ОС(UNIX)

(Взаимодействие между процессами: сигналы, каналы)

Взаимодействие между процессами

- в UNIX процессы выполняются в собственном адресном пространстве и по существу изолированы друг от друга.
- сведены к минимуму возможности влияния процессов друг на друга, что является необходимым в многозадачных ОС.

Взаимодействие между процессами

Для реализации взаимодействия требуется:

- -обеспечить средства взаимодействия между процессами и одновременно исключить нежелательное влияние одного процесса на другой.

Взаимодействие между процессами

Взаимодействие между процессами необходимо для решения следующих задач:

- *Передача данных*. Один процесс передает данные другому процессу, при этом их объем может варьироваться от десятков байтов до нескольких мегабайтов.

Взаимодействие между процессами

- *Совместное использование данных*. Процессы могут *совместно использовать одну копию данных*. Количество взаимодействующих процессов м.б. больше двух.

При совместном использовании ресурсов процессам может понадобиться некоторый протокол взаимодействия для сохранения целостности данных и исключения конфликтов при доступе к ним.

Взаимодействие между процессами

- *Извещения.* Процесс может известить другой процесс или группу процессов о наступлении некоторого события (например, для синхронизации выполнения нескольких процессов).

Средства межпроцессного взаимодействия

В ОС UNIX:

- ***сигналы***
- ***каналы***
- ***FIFO (именованные каналы)***
- ***сообщения (очереди сообщений)***
- ***семафоры***
- ***разделяемую память***

Средства межпроцессного взаимодействия

- Последние три типа IPC обычно обобщенно называют System V IPC.
- Есть еще одно средство IPC — сокеты, впервые предложенные в BSD UNIX

Средства межпроцессного взаимодействия: ***Сигналы***

- ***Сигналы*** изначально были предложены как средство уведомления об ошибках, но могут использоваться и для элементарного IPC, например, для синхронизации процессов или для передачи простейших команд от одного процесса к другому

Средства межпроцессного взаимодействия: ***Сигналы***

- Например, для сервера системы имен (DNS) named, используется сигнал **SIGHUP**, по существу являющийся командой обновления базы данных.

Средства межпроцессного взаимодействия: ***Сигналы***

фазы механизма сигналов

- *генерация* (или отправление сигнала);
- *его доставка и обработка.*

Средства межпроцессного взаимодействия: ***Сигналы***

Сигнал *отправляется*, когда происходит определенное событие, о наступлении которого должен быть уведомлен процесс.

Средства межпроцессного взаимодействия: ***Сигналы***

- Сигнал считается *доставленным*, когда процесс, которому был отправлен сигнал, получает его и выполняет его обработку.
- В промежутке между этими двумя моментами сигнал *ожидает* доставки.

Средства межпроцессного взаимодействия: ***Сигналы***

основные причины отправки сигнала:

Особые ситуации	Когда выполнение процесса вызывает особую ситуацию, например, деление на ноль, процесс получает соответствующий сигнал.
Терминальные прерывания	Нажатие некоторых клавиш терминала, например, <De1>, <Ctrl>+<C> или <Ctrl>+<\>, вызывает отправление сигнала текущему процессу, связанному с терминалом.

Средства межпроцессного взаимодействия: ***Сигналы***

основные причины отправки сигнала:

Другие процессы	Процесс может отправить сигнал другому процессу или группе процессов с помощью системного вызова kill(). В этом случае сигналы являются элементарной формой межпроцессного взаимодействия.

Средства межпроцессного взаимодействия: ***Сигналы***

основные причины отправки сигнала:

Управление заданиями	Командные интерпретаторы, поддерживающие систему управления заданиями, используют сигналы для манипулирования фоновыми и текущими задачами. Когда процесс, выполняющийся в фоновом режиме, делает попытку чтения или записи на терминал, ему отправляется сигнал останова. Когда дочерний процесс завершает свою работу, родитель уведомляется об этом также с помощью сигнала.

