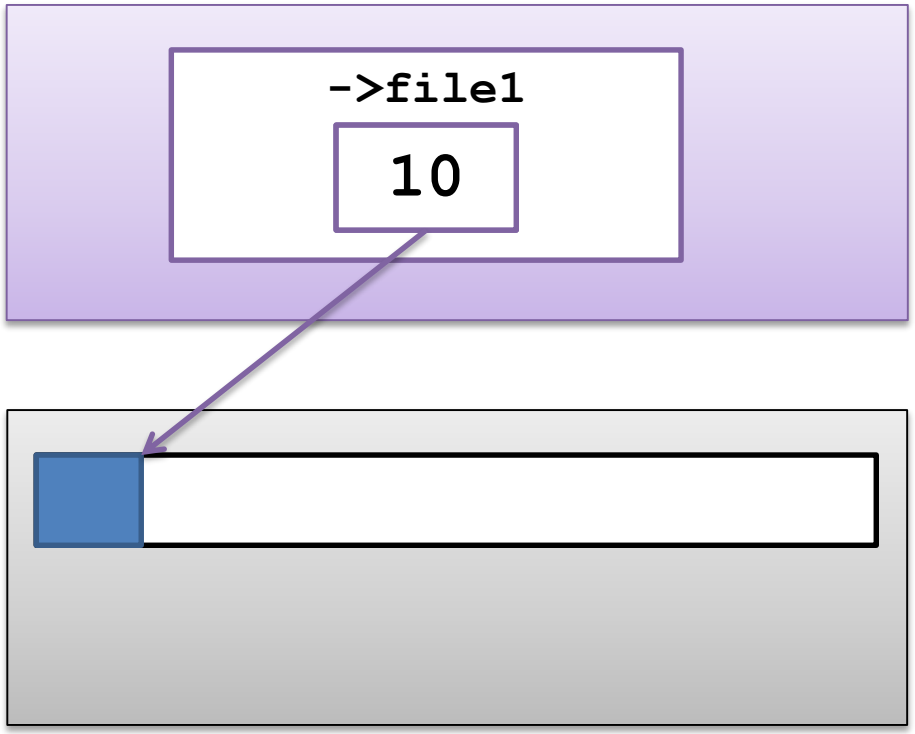
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```



Пример использования низкоуровневого ввода-вывода

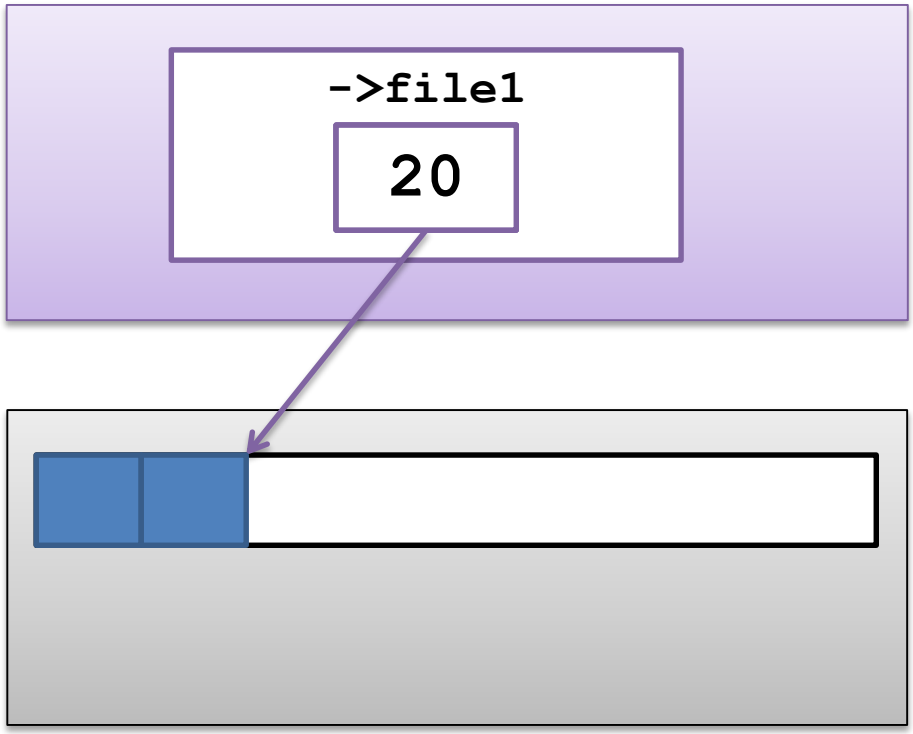
Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```

