

Лекция13

Обмен данными между процессами (продолжение).

Неименованные каналы

```
ewgenij@dew:~$ history | grep mount
441 sudo mount whirl:~/ ~/STUDIO/COUETTE_CUDA/whirl
507 history | grep mount
```

В операционных системах UNIX/LINUX все процессы являются узлами одного дерева процессов с корневым процессом **init**. Соответственно запущенные нами процессы *history* и *grep* являются дочерними по отношению к родительскому процессу **bash**. Обнаружив при разборе командной строки вертикальную линию оболочка создает неименованный канал и передает дескрипторы соответствующим дочерним процессам.

```
#include <windows.h>
```

```
#include <math.h>
```

```
int main()
```

```
{
```

```
HANDLE hPipeIn, hPipeOut;
```

```
SECURITY_ATTRIBUTES sa;
```

```
STARTUPINFO si;
```

```
PROCESS_INFORMATION pi;
```

```
char buff[80];
```

```
unsigned long dw;
```

```
double x;
```

```
sa.nLength=sizeof(sa);
```

```
sa.lpSecurityDescriptor=NULL;
```

```
sa.bInheritHandle=TRUE; //дочерние процессы наследуют  
                        //дескрипторы
```

Файл ***p1.c***

```
if(!CreatePipe(&hPipeIn,&hPipeOut,&sa,0)){
    printf("The pipe could not be created\n");
    exit(1);
}// Создание канала с дескриптором ввода hPipeIn и
// вывода hPipeOut
memset(&si,0,sizeof(si));
si.cb=sizeof(si);
si.dwFlags=STARTF_USESTDHANDLES;//использовать поля
//для стандартных потоков
si.hStdInput=hPipeIn; //перенаправление потока ввода
si.hStdOutput=GetStdHandle(STD_OUTPUT_HANDLE);
si.hStdError=GetStdHandle(STD_ERROR_HANDLE);
if(!
CreateProcess(NULL,"p2",NULL,NULL,TRUE,0,NULL,NULL,&si
,&pi)){
    printf("Could not create process, %i\n",GetLastError());
    exit(1);
}
```

```
CloseHandle(hPipeIn);
```

```
for(x=0.0;x<3.1416;x+=0.1){  
    sprintf(buff,"%g  %g\n",x,sin(x));  
    WriteFile(hPipeOut, buff, strlen(buff), &dw, NULL);  
}
```

```
*buff=(char)26;  
WriteFile(hPipeOut, buff, 1, &dw, NULL);
```

```
CloseHandle(hPipeOut);
```

```
return 0;  
}
```

```
#include <windows.h>
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
int main(int argc, char* argv[])
```

```
{
```

```
char buff[80];
```

```
fprintf(stdout, "<HTML>");
```

```
    fprintf(stdout, "<BODY bgcolor=#FFDD00>");
```

```
    fprintf(stdout, "<CENTER><TABLE>");
```

```
    while(fgets(buff, 80, stdin)) { //поток ввода перенаправлен!
```

```
        char *p=strchr(buff, '\n');
```

```
        if(p) *p='\0';
```

```
        fprintf(stdout, "<TR>");
```

```
            fprintf(stdout, "<TD>");
```

```
                fprintf(stdout, "<H1>");
```

```
                    fprintf(stdout, "%s", buff);
```

Файл **p2.c**

```
        fprintf(stdout,"</H1>");
        fprintf(stdout,"</TD>");
        fprintf(stdout,"</TR>");
    }
    fprintf(stdout,"</TABLE></CENTER>");
    fprintf(stdout,"</BODY>");
    fprintf(stdout,"</HTML>");

    return 0;
}
```

C:\...\Лекции\Лекция11-12\Лаб8>p1 > 1.html



Именованные каналы:

Сервер:

```
#include <stdio.h>
#include <windows.h>
```

Файл *np1.c*

```
void main()
{
HANDLE hPipe;
LPTSTR lpPipeName = TEXT("\\\\.\\pipe\\MyPipe");
char buff[255];
DWORD iBytesToRead = 255, i;
```



```
hPipe = CreateNamedPipe(  
    lpPipeName,                // имя канала  
    PIPE_ACCESS_DUPLEX, // чтение и запись из канала  
    PIPE_TYPE_MESSAGE | // передача сообщений по каналу  
    PIPE_READMODE_MESSAGE //режим чтения сообщений  
    PIPE_WAIT,                // синхронная передача сообщений  
    PIPE_UNLIMITED_INSTANCES, //число экземпляров  
    4096,                      // размер выходного буфера  
    4096,                      // размер входного буфера  
    NMPWAIT_USE_DEFAULT_WAIT, // тайм-аут клиента  
    NULL);                     // защита по умолчанию  
  
if (hPipe == INVALID_HANDLE_VALUE) {  
    printf("CreatePipe failed: error code %d\n",  
        (int)GetLastError());  
  
    return;  
}
```

```
if((ConnectNamedPipe(hPipe, NULL))==0)
{
printf("client could not connect\n");
return;
}
ReadFile(hPipe, buff, iBytesToRead, &iBytesToRead, NULL);
for(i=0; i<iBytesToRead; i++) printf("%c",buff[i]);
}
```