Средства межпроцессного взаимодействия: ***Сигналы***

основные причины отправки сигнала:

Квоты	Когда процесс превышает выделенную ему квоту вычислительных ресурсов или ресурсов ФС, ему отправляется соответствующий сигнал.

Средства межпроцессного взаимодействия: ***Сигналы***

основные причины отправки сигнала:

Уведомления	Процесс может запросить уведомление о наступлении тех или иных событий, например, готовности устройства и т. д. Такое уведомление отправляется процессу в виде сигнала.
Алармы	Если процесс установил таймер, ему будет отправлен сигнал, когда значение таймера станет равным нулю.

Средства межпроцессного взаимодействия: ***Сигналы***

Доставка и обработка сигнала

Для каждого сигнала в системе определена обработка по умолчанию, которую выполняет ядро, если процесс не указал другого действия.

Средства межпроцессного взаимодействия: ***Сигналы***

В общем случае - *возможные действия*:

- завершить выполнение процесса,
- игнорировать сигнал,
- остановить процесс
- продолжить процесс (справедливо для остановленного процесса, для остальных сигнал игнорируется),

наиболее употребительным из которых является первое.

Средства межпроцессного взаимодействия: ***Сигналы***

Процесс *может*

- изменить действие по умолчанию,
- либо зарегистрировать собственный обработчик сигнала,
- либо указать, что сигнал следует игнорировать.
- заблокировать сигнал, отложив на некоторое время его обработку.

Средства межпроцессного взаимодействия: ***Сигналы***

Доставка сигнала происходит после того, как ядро от имени процесса вызывает системную процедуру ***issig()***, которая проверяет, существуют ли ожидающие доставки сигналы, адресованные данному процессу.

Средства межпроцессного взаимодействия: ***Сигналы***

Функция ***issig()*** вызывается ядром в случаях:

- Непосредственно перед возвращением из режима ядра в режим задачи после обработки системного вызова или прерывания.
- Непосредственно перед переходом процесса в состояние сна с приоритетом, допускающим прерывание сигналом.

Средства межпроцессного взаимодействия: ***Сигналы***

- Сразу же после пробуждения после сна с приоритетом, допускающим прерывание сигналом.

Средства межпроцессного взаимодействия: ***Сигналы***

Если процедура ***issig()*** обнаруживает ожидающие доставки сигналы, ядро вызывает функцию доставки сигнала, которая выполняет действия по умолчанию или вызывает специальную функцию ***sendsig()***, запускающую обработчик сигнала, зарегистрированный процессом.

Средства межпроцессного взаимодействия: ***Сигналы***

Количество различных сигналов в современных версиях UNIX около 30, каждый из них имеет уникальное имя и номер.

Описания представлены в файле **<signal.h>.**

Средства межпроцессного взаимодействия: ***Сигналы***

Числ.знач.	Константа	Значение сигнала
2	SIGINT	Прерывание выполнения по нажатию Ctrl-C
3	SIGQUIT	Аварийное завершение работы
9	SIGKILL	Уничтожение процесса
14	SIGALRM	Прерывание от программного таймера
18	SIGCHLD	Завершился процесспотомок

Средства межпроцессного взаимодействия: ***Сигналы***

- Сигналы являются механизмом *асинхронного* взаимодействия, т.е. момент прихода сигнала процессу заранее неизвестен.

Средства межпроцессного взаимодействия: *Сигналы*

Для отправки сигнала существует системный вызов kill():

#include <sys/types.h>

#include <signal.h>

- **int kill (pid_t pid, int sig);**

pid идентификатор процесса, которому посылается сигнал (в частности, процесс может послать сигнал самому себе).

Средства межпроцессного взаимодействия: ***Сигналы***

Если **pid=0**, сигнал будет передан всем процессам, которые принадлежат той же группе, что и процесс, посылающий сигнал, за исключением процессов с идентификаторами 0 и 1.

Средства межпроцессного взаимодействия: ***Сигналы***

sig номер посылаемого сигнала.

