Московский Физико-Технический Институт
Физтех-школа Аэрокосмических технологий
УНИЦ Аэромеханики и Летательной Техники

МФТИ

ФАКТ

Архитектура Компьютера и Операционные Системы. Часть 2. Основы операционных систем

Лекция 4: Потоки ввода-вывода, pipe

Новиков Андрей Валерьевич
д.ф.-м.н.

Жуковский

Потоки ввода-вывода

- Передача данных по каналам связи
  - потоковая модель,
  - модель сообщений

- Потоковая модель передачи данных
  - операции приёма/передачи (ввода/вывода) не интересуются содержимым;
  - данные считаются непрерывным потоком байт без внутренней структуры

- Потоковая передача
  - между процессами;
  - между процессом и диском (файлом).

C-функции для работы с файлами

```
#include <stdio.h>


FILE* fopen(const char* filename, const char* mode);
int fclose(FILE* stream);


char* fgets(char* str, int num, FILE* stream);
int fscanf(FILE* stream, const char* format,...);
int fprintf(FILE* stream, const char* format, ... );


size_t fread(void* data, size_t item_size, size_t count, FILE* stream);
size_t fwrite(void* data, size_t item_size, size_t count, FILE* stream);
```

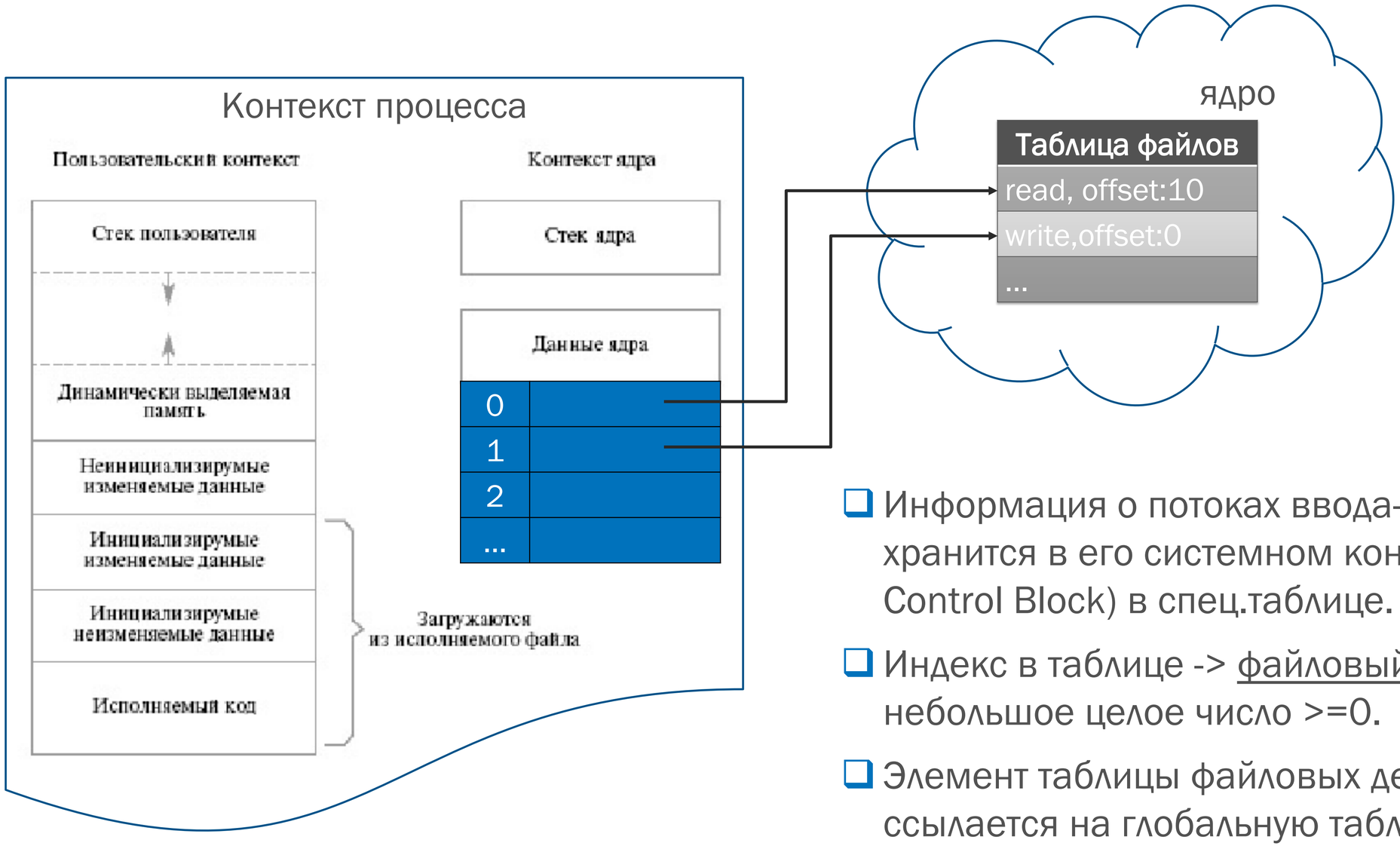
C-функции для работы с файлами

```
#include <stdio.h>
FILE* stream;
fopen(…), fclose(…);
fgets(…), fscanf(…), fprintf(…);
fread(…), fwrite(…);
```

```
#include <fcntl.h>
int file_descriptor;
open(…), close(…);
read(…), write(…);
```

