

Shell redirection

Source: <http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Sh/lecture.html>

Shell

- There are many different shells that people use under Unix. Two important ones are the Bourne shell (sh) and the C shell (csh).
- Most people use the C shell (or Bash or tcsh or zcsh) as an interactive interpreter to execute programs because it has the history function and some other useful properties.

Redirection operators

- The basic redirection functions are used similarly on Bourne shell and C shell.
- `> d` : Writes standard output to the file `d`
- `>> d` : Appends standard output to the file `d`.
- `< d` : Takes standard input from the file `d`.

Shell redirection

echo prints the given string to stdout.

File Edit View Search Terminal Help

```
abc:test$ echo "System programming"
```

```
System programming
```

```
abc:test$ echo "System programming" > file1
```

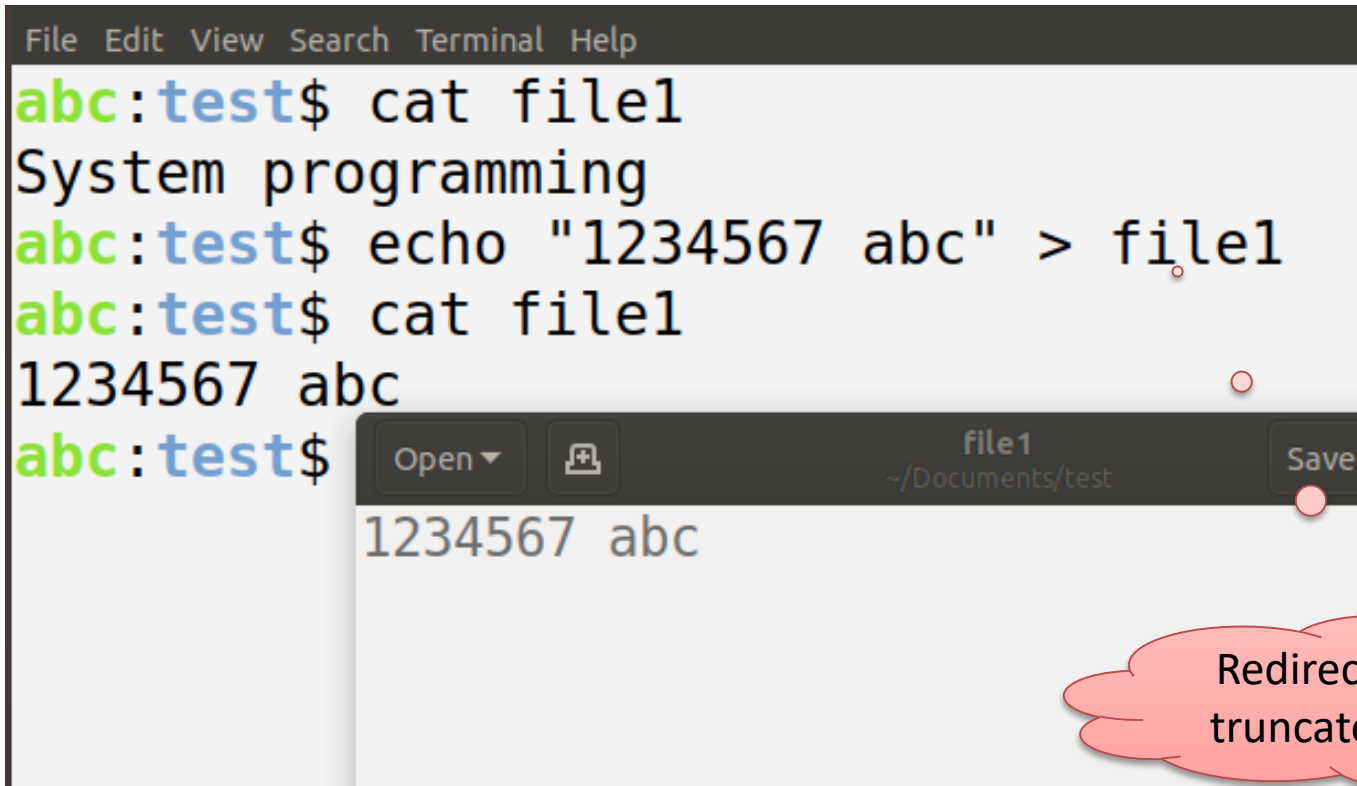
```
abc:test$ ls -l
```

```
total 4
```

```
-rw-r--r-- 1 abc abc 19 Apr  5 22:47 file1
```

