ЛЕКЦИЯ 8

Кооперация процессов
и основные аспекты её логической организации

Основные причины для объединения усилий процессов

- Повышение скорости решения задач
- Совместное использование данных
- Модульная конструкция какой-либо системы
- Для удобства работы пользователя

Кооперативные или взаимодействующие процессы
это процессы, которые влияют на поведение друг друга
путем обмена информацией

Категории средств обмена информацией

- Сигнальные
- Канальные
- Разделяемая память

Основные аспекты логической организации передачи информации

Как устанавливается связь

Нужна или не нужна инициализация?

Способы адресации:

- прямая адресация
  - симметричная
  - асимметричная
- непрямая или косвенная адресация

Информационная валентность процессов и средств связи

- Сколько процессов может быть ассоциировано с конкретным средством связи.
- Сколько идентичных средств связи может быть задействовано между двумя процессами
- Направленность связи:
  - симплексная
  - полудуплексная
  - дуплексная

Особенности канальных средств связи

Буферизация

Буфера нет (нулевая емкость)

процесс-передатчик всегда обязан ждать приема

Буфер конечной емкости

процесс-передатчик обязан ждать освобождения места в буфере, если буфер заполнен

Буфер неограниченной емкости (нереализуемо!)

процесс-передатчик никогда не ждет

Модели передачи данных

- Потоковая модель

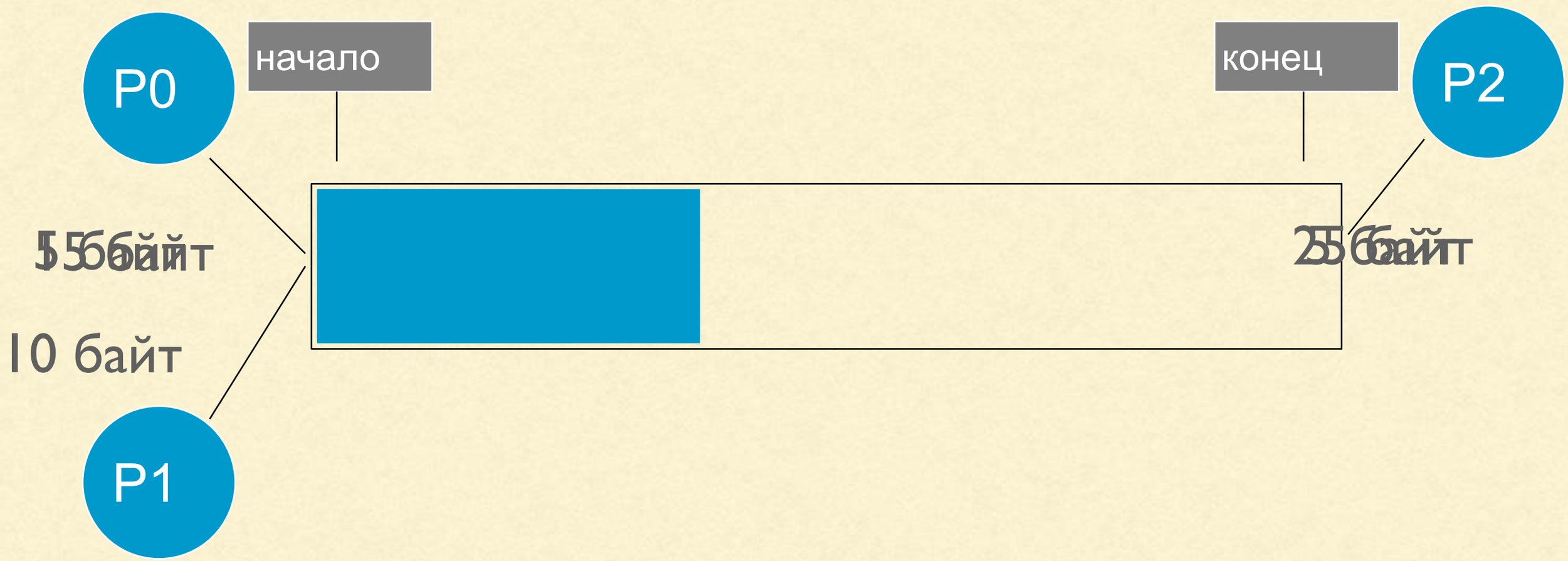
  операции приема/передачи не интересуются содержимым данных и их происхождением, данные не структурируются