- Надстройка (сервисный интерфейс) над системными вызовами
- C-функции оперируют объектом FILE
- Системные вызовы оперируют с файловыми дескрипторами

Файловый дескриптор



- Информация о потоках ввода-вывода процесса хранится в его системном контексте (PCB = Process Control Block) в спец.таблице.
- Индекс в таблице -> файловый дескриптор – небольшое целое число >=0.
- Элемент таблицы файловых дескрипторов процесса ссылается на глобальную таблицу открытых каналов ввода-вывода.

Файловый дескриптор

- Файловый дескриптор — небольшое целое неотрицательное число, которое для текущего процесса, в данный момент времени однозначно определяет действующий канал ввода-вывода.

- Файловые дескрипторы, создаваемые по-умолчанию
  - 0 (STDIN_FILENO) – стандартный поток ввода (клавиатура);
  - 1 (STDOUT_FILENO) – стандартный поток вывода (экран);
  - 2 (STDERR_FILENO) – стандартный поток ошибок (экран).

Открытие потока (файла)

```
#include <fcntl.h>


int open(const char* path, int
flags, int mode);
```

- Возвращаемое значение

  >=0 – файловый дескриптор

  -1 – ошибка

- Параметр **path** – путь (полное имя) файла

- Параметр **flags**

  - O_RDONLY – только чтение
  - O_WRONLY – только запись
  - O_RDWR – чтение/запись

  комбинируются с помощью | «битовое или»

  - O_CREAT – создать если не существует
  - O_EXCL – вместе с предыдущим, создавать эксклюзивно, ошибка если сущ.
  - O_APPEND – при каждой записи писать в конец
  - O_TRUNC – при открытии удалить содержимое

Открытие потока (файла)

```
#include <fcntl.h>


int open(const char* path, int
flags, int mode);
```

- … flags
  - O_NDELAY – применяется к специальным каналам связи, не ожидать открытия и завершения последующих операций
  - O_SYNC – все последующие операции записи будут ожидать физической записи

- mode – права доступа при создании файла
  - 0400 – только чтение для владельца
  - 0200 – только запись для владельца
  - 0100 – только исполнение для владельца
  - 0040 – только чтение для группы
  - …

Открытие файла. Пример

```
#include <fcntl.h>


int main(int argc, char* argv[], char* envp[])
{
   int fd = open(“test.txt”, O_WRONLY|O_CREAT|O_EXCL, 0664);
   if( fd == -1 ){  perror(“open”); return -1; }
   // write(fd,…);
   return 0;
}
```

Запись/чтение из потока

```
#include <sys/types.h>
#include <unistd.h>

size_t read(int fd, void* ptr,
size_t nbytes);

size_t write(int fd, void* ptr,
size_t nbytes);
```

- Возвращаемое значение
  >=0 – количество считанных/записанных байт (может быть меньше ожидаемого)
  -1 – ошибка
- **fd** – файловый дескриптор
- **ptr** – указатель в памяти начиная с которого размещается информация при чтении или берётся информация при записи
- **nbytes** – количество байт, которые нужно считать/записать

Чтение из потока. Пример

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
int main(int argc, char* argv[], char* envp[])
{
    int fd = open(“test.txt”, O_RDONLY);
    if( fd == -1 ){  perror(“open”); return -1; }

    const int len = 128;
    char data[len];
    if( read(fd, data, len) < len ) ; //
    return 0;
}
```

Закрытие потока

```
#include <unistd.h>


int close(int fd);
```

- Возвращаемое значение
  - 0 – успешное завершение
  - -1 – ошибка
- Параметр **fd** – файловый дескриптор

Потоковая передача через «безымянный канал» pipe

- pipe — «труба», «канал», «конвейер» -- потоковая передача между **родственными** процессами.
- При чтении информации из «трубы» она удаляется(!).
- **pipe** – некая область памяти в ядре в виде кольцевого буфера, куда можно временно поместить данные и затем считать в порядке FIFO (first-in-first-out).

Создание «безымянного канала»

