Межпроцессное взаимодействие

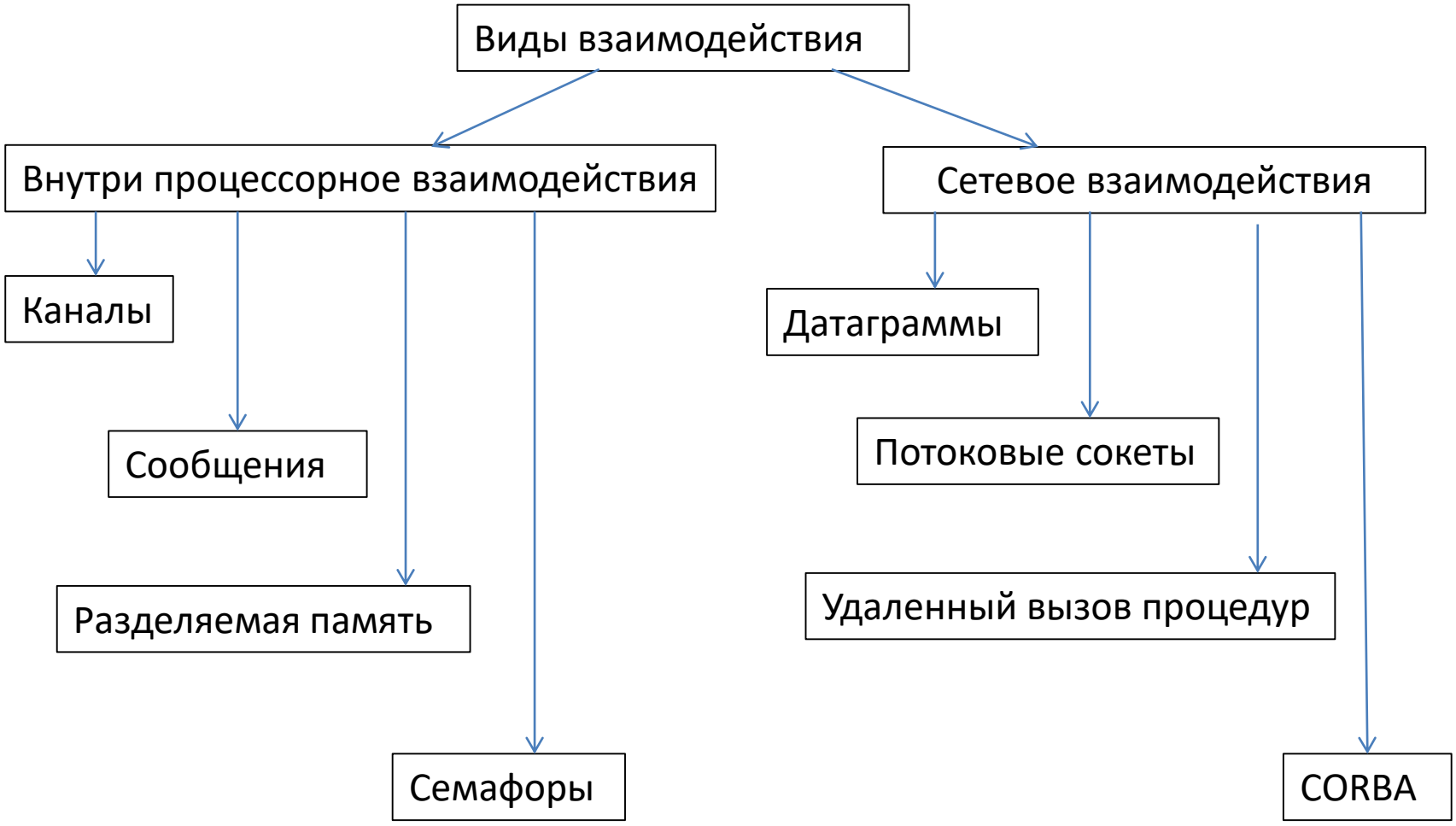


Схема передачи через канал

Данные обрабатываются по алгоритму FIFO "первым пришел, первым обслужен".