- Модель сообщений

  на передаваемые данные накладывается определенная структура

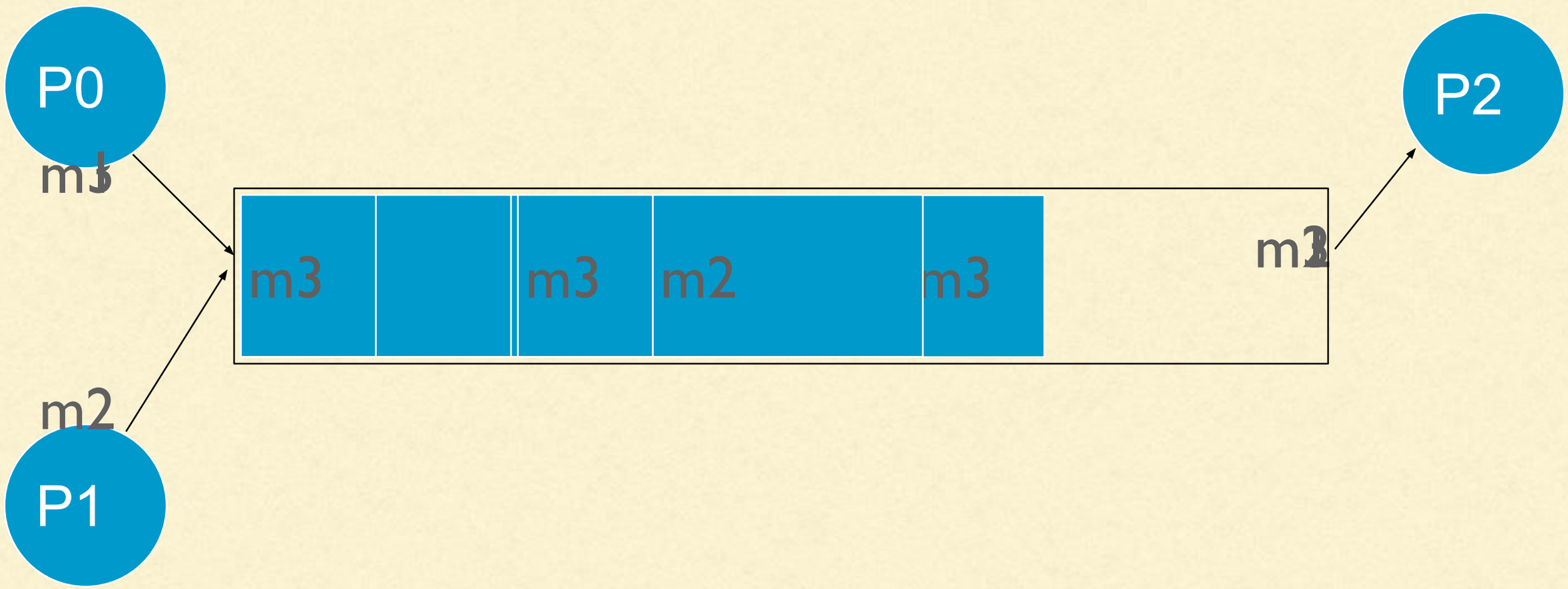
Особенности канальных средств связи

Потоковая модель - FIFO



Особенности канальных средств связи

Модель сообщений



Надежность средств связи

Средство связи считается надежным, если:

- Нет потери информации
- Нет повреждения информации
- Нет нарушения порядка поступления информации
- Не появляется лишняя информация

Как завершается связь

- Нужны ли специальные действия для прекращения использования средства связи?
- Как влияет прекращение использования средства связи одним процессом на поведение других участников взаимодействия?

Примеры из Linux

Сигналы(исторически, ключевой сигнал - приостановка процесса)

Для остановки процесса используется команда kill, она же позволяет посылать любые другие сигналы.

```
Alexandrs-MBP:~ alexsharamet$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
 9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2
Alexandrs-MBP:~ alexsharamet$
```

