Малки примерчета с dup и dup2

```c
//4.c
#include <unistd.h>

int main()
{
  write(3,"alabala\n",9);
  return 0;
}
```

```
dvd@BlackPearl:~$ gcc -o 4 4.c
dvd@BlackPearl:~$ ./4 3> 4.out
dvd@BlackPearl:~$ cat 4.out
alabala
dvd@BlackPearl:~$ ./4 3>> 4.out
dvd@BlackPearl:~$ cat 4.out
alabala
alabala
dvd@BlackPearl:~$
```

DUP(2)   Linux Programmer's Manual   DUP(2)

NAME

dup, dup2 - duplicate a file descriptor

SYNOPSIS

```
#include <unistd.h>

int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

DESCRIPTION

These system calls create a copy of the file descriptor oldfd.

dup() uses the lowest-numbered unused descriptor for the new descriptor.

dup2() makes newfd be the copy of oldfd, closing newfd first if necessary, but note the following:

* If oldfd is not a valid file descriptor, then the call fails, and newfd is not closed.
* If oldfd is a valid file descriptor, and newfd has the same value as oldfd, then dup2() does nothing, and returns newfd.

After a successful return from one of these system calls, the old and new file descriptors may be used interchangeably. They refer to the same open file description (see open(2)) and thus share file offset and file status flags; for example, if the file offset is modified by using lseek(2) on one of the descriptors, the offset is also changed for the other.

```c
//1.c
#include <unistd.h>
#include <fcntl.h>
int main(int argc, char** argv)
{
   //redirect standard output to argv[1]
   int fd=open(argv[1],O_WRONLY|O_TRUNC|O_CREAT, 0666);
   close(1);
   dup(fd);
   //dup2(fd,1);

   write(1,"ala\n",sizeof("ala\n"));
   write(fd,"bala\n",sizeof("bala\n"));

   close(fd);
   write(fd,"ne\n",sizeof("ne\n"));

   write(1,"da\n",sizeof("da\n"));

   return 0;
}
```

```
dvd@BlackPearl:~$ gcc -o 1 1.c
dvd@BlackPearl:~$ ./1 1.out
dvd@BlackPearl:~$ cat 1.out
ala
bala
da
dvd@BlackPearl:~$
```

```
dvd@BlackPearl:~$ rm 1.out
dvd@BlackPearl:~$ ./1 1.out > 1.out?
dvd@BlackPearl:~$ ls 1.out*
1.out  1.out?
dvd@BlackPearl:~$ cat 1.out
ala
bala
da
dvd@BlackPearl:~$ cat 1.out?
dvd@BlackPearl:~$
```

```
//2.c
#include <unistd.h>
#include <fcntl.h>
#include<stdio.h>
int main(int argc, char** argv)
{

   int fd=dup(1);
   printf("%d\n",fd);

   write(1,"ala\n",sizeof("ala\n"));
   write(fd,"bala\n",sizeof("bala\n"));

   close(1);
   write(1,"ne\n",sizeof("ne\n"));

   write(fd,"da\n",sizeof("da\n"));

   return 0;
}
```

```
dvd@BlackPearl:~$ gcc -o 2 2.c
dvd@BlackPearl:~$ ./2
3
ala
bala
da
dvd@BlackPearl:~$ ./2 > 2.out
dvd@BlackPearl:~$ cat 2.out
ala
bala
da
dvd@BlackPearl:~$
```

```c
//3.c
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

char buffer[8];
int main(int argc, char** argv)
{
   int fd2=dup(0);
   ssize_t rb=read(fd2,buffer,8);
   printf("Read %s\n",buffer);
   int fd1=open(argv[1],O_WRONLY|O_TRUNC);
   int fd3=open(argv[1],O_RDONLY);
   dup2(fd3,fd1);
   if(write(fd1,buffer,rb)==-1)
   {
      perror(argv[1]);
   }
   if(read(fd3,buffer,32)==-1)
   {
      perror(argv[1]);
   }
   return 0;
}
```

```
dvd@BlackPearl:~$ gcc -o 3 3.c
dvd@BlackPearl:~$ echo 1234567890 >
3.txt
dvd@BlackPearl:~$ ./3 3.txt
abc
Read abc

3.txt: Bad file descriptor
dvd@BlackPearl:~$ cat 3.txt
dvd@BlackPearl:~$ ./3 3.txt
abcdefghij
Read abcdefgh
3.txt: Bad file descriptor
dvd@BlackPearl:~$ ij
bash: ij: command not found
dvd@BlackPearl:~$
```

Край