Если этот параметр равен 0, то будет выполнена проверка корректности обращения к kill() (в частности, существование процесса с идентификатором pid), но никакой сигнал в действительности посылаться не будет.

Средства межпроцессного взаимодействия: ***Сигналы***

- сигналы очень ресурсоемки.
- Отправка сигнала требует выполнения системного вызова, а его доставка — прерывания процесса-получателя и интенсивных операций со стеком процесса для вызова функции обработки и продолжения его нормального выполнения.
- сигналы слабо информативны и их число весьма ограничено.

Средства межпроцессного взаимодействия: Каналы

при работе в командной строке shell:

- **cat myfile | wc**

При этом (стандартный) вывод программы **cat**(), которая выводит содержимое файла **myfile**, передается на (стандартны) ввод программы **wc**(), которая, в свою очередь подсчитывает количество строк, слов и символов.

В результате мы получим:

- 12 45 260

что будет означать количество строк, слов и символов в файле myfile.

Средства межпроцессного взаимодействия: ***Каналы***

- Т.о., два процесса обменялись данными. При этом использовался **программный канал**, обеспечивающий *однонаправленную* передачу данных между двумя задачами.

Средства межпроцессного взаимодействия: ***Каналы***

Для ***создания канала*** используется системный вызов ***pipe():***

- **int pipe(int *fieldes);**

который возвращает два файловых дескриптора

— **fildes [0]** для записи в канал и

— **fildes[1]** для чтения из канала.

Средства межпроцессного взаимодействия: ***Каналы***

Теперь, если один процесс записывает данные в **fildes[0]**, другой сможет получить эти данные из **fildes[1]** .

Средства межпроцессного взаимодействия: ***Каналы***

Дочерний процесс наследует и разделяет все назначенные файловые дескрипторы родительского.

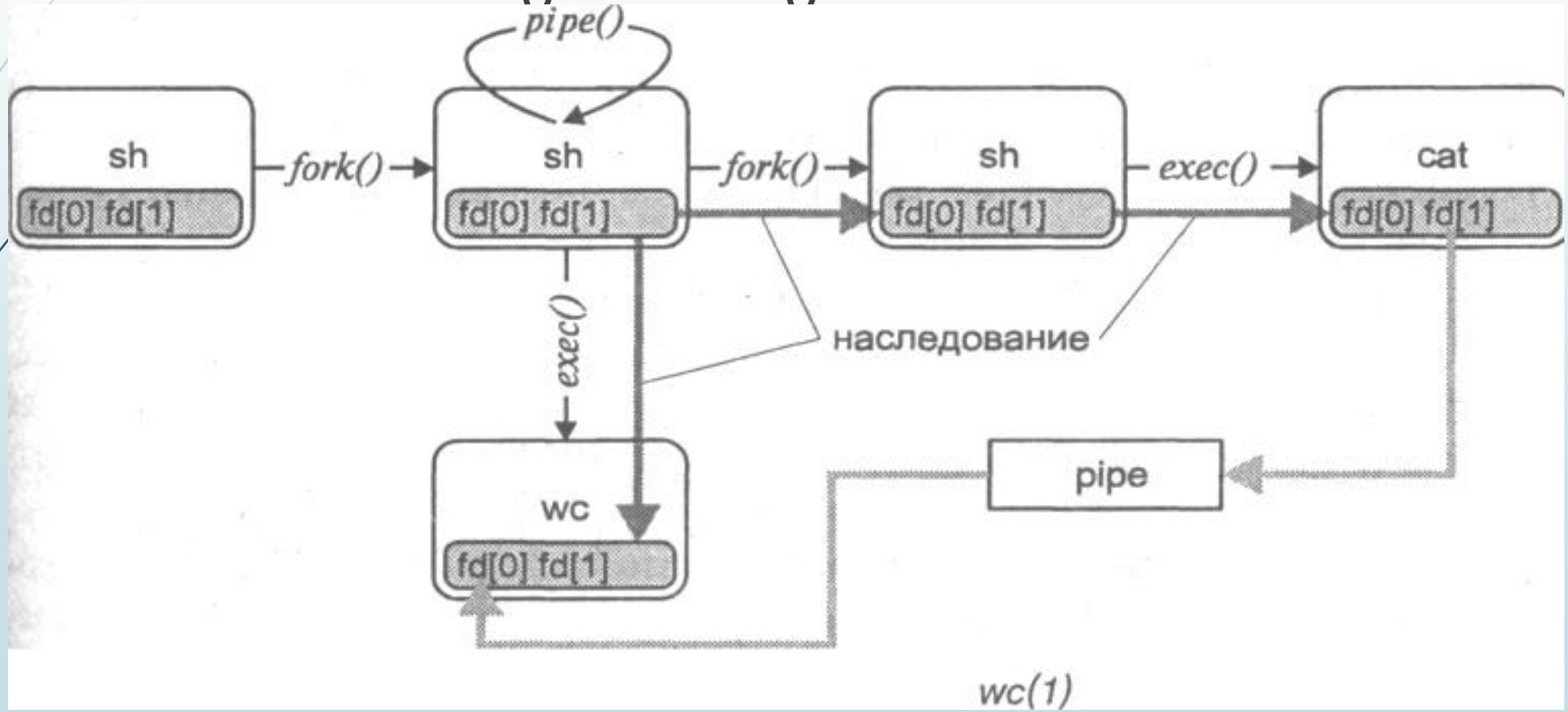
Т.е. доступ к дескрипторам **fildes** канала может получить сам процесс, вызвавший **pipe()**, и его дочерние процессы.