Stdout is redirected to file

Redirection in truncate mode



The image shows a terminal window with a dark title bar containing 'File Edit View Search Terminal Help'. The terminal output is as follows:

```
abc:test$ cat file1
System programming
abc:test$ echo "1234567 abc" > file1
abc:test$ cat file1
1234567 abc
abc:test$
```

Overlaid on the terminal is a file editor window titled 'file1' with the path '~/Documents/test'. It has an 'Open' button, a file icon, and a 'Save' button. The editor shows the text '1234567 abc'.

Redirection to file1
truncate its content

Shell redirection

```
File Edit View Search Terminal Help  
abc:test$ ./prog1
```

```
i=0  
i=1  
i=2  
i=3  
i=4  
i=5  
i=6  
i=7  
i=8  
i=9
```

```
abc:test$
```

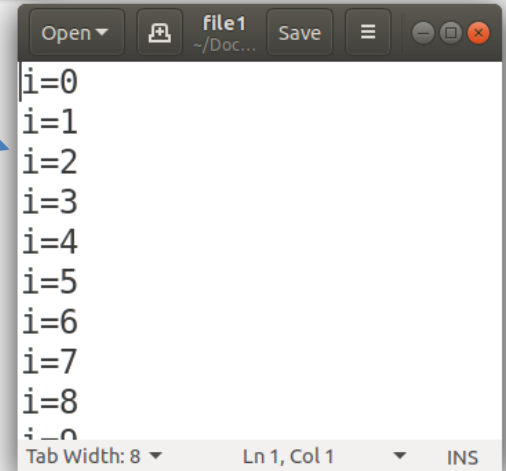
```
i=0  
i=7  
i=8  
i=9
```

```
abc:test$ ./prog1 > file1
```

```
abc:test$ cat file1
```

```
i=0  
i=1  
i=2  
i=3  
i=4  
i=5  
i=6  
i=7  
i=8  
i=9
```

Output of the
program is redirected to
file1



```
Open File1 Save  
~/Doc...  
i=0  
i=1  
i=2  
i=3  
i=4  
i=5  
i=6  
i=7  
i=8  
i=9  
Tab Width: 8 Ln 1, Col 1 INS
```

Redirection in append mode

```
abc:~$ echo "12345 67890" > file1.txt
abc:~$ echo "abcdef" >> file1.txt
abc:~$ cat file1.txt
12345 67890
abcdef
```

Truncate mode

Append mode

- While **>** truncates the content **>>** works in append mode

cat

- **cat** is one of the frequently used Shell commands.

```
Terminal File Edit View Search Terminal Help
CAT(1) User Commands CAT(1)

NAME
    cat - concatenate files and print
    on the standard output

SYNOPSIS
    cat [OPTION]... [FILE]...

DESCRIPTION
    Concatenate FILE(s) to standard
    output.

1/95 12% (press h for help or q to quit)
```


cat

```
File Edit View Search Terminal Help
abc:~$ cat
abc
abc
98765
98765
ZXC
ZXC
█
```

- If we just write `cat` and press enter, it writes the entries entered on the screen again.
- In other words it takes ***stdin*** and directs it to ***stdout***.

cat

File Edit View Search Terminal Help

```
abc:~$ cat file1.txt
```

```
12345 67890
```

```
abcdef
```

```
abc:~$ echo "zyxw" >file2.txt
```

```
abc:~$ cat file1.txt file2.txt
```

```
12345 67890
```

```
abcdef
```

```
zyxw
```

```
abc:~$ cat file1.txt file2.txt >file3.txt
```

```
abc:~$ cat file3.txt
```

```
12345 67890
```

```
abcdef
```

```
zyxw
```