Память ОС



Файловая система

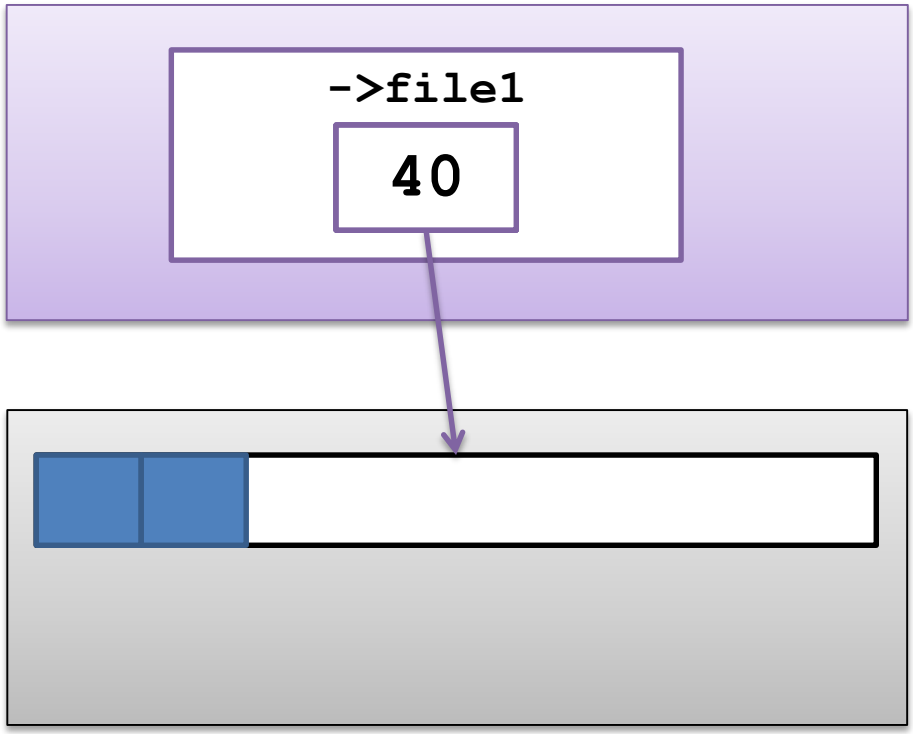
Пример использования низкоуровневого ввода-вывода

Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```



Пример использования низкоуровневого ввода-вывода

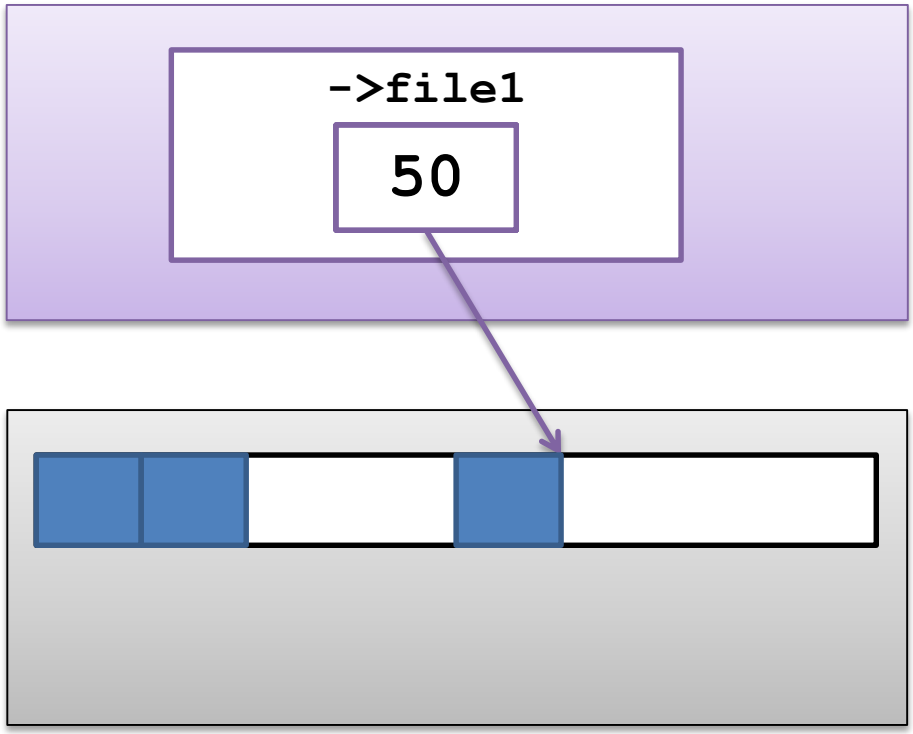
Код программы

```
int fd1, fd2;
int ret1,ret2;
char buf[256];
              ...
fd1=open("file1",O_RDONLY);
ret1 = read(fd1,buf,10);
ret1 = read(fd1,buf,10);

lseek(fd1,40,SEEK_SET);

ret1 = read(fd1,buf,10);
close(fd1);
```