```
#include <unistd.h>


int pipe(int fd[2]);
```

- Создаёт «pipe» внутри ядра. Размер предопределён и не регулируется пользователем.
- Параметр fd[2] – массив из 2х значений, заполняемый при успешном вызове
  - **fd[0]** – файловый дескриптор для **чтения** (**выход** трубы)
  - **fd[1]** – файловый дескриптор для **записи** (**вход** трубы)
- Возвращает
  - 0 – успешное создание
  - -1 – ошибка

pipe. Пример

```
#include <unistd.h>
int main(int argc, char* argv[], char* envp[])
{
    int fd[2];
    if( pipe(fd) == -1 ){  perror(“pipe”); return -1; }
    char out[] = “Hello world”;
    write(fd[1], out, sizeof(out));

    char in[128];
    read(fd[0], in, sizeof(in));
    //
    return 0;
}
```

Особенности read() из pipe

Ситуация	Поведение
Попытка прочитать меньше байт, чем есть в наличии в канале связи.	Читает требуемое количество байт и возвращает значение, соответствующее прочитанному количеству. Прочитанная информация удаляется из канала связи.
В канале связи находится меньше байт, чем затребовано, но не нулевое количество.	Читает все, что есть в канале связи, и возвращает значение, соответствующее прочитанному количеству. Прочитанная информация удаляется из канала связи.
Попытка читать из канала связи, в котором нет информации. Блокировка вызова разрешена.	Вызов блокируется до тех пор, пока не появится информация в канале связи и пока существует процесс, который может передать в него информацию. Если информация появилась, то процесс разблокируется, и поведение вызова определяется двумя предыдущими строками таблицы. Если в канал некому передать данные (нет ни одного процесса, у которого этот канал связи открыт для записи), то вызов возвращает значение 0. Если канал связи полностью закрывается для записи во время блокировки читающего процесса, то процесс разблокируется, и системный вызов возвращает значение 0.
Попытка читать из канала связи, в котором нет информации. Блокировка вызова не разрешена.	Если есть процессы, у которых канал связи открыт для записи, системный вызов возвращает значение -1 и устанавливает переменную errno в значение EAGAIN. Если таких процессов нет, системный вызов возвращает значение 0.

Особенности write() из pipe

Ситуация	Поведение
Попытка записать в канал связи меньше байт, чем осталось до его заполнения.	Требуемое количество байт помещается в канал связи, возвращается записанное количество байт.
Попытка записать в канал связи больше байт, чем осталось до его заполнения. Блокировка вызова разрешена.	Вызов блокируется до тех пор, пока все данные не будут помещены в канал связи. Если размер буфера канала связи меньше, чем передаваемое количество информации, то вызов тем самым будет ждать, пока часть информации не будет считана из канала связи. Возвращается записанное количество байт.
Попытка записать в канал связи больше байт, чем осталось до его заполнения, но меньше, чем размер буфера канала связи. Блокировка вызова запрещена.	Системный вызов возвращает значение -1 и устанавливает переменную errno в значение EAGAIN.
В канале связи есть место. Попытка записать в канал связи больше байт, чем осталось до его заполнения, и больше, чем размер буфера канала связи. Блокировка вызова запрещена.	Записывается столько байт, сколько осталось до заполнения канала. Системный вызов возвращает количество записанных байт.
Попытка записи в канал связи, в котором нет места. Блокировка вызова не разрешена.	Системный вызов возвращает значение -1 и устанавливает переменную errno в значение EAGAIN.
Попытка записи в канал связи, из которого некому больше читать, или полное закрытие канала на чтение во время блокировки системного вызова.	Если вызов был заблокирован, то он разблокируется. Процесс получает сигнал SIGPIPE. Если этот сигнал обрабатывается пользователем, то системный вызов вернет значение -1 и установит переменную errno в значение EPIPE.

Потоковая передача через «именованный канал» FIFO

- FIFO или named pipe — именованная «труба», «канал», «конвейер» -- потоковая передача между **любыми** процессами.
- При чтении информации из «трубы» она удаляется(!).
- **FIFO** – некая область памяти в ядре в виде кольцевого буфера, куда можно временно поместить данные и затем считать в порядке FIFO (first-in-first-out).
- FIFO даётся имя путём создания «**метки**» в файловой системе.

Создание именованного канала

```
$ mkfifo mylabel.fifo
```

```
#include <fcntl.h>
int main(int argc, char* argv[], char* envp[])
{
    int fd = open(“mylabel.fifo”, O_RDONLY);
    const int len = 128;
    char data[len];
    if( read(fd, data, len) < len ) ; //
    return 0;
}
```