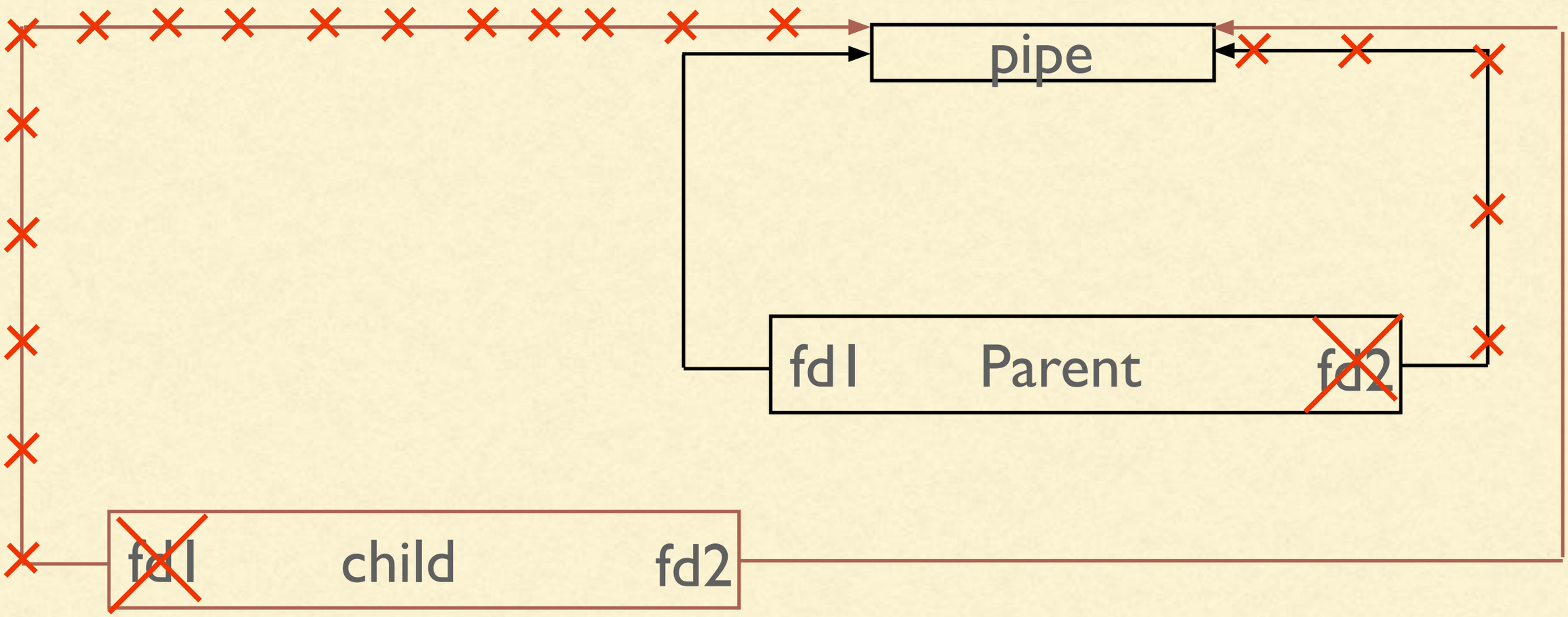
```c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

void mysignal_handler(int signalno)
{
    printf("Called with %d\n",signalno);
}

int main(){
    int c;
    signal(SIGUSR1,mysignal_handler);
    while(1){
        printf("hello %d\n",c++);
        usleep(500000);
    }
    return 0;
}
```

В одной консоли работает программа в бесконечном цикле, во второй процессу подаётся сигнал.

```
hello 420
Called with 10
hello 421
hello 422
Called with 10
hello 423
hello 424
hello 425
hello 426
hello 427
hello 428
```

```
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$ kill -s SIGUSR1  16254
[asharamet@mgmt stepic]$
```

Канальные средства связи:
Создание неименованных каналов(только в рамках parent-child).



```c
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]){
    int pipefd[2];
    pipe(pipefd);
    pid_t cpid = fork();
    int buf[100],i;

    if (cpid) {//parent
        close(pipefd[0]);          /* Close unused read end */
        for(i=0;i<100;i++)buf[i]=i;
        write(pipefd[1], buf, 100*sizeof(int));
        close(pipefd[1]);          /* Reader will see EOF */
        wait(NULL);                /* Wait for child */
        exit(0);
    } else {//child
        close(pipefd[1]);          /* Close unused write end */
        while (read(pipefd[0], buf, 100*sizeof(int)) > 0)
        for(i=0;i<100;i++)printf("%d ",buf[i]);
        printf("\n");
        close(pipefd[0]);
        exit(0);
    }
}
```

Именованные каналы

```c
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>

int main(){
    int fd;
    char * myfifo = "/tmp/myfifo";

    /* create the FIFO (named pipe) */
    mkfifo(myfifo, 0666);

    /* write "Hi" to the FIFO */
    fd = open(myfifo, O_WRONLY);
    write(fd, "Hi", sizeof("Hi"));
    close(fd);

    /* remove the FIFO */
    unlink(myfifo);

    return 0;
}
```

```c
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

#define MAX_BUF 1024

int main(){
    int fd;
    char * myfifo = "/tmp/myfifo";
    char buf[MAX_BUF];

    /* open, read, and display the message from the FIFO */
    fd = open(myfifo, O_RDONLY);
    read(fd, buf, MAX_BUF);
    printf("Received: %s\n", buf);
    close(fd);

    return 0;
}
```

```
[asharamet@mgmt stepic]$ ./writer
```

```
[asharamet@mgmt stepic]$ ./reader
Received: Hi
```