Память ОС



Файловая система

Высокоуровневый интерфейс ввода-вывода языка СИ

```
#include <stdio.h>
```

Управление связанными программными объектами

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *fp);
```

Чтение/запись данных

```
int fscanf(FILE *stream, const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);

char *fgets(char *s, int size, FILE *stream);

size_t fread(void *ptr, size_t size, size_t nmemb,
                                              FILE *stream);
size_t fwrite(const void *ptr, size_t size, size_t nmemb,
                                              FILE *stream);
```

Позиционирование в потоке

```
int fseek(FILE *stream, long offset, int whence);
long ftell(FILE *stream);
void rewind(FILE *stream);
```

Пример использования высокоуровневого ввода-вывода

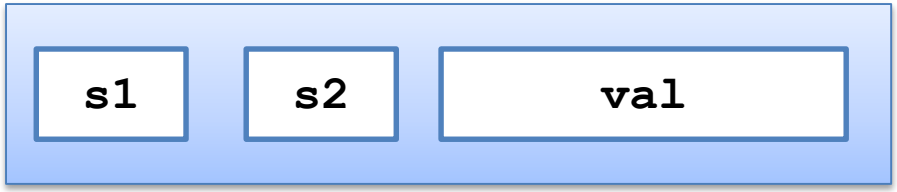
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
              ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Ядро ОС