Типы каналов в Linux

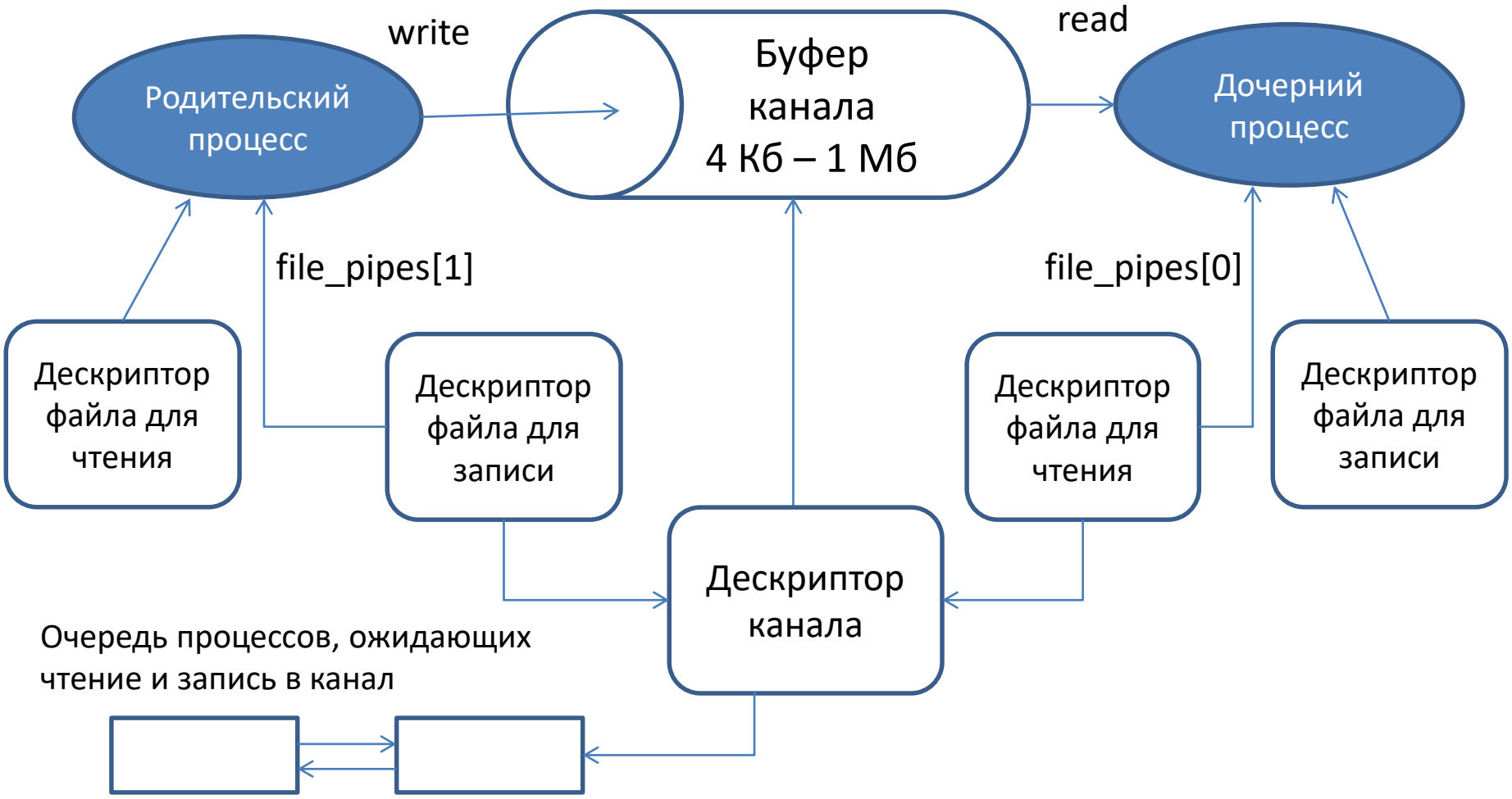
- Безымянный канал, также называемый анонимным, неименованным.

Неименованный канал является средством взаимодействия между процессами, связанными родственными отношениями.

- Именованные каналы

Процессы открывают поименованные каналы так же, как и обычные файлы, и, следовательно, с помощью поименованных каналов могут взаимодействовать между собой даже процессы, не имеющие друг к другу близкого отношения.

Неименованные каналы



Операции с каналом

Открытие канала

int pipe(int pipe_fd[2]);

Запись в канал

int write(int pipe_fd[1], void *area, int cnt);

Чтение из канала

int read(int pipe_fd[0], void *area, int cnt);

Закрытие канала

int close(int pipe_fd);

Запрет блокировки чтения или записи

fcntl(pipe_fd, F_SETFL, O_NONBLOCK);

Алгоритм открытия канала

алгоритм pipe
входная информация: отсутствует
выходная информация: дескриптор файла для чтения и дескриптор файла для записи
{
назначить новый индекс новому файлу (права собственности, права доступа, размер файла и расположение данных файла в файловой системе);
выделить одну запись в таблице файлов для чтения, одну - для записи;
инициализировать записи в таблице файлов таким образом,
чтобы они указывали на новый индекс;
выделить один пользовательский дескриптор файла для чтения, один - для записи, проинициализировать их таким
образом, чтобы они указывали на соответствующие точки
входа в таблице файлов;
установить значение счетчика ссылок в индексе равным 2;
установить значение счетчика числа процессов, производя-
щих чтение, и процессов, производящих запись, равным 1;
}

Алгоритм записи в канал

1. Запись в канал разрешена, если канал открыт для чтения. Если процесс пробует записать в канал, не имеющий читателя, то будет послан сигнал SIGPIPE.
2. Запись в канал осуществляется, если имеется достаточно места для всех данных и их объем не превышает размер буфера.
3. Если места нет, то процесс приостанавливается и помещается в очередь к каналу (не установлен флаг O_NONBLOCK).
4. Если записываются в канал данные, объем которых превышает емкость буфера, то в канал записываются столько данных, сколько он может вместить в себя, и процесс помещается в очередь до тех пор, пока не освободится дополнительное место (возможно перемешивание данных).

