

---

# 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
-

---

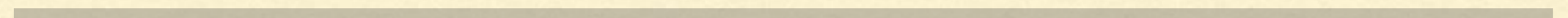
# Основные аспекты логической организации передачи информации

## 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

- Поточковая модель

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

- Модель сообщений

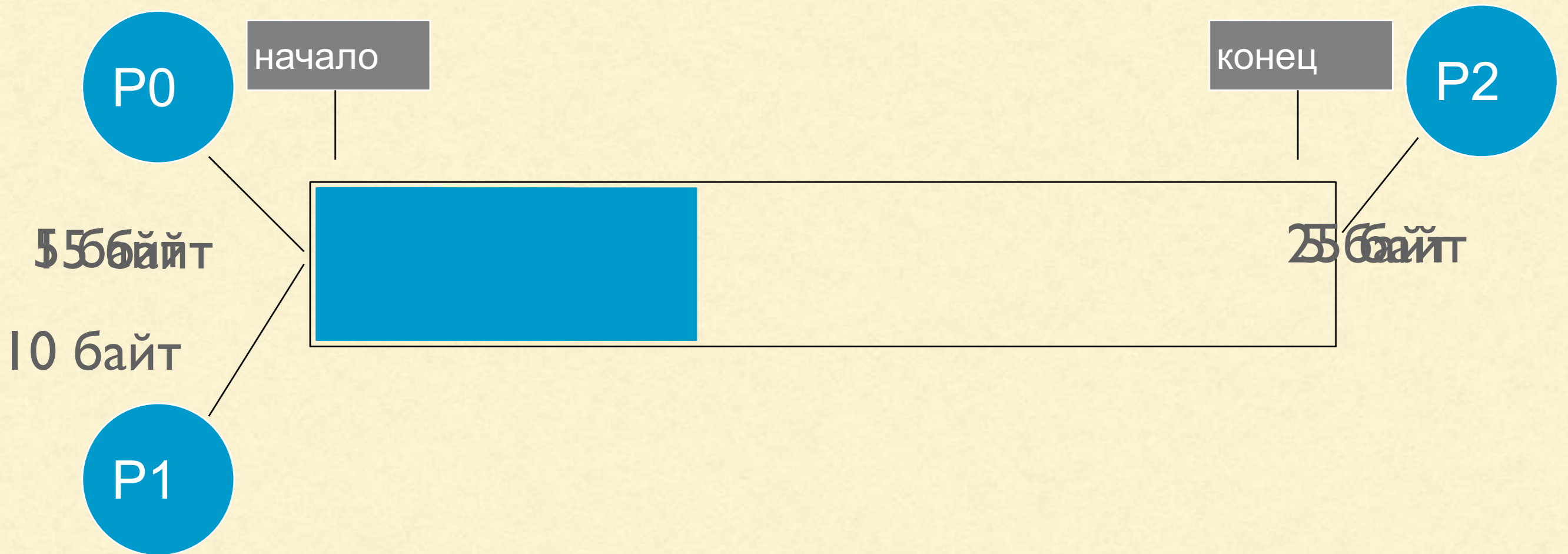
на передаваемые данные накладывается определенная