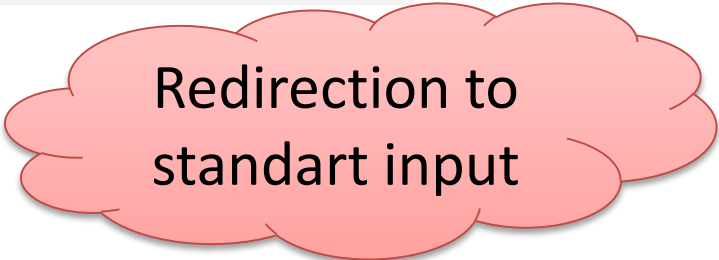
```
abc:~$
```

Cat prints the contents of two files

We can merge the files into one file

Shell redirection

```
abc:~$ cat <file1.txt  
12345 67890  
abcdef  
abc:~$
```



Redirection to
standart input

Shell redirection

```
File Edit View Search Terminal Help
abc:test$ ./prog2
abcde
Entered string: abcde
abc:test$
```

```
~/Documents/test
#include <stdio.h>
#include <stdlib.h>
int main(){
char str[100];
scanf("%s",str);
printf("Entered string: %s\n",str);
return 0;
}
```

```
File Edit View Search Terminal Help
abc:test$ echo "abcde" >file1.txt
abc:test$ cat file1.txt
abcde
abc:test$ ./prog2 <file1.txt
Entered string: abcde
abc:test$
```

Redirection to
standart input.

Shell redirection

```
File Edit View Search Terminal Help
abc:test$ echo "d1 file" > d1
abc:test$ cat d1
d1 file
abc:test$ cat d2
cat: d2: No such file or directory
abc:test$ cat d1 d2
d1 file
cat: d2: No such file or directory
abc:test$ cat d1 d2 > d3
cat: d2: No such file or directory
abc:test$ cat d3
d1 file
abc:test$
```

It directs stdout to file.
Error message printed
using stderr stream.

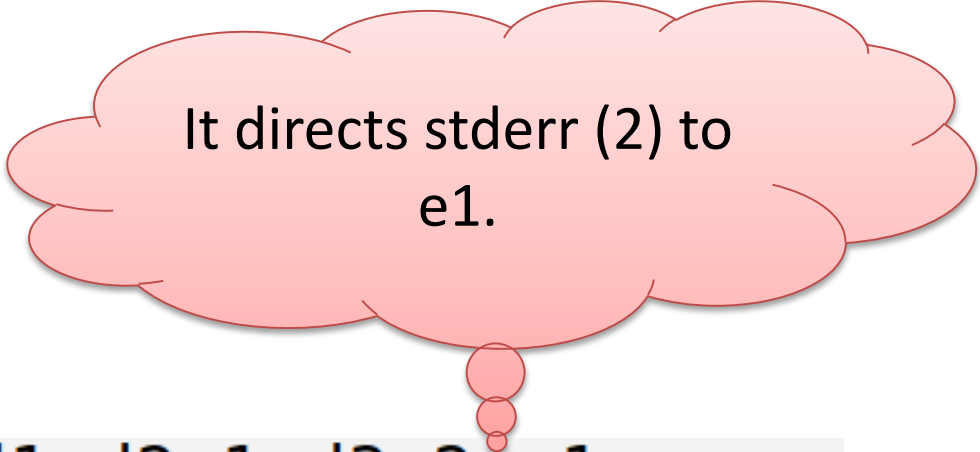
Shell redirection

```
abc:test$ cat d1 d2 1> d3  
cat: d2: No such file or directory  
abc:test$
```

It directs stdout (1) to file. Error message printed using stderr (2) stream.