Алгоритм чтения данных из канала

1. Если канал не открыт для записи (отсутствуют писатели), возвращается 0. Если канал используется для записи в родительском и в дочернем процессах, то необходимо закрыть файловые дескрипторы записи в канал в обоих этих процессах, прежде чем канал будет считаться закрытым.
2. Если процесс пытается считать больше данных, чем фактически есть в канале, то будут прочитаны все данные, находящиеся в данный момент в канале.
3. Если канал пуст, то процесс приостанавливается и помещается в очередь к каналу (не установлен флаг O_NONBLOCK, а при установленном флаге будет возвращено -1).
4. На время выполнения операции доступ другим процессам к каналу запрещен, после завершения операции все процессы из очереди активизируются.

Алгоритм закрытия канала

1. Уменьшается на 1 количество процессов чтения из канала или записи в канал в зависимости от типа закрываемого файлового дескриптора.
2. Если значение счетчика числа записывающих в канал процессов становится равным 0 и имеются процессы, приостановленные в ожидании чтения данных из канала, ядро возобновляет выполнение последних и они завершают свои операции чтения без возврата каких-либо данных.
3. Если становится равным 0 значение счетчика числа считывающих из канала процессов и имеются процессы, приостановленные в ожидании возможности записи данных в канал, ядро возобновляет выполнение последних и посылает им сигнал SIGPIPE.
4. Если к каналу не обращается ни один записывающий или считывающий процесс, ядро освобождает все информационные блоки канала и переустанавливает индекс таким образом, чтобы он указывал на то, что канал пуст.

Пример обмена данными через канал

```
/* ПРОГРАММА ПРОЦЕССА ПРЕДКА*/
#include <stdio.h>
main(int argc, char** argv)
{   int fildes[2]; /* дескрипторы файлов канала: fildes[0] - для  чтения из канала и fildes[1] - для записи в
       канал*/
  char ch;
  FILE *fp;
  int stat;
  pipe(fildes); /* открытие канала*/
  fp=fopen(argv[1],"r"); /* открытие текстового файла */
  if (!fork())  /*порождение потомка и вызов его программы */
     execl("p1","p1",&fildes[0], &fildes[1], NULL); /* в программу потомка передается
                        номер дескриптора файла для чтения из канала */
    else
    {close(fildes[0]); /* закрытие предком канала на чтение*/
       while(!feof(fp)) {  /* цикл записи символов в канал, пока в него не  будет записан  символ конца файла*/
          ch=fgetc(fp);
          write(fildes[1],&ch,1);
       }
       fclose(fp); /* закрытие текстового файла*/
       wait(&stat); /* ожидание завершения потомка*/
       close(fildes[1]); /* закрытие предком канала на запись*/
    }  }
```

Пример обмена данными через канал

```
/*ПРОГРАММА ПРОЦЕССА ПОТОМКА*/
#include <stdio.h>
main(int argc, char** argv)
{
  char ch;
  int c;
  int fildes[2];
  fildes[0]=*argv[1]; /* номер дескриптора файла для чтения из канала*/
  fildes[1]=*argv[2]; /* номер дескриптора файла для записи в канал*/
  close(fildes[1]); /* закрытие потомком канала на запись*/
   do {     /*  цикл чтения символов из канала*/
    read(fildes[0], &ch, 1);
    printf("%c",ch);
   }
   while((c=ch) !=EOF ); /* конец цикла при чтении символа конца файла*/
   close(fildes[0]); /* закрытие потомком канала на чтение*/
   exit(0);
 }
```

Задание 1 к лабораторной работе 7

Написать программу, которая в качестве параметров принимает имена 3 текстовых файла (два входных и один выходной). Программа должна открыть канал и выходной файл, а затем породить двух потомков, которым предаются дескриптор канала для записи и имя входного файла. **Каждый потомок выполняет свою программу**, читая построчно текст из входного файла и записывая его в канал. Программа параллельно посимвольно читает данные из канала и записывает их в выходной файл, **до тех пор пока оба потомка не закончат свою работу и канал будет пуст.**

Задание 2 к лабораторной работе 7

Разработать программу, которая обменивается данными через канал с двумя потомками. Программа открывает входной файл, построчно читает из него данные и записывает их в канал. **Потомки в своих программах** поочередно читают символы из канала и записывают их в свои выходные файлы, первый потомок нечетные символы, а второй – четные. Синхронизация работы потомков должна осуществляться **напрямую** с использованием сигналов SIGUSR1 и SIGUSR2. Об окончании записи файла в канал программа оповещает потомков сигналом SIGQUIT и ожидает завершение работы потомков. Когда они заканчивают работу, программа закрывает канал.