структура

---

# Особенности канальных средств связи

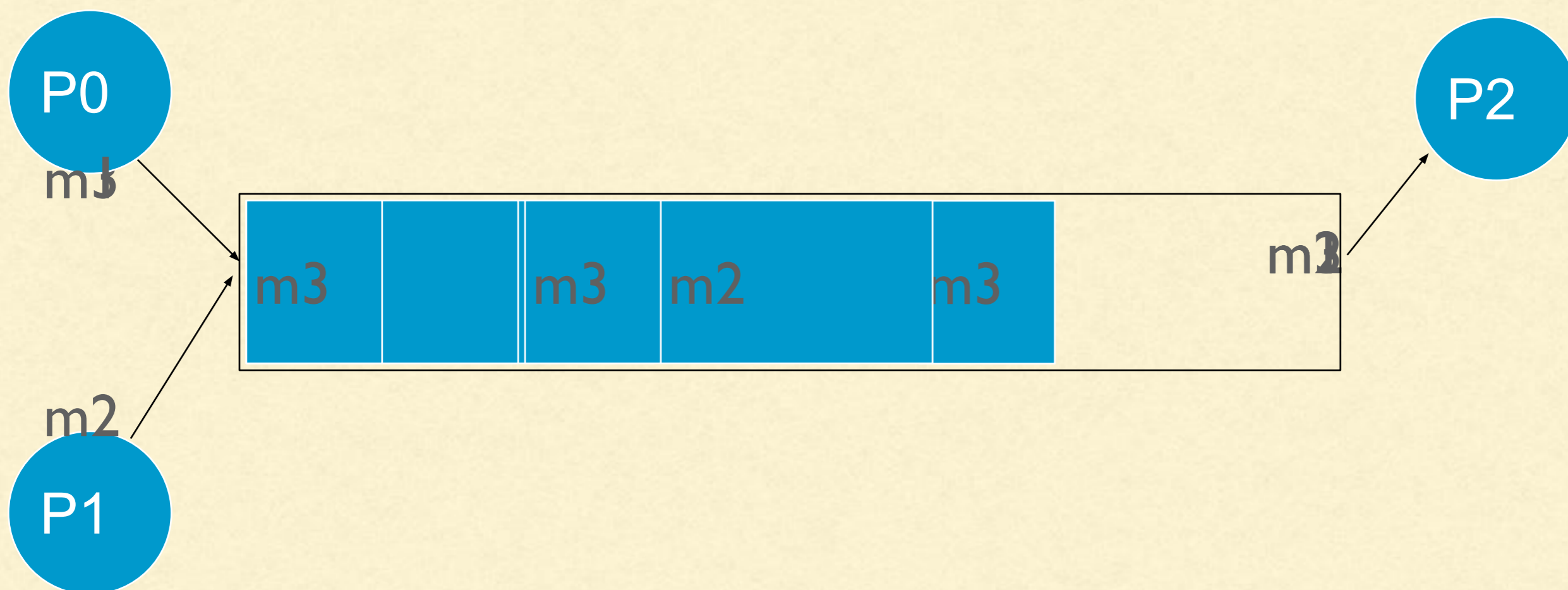
## Потоковая модель - FIFO



---

# 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$
```

---

```
1 #include <signal.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 void mysignal_handler(int signalno)
6 {
7     printf("Called with %d\n",signalno);
8 }
9
10 int main(){
11     int c;
12     signal(SIGUSR1,mysignal_handler);
13     while(1){
14         printf("hello %d\n",c++);
15         usleep(500000);
16     }
17     return 0;
18 }
```

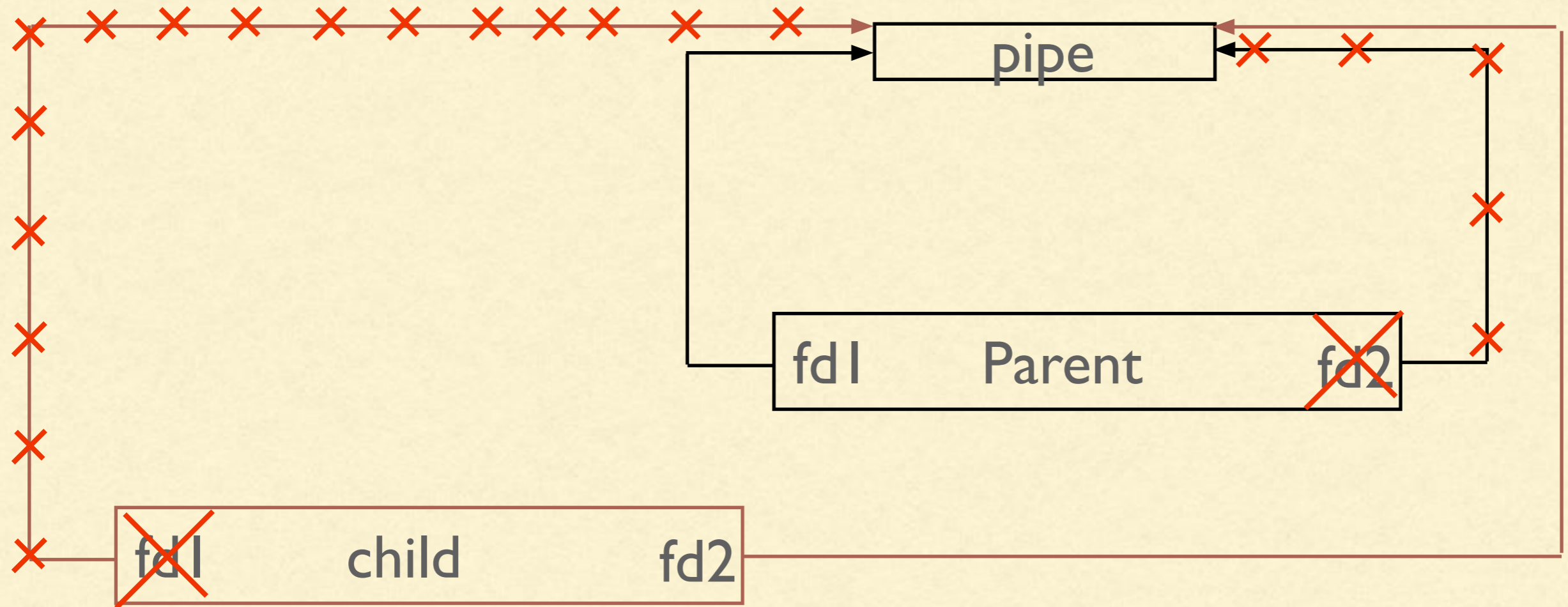
---

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

```
hello 420 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
Called with 10 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 421 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 422 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
Called with 10 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 423 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 424 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 425 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 426 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 427 [asharamet@mgmt stepic]$ kill -s SIGUSR1 16254
hello 428 [asharamet@mgmt stepic]$ █
```

# Канальные средства связи:

Создание неименованных каналов (только в рамках parent-child).



```
1 #include <sys/wait.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6 int main(int argc, char *argv[]){
7     int pipefd[2];
8     pipe(pipefd);
9     pid_t cpid = fork();