🖐 File descriptors:
0: stdin
1: stdout
2: stderr

Shell redirection



It directs stderr (2) to e1.

```
abc:test$ cat d1 d2 1>d3 2>e1
abc:test$ cat d3
d1 file
abc:test$ cat e1
cat: d2: No such file or directory
```

Shell redirection

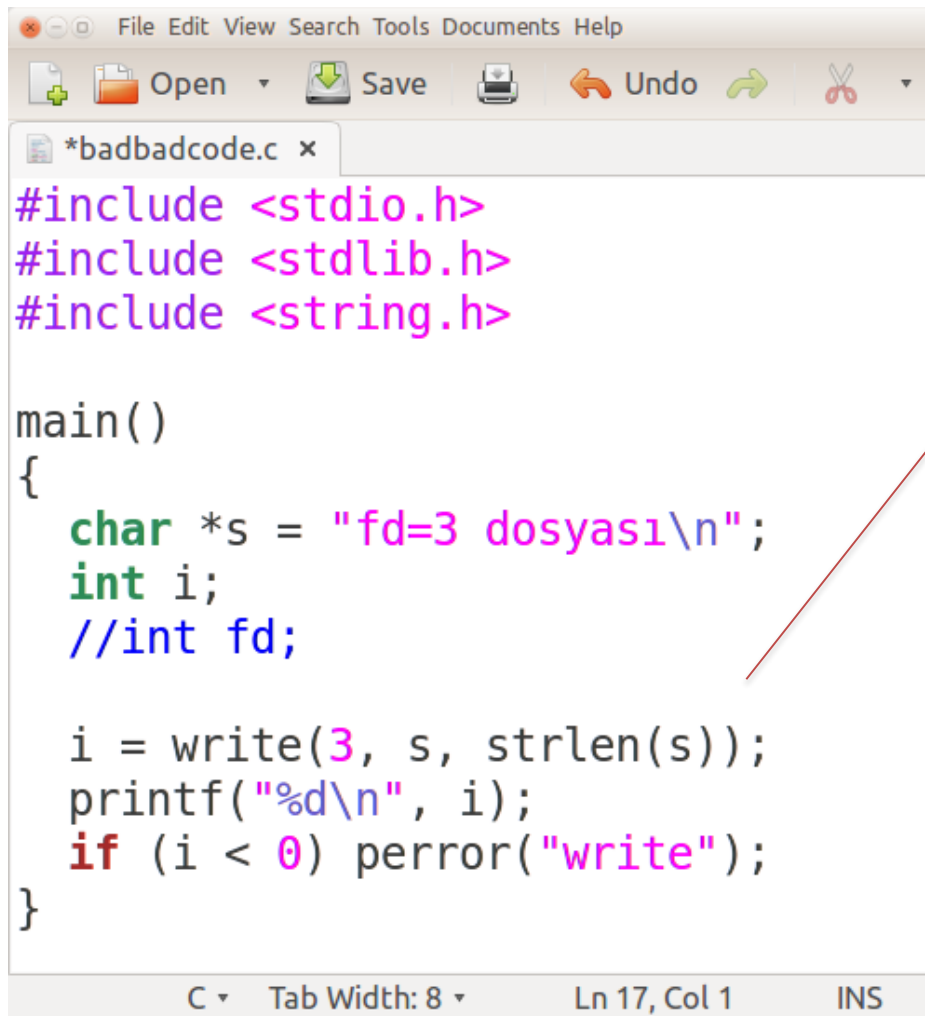
```
abc:test$ cat d1 d2 >& d3
abc:test$ cat d3
d1 file
cat: d2: No such file or directory
abc:test$
```

Stdout and stderr
merged together and
directed to d3

```
abc:test$ cat d1 d2 >d3 2>&1
abc:test$ cat d3
d1 file
cat: d2: No such file or directory
abc:test$
```

Alternative
to above

Shell redirection

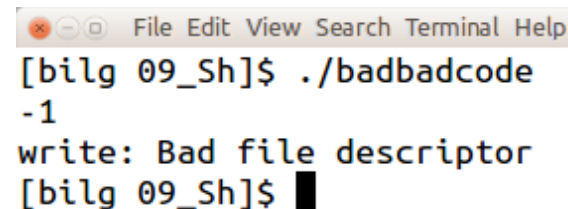


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

main()
{
    char *s = "fd=3 dosyas1\n";
    int i;
    //int fd;