Станд. библиотека GLibC

Файловая система

Пример использования высокоуровневого ввода-вывода

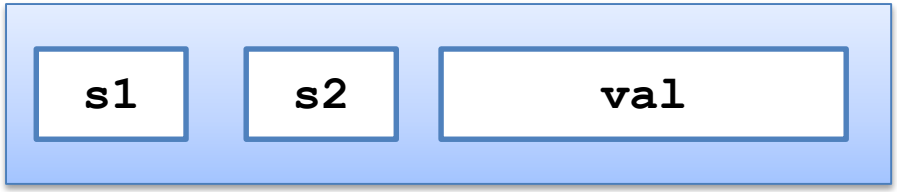
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
              ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Ядро ОС

Станд. библиотека GLibC

Файловая система

Пример использования высокоуровневого ввода-вывода

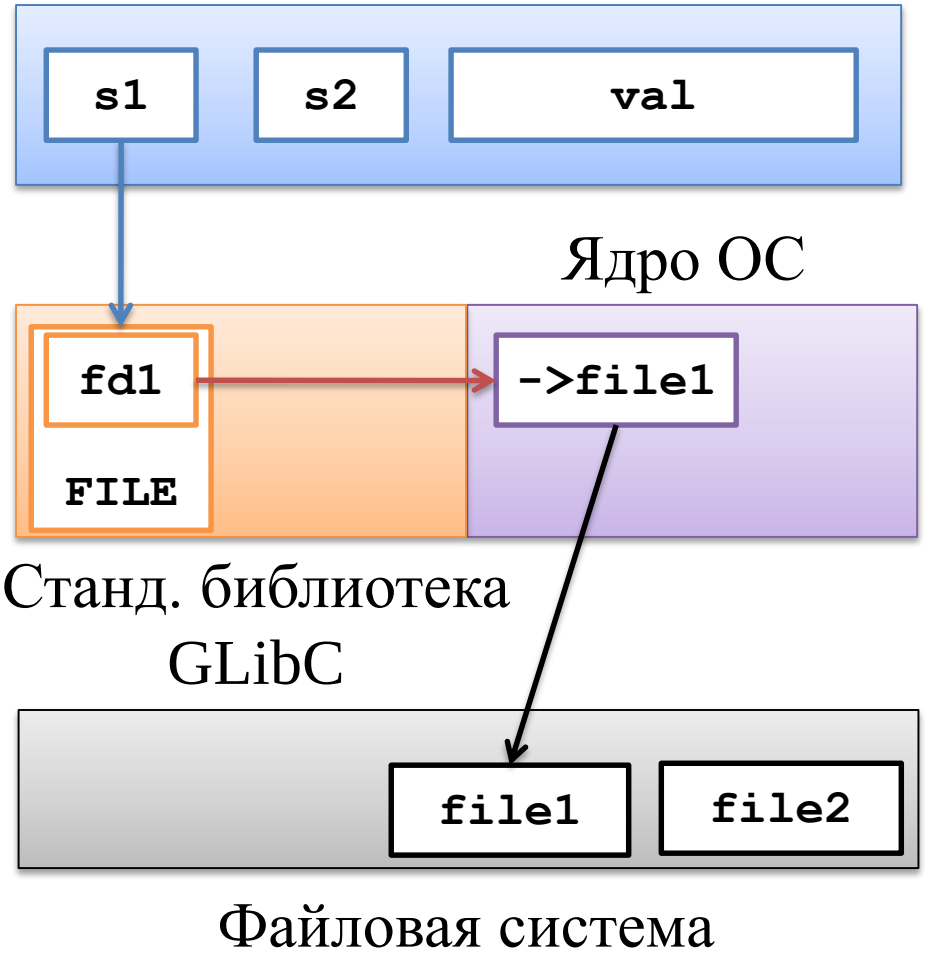
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