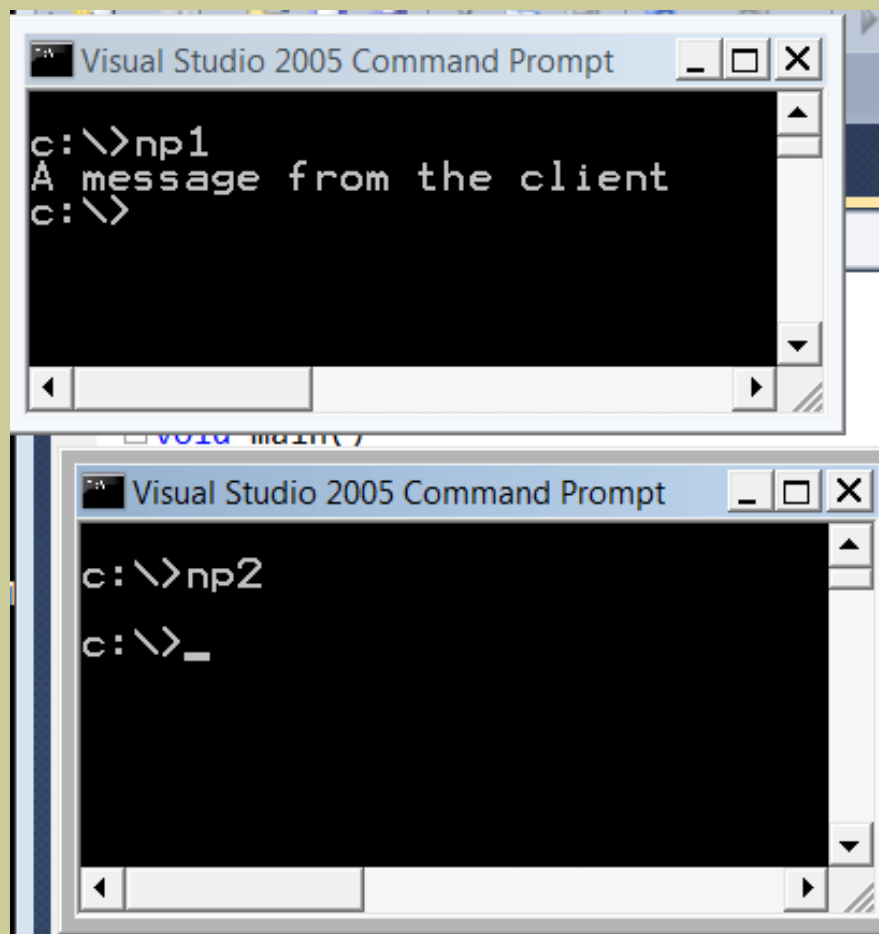
Клиент:

```
#include <stdio.h>
#include <windows.h>
void main(){
HANDLE hPipe;
LPTSTR lpPipeName = TEXT("\\\\.\\pipe\\MyPipe");
//TEXT("\\\\.\\WANDERER\\pipe\\MyPipe");
DWORD iBytesToWrite;
char buff[] = "A message from the client";
```

Файл ***np2.c***

```
hPipe = CreateFile(  
    lpPipeName, // имя канала  
    GENERIC_READ | // чтение и запись в канал  
    GENERIC_WRITE,  
    0,           // нет разделяемых операций  
    NULL,        // защита по умолчанию  
    OPEN_EXISTING, // открытие существующего канала  
    0,           // атрибуты по умолчанию  
    NULL);       // нет шаблона атрибутов
```

```
WriteFile(hPipe, buff, strlen(buff), &iBytesToWrite, NULL);  
CloseHandle(hPipe);  
}
```



Упражнение 1: протестировать разобранные программы.

Упражнение 2: написать «чат» один-к-одному в локальной сети.

Тема курсовой №3: написать многопользовательский «чат» в локальной сети.