10    int buf[100],i;
11
12    if (cpid) { //parent
13        close(pipefd[0]);          /* Close unused read end */
14        for(i=0;i<100;i++)buf[i]=i;
15        write(pipefd[1], buf, 100*sizeof(int));
16        close(pipefd[1]);          /* Reader will see EOF */
17        wait(NULL);                /* Wait for child */
18        exit(0);
19    } else { //child
20        close(pipefd[1]);          /* Close unused write end */
21        while (read(pipefd[0], buf, 100*sizeof(int)) > 0)
22            for(i=0;i<100;i++)printf("%d ",buf[i]);
23        printf("\n");
24        close(pipefd[0]);
25        exit(0);
26    }
27 }
```

# Именованные

WRITER

```
1 #include <fcntl.h>
2 #include <sys/stat.h>
3 #include <unistd.h>
4
5 int main(){
6     int fd;
7     char * myfifo = "/tmp/myfifo";
8
9     /* create the FIFO (named pipe) */
10    mkfifo(myfifo, 0666);
11
12    /* write "Hi" to the FIFO */
13    fd = open(myfifo, O_WRONLY);
14    write(fd, "Hi", sizeof("Hi"));
15    close(fd);
16
17    /* remove the FIFO */
18    unlink(myfifo);
19
20    return 0;
21 }
```

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

```
1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <unistd.h>
4
5 #define MAX_BUF 1024
6
7 int main(){
8     int fd;
9     char * myfifo = "/tmp/myfifo";
10    char buf[MAX_BUF];
11
12    /* open, read, and display the message from the FIFO */
13    fd = open(myfifo, O_RDONLY);
14    read(fd, buf, MAX_BUF);
15    printf("Received: %s\n", buf);
16    close(fd);
17
18    return 0;
19 }
```

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