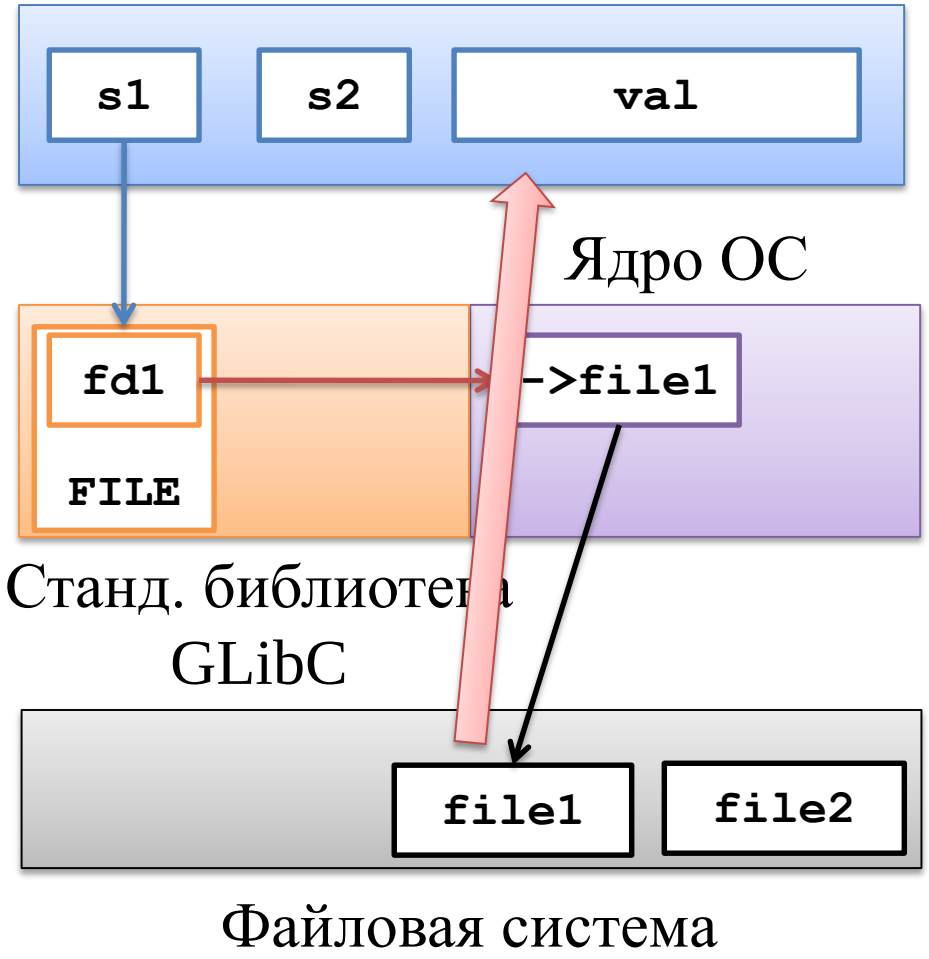
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

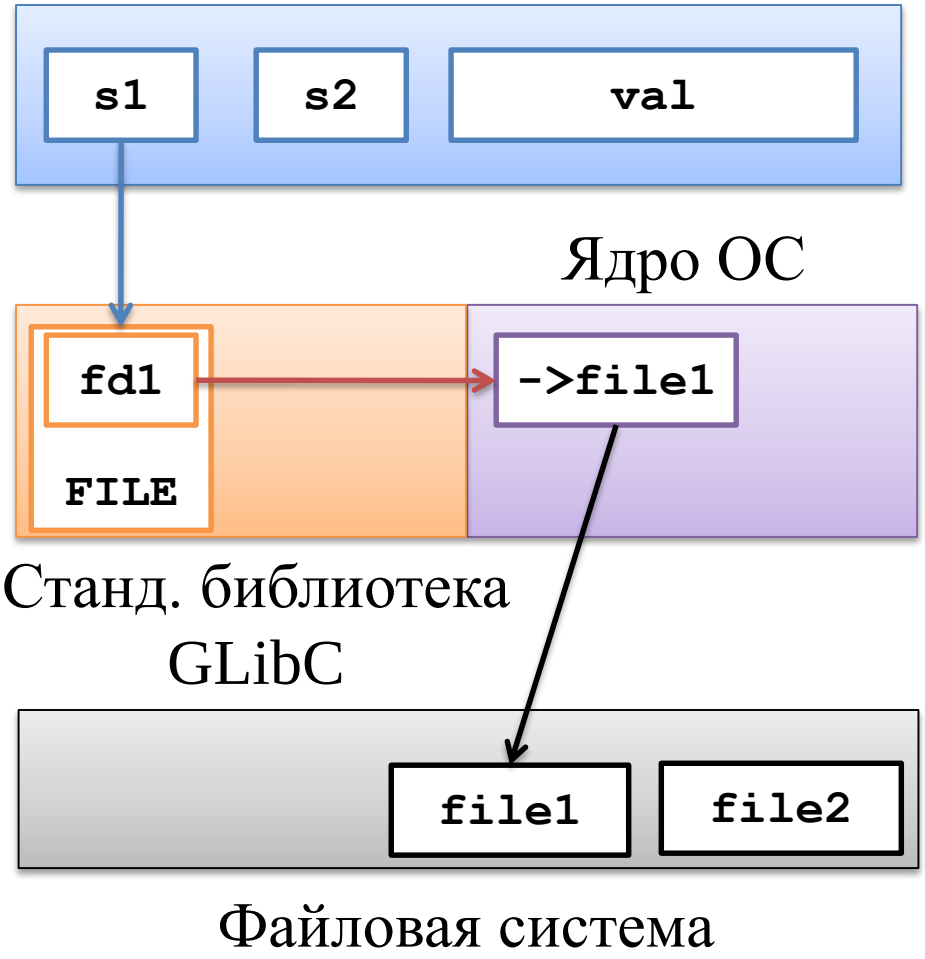
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

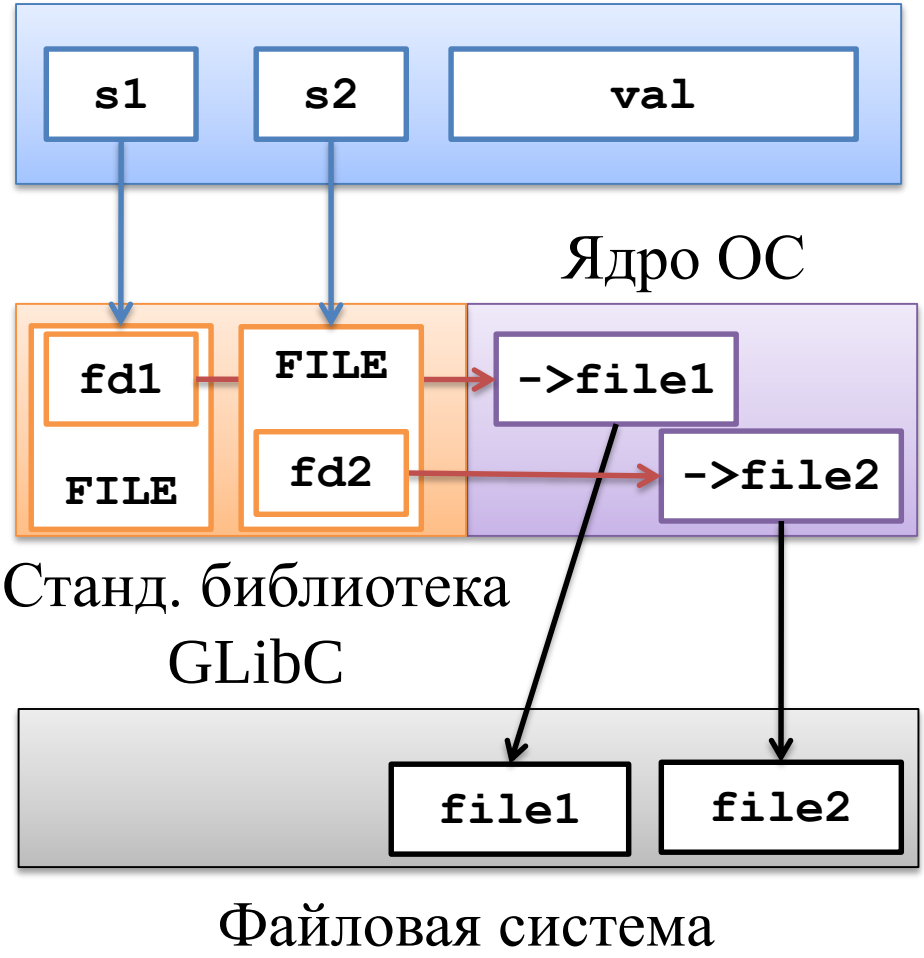
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

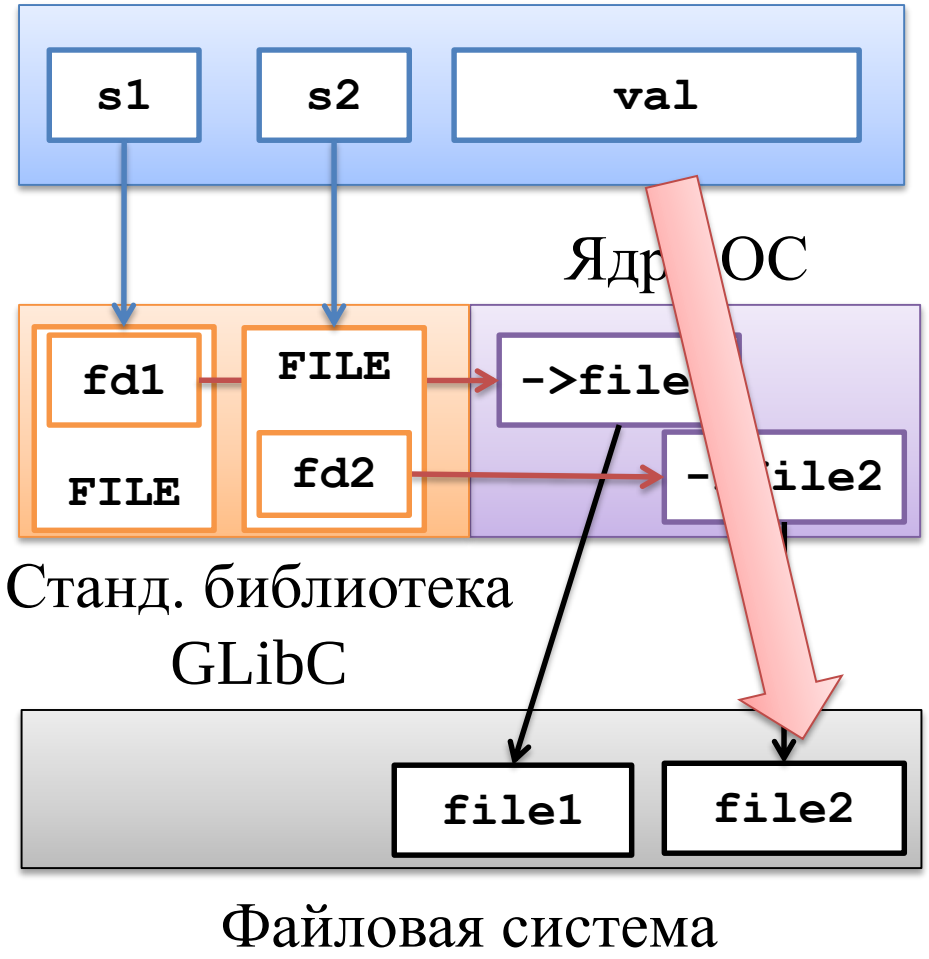
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

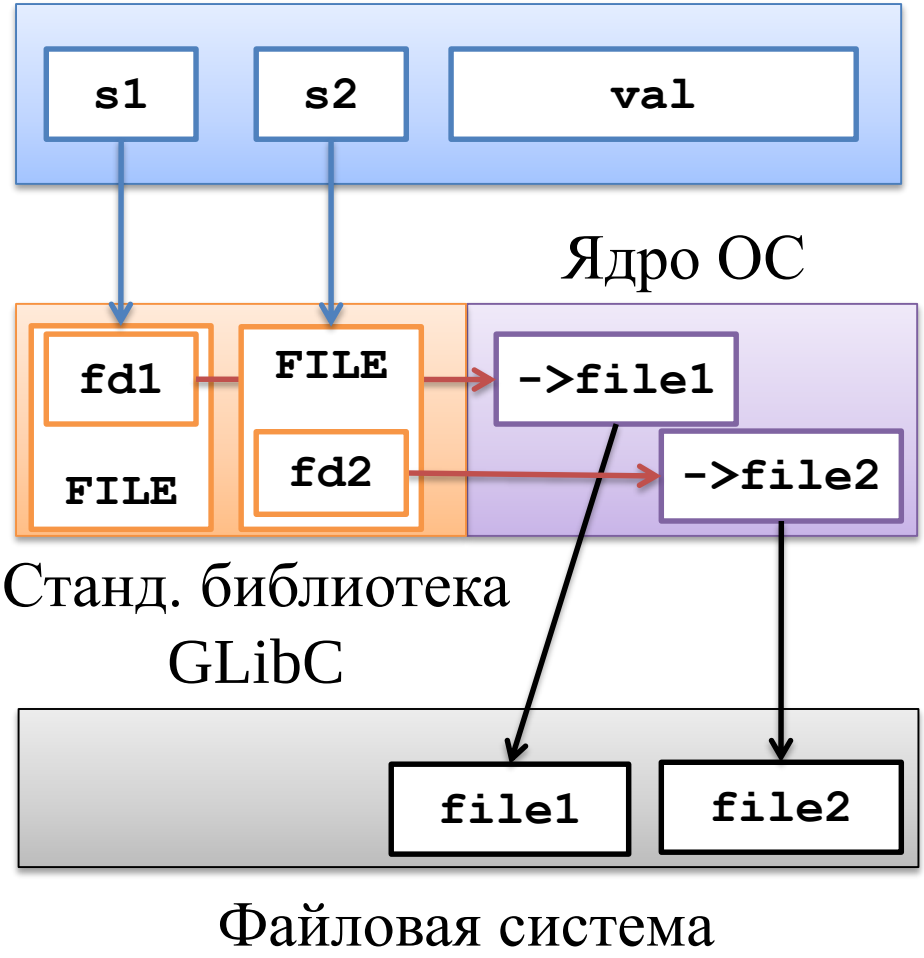
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