Средства межпроцессного взаимодействия: ***Каналы***

- Этот факт - серьезный **недостаток** каналов, поскольку они могут быть *использованы* ***для передачи данных только между родственными процессами.***
- **Каналы не могут использоваться в качестве средства межпроцессного взаимодействия между независимыми процессами.**

Средства межпроцессного взаимодействия: ***Каналы***

ПРИМЕР Создание канала между задачами cat() и wc()



Средства межпроцессного взаимодействия: ***FIFO***

Firt In First Out (первый пришел — первый вышел).

- FIFO очень похожи на каналы, поскольку являются ***однонаправленным средством передачи данных***, причем чтение данных происходит в порядке их записи.

Средства межпроцессного взаимодействия: ***FIFO***

- FIFO имеют ***имена***, которые позволяют независимым процессам получить к этим объектам доступ.
- Поэтому иногда FIFO также называют ***именованными каналами***

Средства межпроцессного взаимодействия: ***FIFO***

- FIFO являются средством UNIX System V и не используются в BSD.
- Впервые FIFO были представлены в System III, однако они до сих пор не документированы и поэтому мало используются.

Средства межпроцессного взаимодействия: ***FIFO***

FIFO является отдельным типом файла в файловой системе UNIX

- **ls -l**

покажет символ **p** в первой позиции

Средства межпроцессного взаимодействия: ***FIFO***

FIFO м.б. создан и из командной строки shell:

- **$ mknod name p**

После создания FIFO может быть открыт на запись и чтение, причем запись и чтение могут происходить в разных независимых процессах.

Средства межпроцессного взаимодействия: ***FIFO***

Для ***создания FIFO*** используется системный вызов ***mknod():***

- **int mknod(char *pathname, int mode, int dev);**

где

pathname - имя файла в ФС(имя FIFO),

mode - флаги владения, прав доступа …

dev - при создании FIFO игнорируется.

Средства межпроцессного взаимодействия: ***FIFO***

Правила работы канала FIFO :

- 1. При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.

Средства межпроцессного взаимодействия: ***FIFO***

Правила работы канала FIFO :

- 2. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.

Средства межпроцессного взаимодействия: ***FIFO***

Правила работы канала FIFO :

- 3. Если канал пуст и ни один процесс не открыл его на запись, при чтении из канала будет получено 0 байтов.
- Если один или более процессов открыли канал для записи, вызов **read()** будет заблокирован до появления данных .