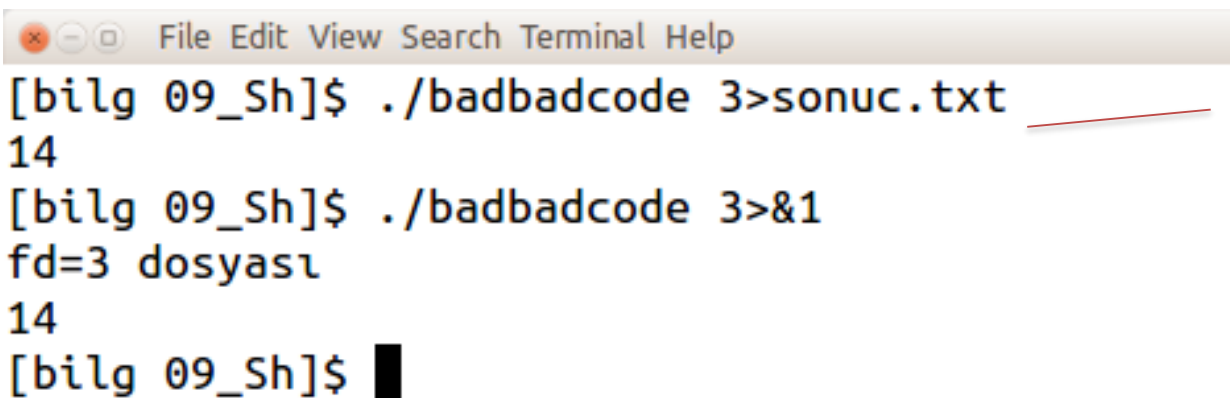
    i = write(3, s, strlen(s));
    printf("%d\n", i);
    if (i < 0) perror("write");
}
```

s character array is written to a file described by 3 file descriptor.
But this file should be opened before writing something.



```
[bilg 09_Sh]$ ./badbadcode
-1
write: Bad file descriptor
[bilg 09_Sh]$
```

Shell redirection

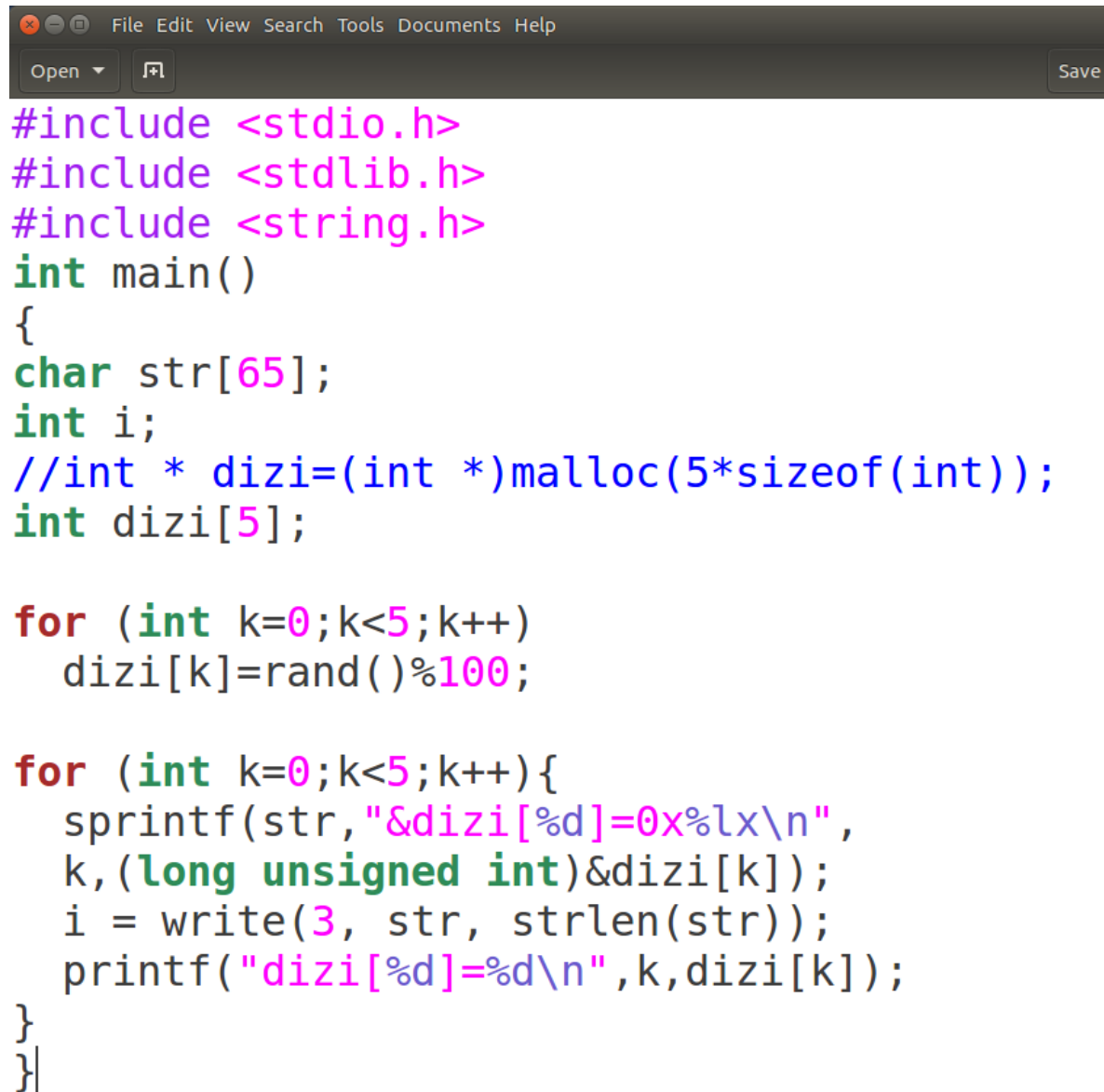


