Парные сокеты

Сокеты впервые были реализованы именно на платформе Unix (4.2BSD), однако, концепция сокетов, как универсального средства обмена данными между процессами, оказалась настолько удачна, что все современные системы поддерживают, по крайней мере, некоторое подмножество сокетов (Статья из серии "Программирование для Linux", журнал Linux Format *Андрей Боровский, symmetrica.net).*

Программы, обменивающиеся данными с помощью сокетов, могут работать в одной системе и в разных, используя для обмена данными как специальные объекты системы, так и сетевой стек.

Среди разных типов сокетов есть аналог неименованных каналов – парные сокеты (*socket pairs*). Как и не именованные каналы, парные сокеты создаются парами и не имеют имен. Естественно, что область применения парных сокетов – та же, что и у неименованных каналов, - взаимодействие между родительским и дочерним процессом.

Так же, как и в случае неименованного канала, один из дескрипторов используется одним процессом, другой – другим. В качестве примера использования парных сокетов рассмотрим программу sockpair.c, создающую два процесса с помощью fork().

Как и каналы (pipe), сокеты используют простой интерфейс, основанный на «файловых» функциях read(2) и write(2) (открывая сокет, программа Unix получает дескриптор файла, благодаря которому можно работать с сокетами, используя файловые функции), но, в отличие от каналов, сокеты позволяют передавать данные в обоих направлениях, как в синхронном, так и в асинхронном режиме.

Большинство программистов используют для работы с сокетами различные библиотеки высокого уровня, однако, высокоуровневые библиотеки, как правило, не позволяют задействовать всю мощь и все многообразие сокетов.

Парные сокеты создаются функцией socketpair(2). У функции socketpair() - четыре параметра. Первые три параметра функции те же, что и у функции socket(), а четвертым параметром является массив из двух переменных, в которых возвращаются дескрипторы сокетов.

Краткий обзор (SYNOPSIS)

```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol, int sv[2]);
```

Функция создает пару связанных сокетов.

Описание

При вызове функции **socketpair**() создается неименованная пара связанных сокетов, для определенного *домена или пространства имен (domain)*, типа (*type)* и протокола (protocol), который указывается опционально. Для более полной информации о данных параметрах см. socket(2). Четвертый параметр – массив файловых дескрипторов: файловые дескрипторы, используемые при ссылке на новые сокеты, возвращаются в элементы массива sv[0] и sv[1]. Эти два сокета неразличимы.

Возвращаемое значение

В случае успеха возвращается ноль. При ошибке возвращается значение -1, errno равно устанавливается для указания ошибки, а sv остается неизменным. В Linux (и других системах) socketpair() не изменяет sv при сбое. Требование, стандартизирующее это поведение, было добавлено в POSIX.1-2008 TC2.

Возвращаемые ошибки

EAFNOSUPPORT
The specified address family is not supported on this machine.

EFAULT The address *sv* does not specify a valid part of the process address space.

EMFILE The per-process limit on the number of open file descriptors has been reached.

ENFILE The system-wide limit on the total number of open files has been reached.

EOPNOTSUPP
The specified protocol does not support creation of socket pairs.

EPROTONOSUPPORT
The specified protocol is not supported on this machine.

Дескрипторы сокетов, возвращенные socketpair(), уже готовы к передаче данных, так что можно сразу применять к ним функции read()/write().

После вызова **fork()** каждый процесс получает оба дескриптора, один из которых он должен закрыть. Для закрытия сокета мы используем функцию close().

При использовании интерфейса программирования парных сокетов может возникнуть вопрос, а почему собственно эти функции относятся к сокетам? Ведь при работе с ними не используются ни адреса, ни модель клиент-сервер.

Да, это верно, но при этом, что функции **socketpair()** передаются значения домена и типа сокета, так что и формально, и с точки зрения реализации в системе используются именно сокеты.

Следует отметить, что указание домена в функции socketpair() является явно лишним, поскольку для этой парных советов система поддерживает только сокеты в домене **AF_UNIX** (вполне логичное ограничение, если учесть, что парные сокеты не имеют имен и предназначены для обмена данными между родственными процессами).

Дочерние процессы в программе sockpair.c используют парные сокеты для обмена приветствием.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#define STR1 "How are you?"
#define STR2 "I'm ok, thank you."
#define BUF_SIZE 1024
int main(int argc, char ** argv)
{
    int sockets[2];
   char buf[BUF_SIZE];
   int pid;
   if (socketpair(AF_UNIX, SOCK_STREAM, 0, sockets) < 0)
  {
     perror("socketpair() failed");
     return EXIT_FAILURE;
  }
   pid = fork(); // добавить проверку на -1
{
 …
```

```
}
   if (pid = = 0)
   {
      close(sockets[1]);
      write(sockets[0], STR1, sizeof(STR1));
      read(sockets[0], buf, sizeof(buf));
      printf("%s\n", buf);
      close(sockets[0]);
    } else
        {
            close(sockets[0]);
            read(sockets[1], buf, sizeof(buf));
            printf("%s\n", buf);
            write(sockets[1], STR2, sizeof(STR2));
            close(sockets[1]);
         }
}
```

Как видно из кода, парные сокеты в отличие от неименованных программных каналов (pipe) позволяют создать дуплексную связь между родственными процессами.

Задание на лабораторную работу «Парные сокеты»

Создать несколько дочерних процессов и организовать обмен сообщениями между процессом предком и процессами потомками.