Средства межпроцессного взаимодействия: ***FIFO***

Правила работы канала FIFO :

- 4. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно.

Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.

Средства межпроцессного взаимодействия: ***FIFO***

Правила работы канала FIFO :

- 5. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов **write()** блокируется до освобождения требуемого места, При этом атомарность операции не гарантируется.

Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал SIGPIPE, а вызов **write()** возвращает **0** с установкой ошибки (**errno=EPIPE**)

(если процесс не установил обработки сигнала **SIGPIPE**, производится обработка по умолчанию — процесс завершается).

Средства межпроцессного взаимодействия: ***FIFO***

пример приложения клиент-сервер, использующего FIFO для обмена данными.

клиент посылает серверу сообщение "Здравствуй, Мир!",

а сервер выводит это сообщение на терминал.

Средства межпроцессного взаимодействия: ***FIFO***

Сервер:

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

#define FIFO “fifo.1”
#define MAXBUFF 80
main()
```

Средства межпроцессного взаимодействия: ***FIFO***

```
main() {                                        // Сервер:
int readfd, n;
char buff[MAXBUFF]; //буфер для чтения данных из
                                    //FIFO
//Создадим специальный файл FIFO с открытыми
//для //всех правами доступа на чтение и запись
if(mknod(FIFO, S_FIFO | 0666, 0)<0 {
   printf("Невозможно создать FIFO\n"); exit(1);
}
//Получим доступ к FIFO
```

Средства межпроцессного взаимодействия: ***FIFO***

```
/*Получим доступ к FIFO*/                    // Сервер:
if ((readfd = open(FIFO, O_RDONLY))<0)
{
printf("Невозможно открыть FIFO\n");
exit(1); }
 /*Прочитаем сообщение ("здравствуй, мир") и
выведем его на экран*/
```

Средства межпроцессного взаимодействия: ***FIFO***

```
/*Прочитаем сообщение (“здравствуй, мир”) и выведем
его на экран*/                                  // Сервер:
while ((n = read(readfd, buff, MAXBUFF)) > 0)
        if(write(1, buff, n) != n)
                {printf(“Ошибка вывода\n”); exit(1); }
/*Закроем FIFO, удаление FIFO – дело клиента*/
close(readfd);
exit(0);
}
```

Средства межпроцессного взаимодействия: ***FIFO***

```
//Клиент:
#include <sys/types.h>
#include <sys/stat.h>
/*Соглашение об имени FIFO*/
#define FIFO “fifo.1”
main()
```

Средства межпроцессного взаимодействия: ***FIFO***

```
main()      //Клиент:
    {
    int writefd, n;
    /*Получим доступ к FIFO*/
    if ((writefd = open(FIFO, WRONLY)) < 0)
            {printf("Невозможно открыть FIFO\n");
             exit(1); }
    /*Передадим сообщение серверу FIFO*/
```

Средства межпроцессного взаимодействия: ***FIFO***

```
/*Передадим сообщение серверу FIFO*//Клиент:
if(write(writefd,”Здравствуй, Мир!\n”, 18) != 18)
        {printf(“Ошибка записи\n”);
         exit(1);}
/* Закроем FIFO */
close(writefd);
/* Удалим FIFO */
```

Средства межпроцессного взаимодействия: ***FIFO***

```
/* Удалим FIFO */          //Клиент:

if(unlink(FIFO)<0    {
        printf("Невозможно удалить    FIFO\n");
        exit(1); }
exit(0);
}
```

Средства межпроцессного взаимодействия: ***IPC***

- Отсутствие имен у каналов делает их недоступными для независимых процессов.
- Этот недостаток устранен у FIFO, которые имеют имена.

Другие средства межпроцессного взаимодействия, являющиеся более сложными, ***требуют дополнительных соглашений по именам и идентификаторам***.