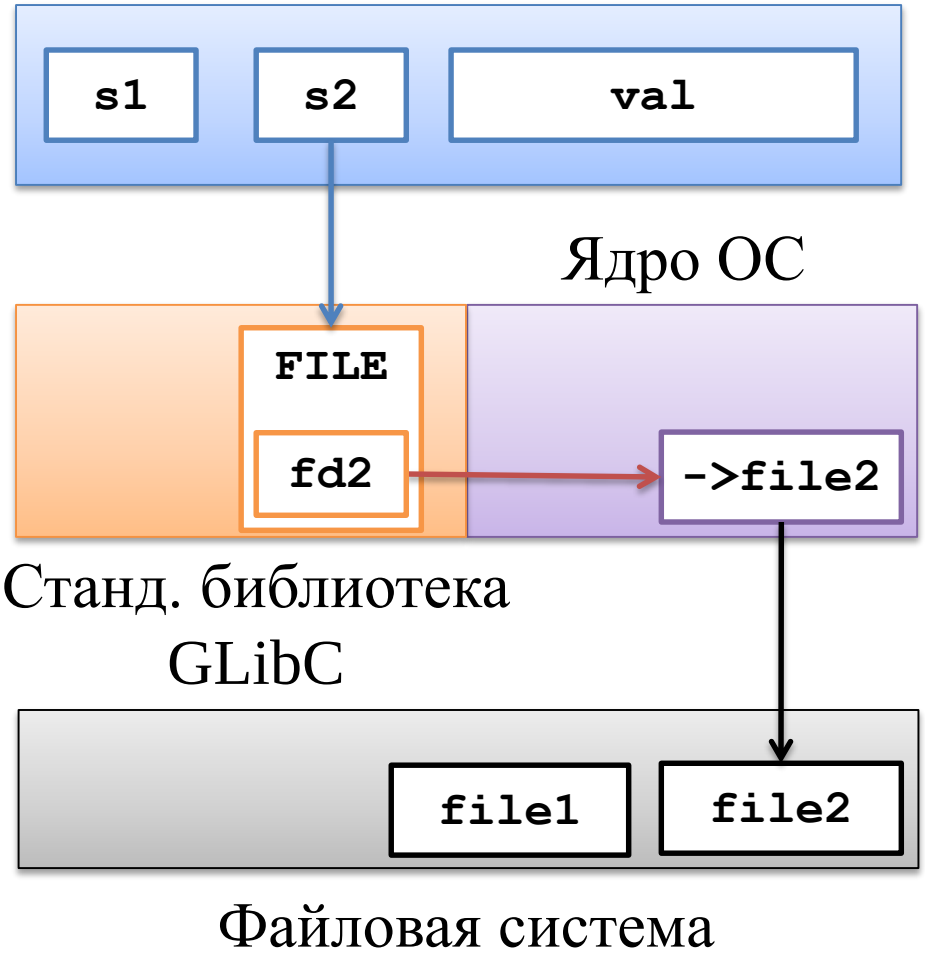
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
              ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Пример использования высокоуровневого ввода-вывода

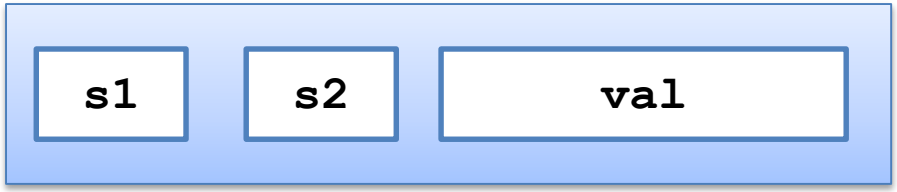
Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
              ...
s1 = fopen("file1","r");
fscanf(s1,"%d", &val);

s2=fopen("file2","w");
fprintf(s2,"%d",val);

fclose(fd1);
fclose(fd2);
```

Программа



Ядро ОС

Станд. библиотека GLibC

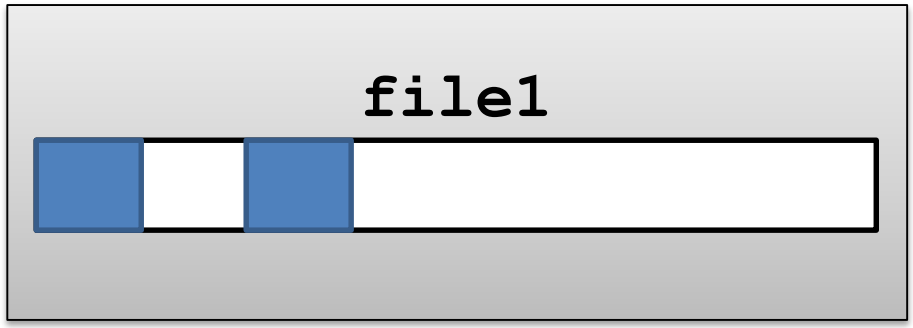
Файловая система

Пример использования высокоуровневого ввода-вывода

Код программы

```
FILE *s1, *s2;
int val;
char buf[256];
                ...
s1 = fopen("file1","r");
fscanf(s1,"%d",&val);
fseek(s1,4, SEEK_CUR);
fscanf(s1,"%d",&val);

fclose(s1);
```



Файловая система

Литература

1. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. – М.: Мир, - 360 с., ил.