```
File Edit View Search Terminal Help
[biłg 09_Sh]$ ./badbadcode 3>sonuc.txt
14
[biłg 09_Sh]$ ./badbadcode 3>&1
fd=3 dosyası
14
[biłg 09_Sh]$
```

If the file descriptor specified with 3 in the code is directed to a file such as sonuc.txt, the program will not generate an error.

Alternatively fd=3 can be redirected to 1 and the message is printed on the screen.

Shell redirection



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char str[65];
    int i;
    //int * dizi=(int *)malloc(5*sizeof(int));
    int dizi[5];

    for (int k=0;k<5;k++)
        dizi[k]=rand()%100;

    for (int k=0;k<5;k++){
        sprintf(str, "&dizi[%d]=0x%lx\n",
            k, (long unsigned int)&dizi[k]);
        i = write(3, str, strlen(str));
        printf("dizi[%d]=%d\n", k, dizi[k]);
    }
}
```

Shell redirection

Terminal File Edit View Search Terminal Help

```
bilg:8_Sh$ ./cikti
```

```
dizi[0]=83
```

```
dizi[1]=86
```

```
dizi[2]=77
```

```
dizi[3]=15
```

```
dizi[4]=93
```

```
bilg:8_Sh$ ./cikti 3>dosya
```

```
dizi[0]=83
```

```
dizi[1]=86
```

```
dizi[2]=77
```

```
dizi[3]=15
```

```
dizi[4]=93
```

```
bilg:8_Sh$ cat dosya
```

```
&dizi[0]=0x7ffef07a8300
```

```
&dizi[1]=0x7ffef07a8304
```

```
&dizi[2]=0x7ffef07a8308
```

```
&dizi[3]=0x7ffef07a830c
```

```
&dizi[4]=0x7ffef07a8310
```

```
bilg:8_Sh$ █
```

File Edit View Search Tools Documents Help

Open ▾



Save

```
&dizi[0]=0x7ffef07a8300
```

```
&dizi[1]=0x7ffef07a8304
```

```
&dizi[2]=0x7ffef07a8308
```

```
&dizi[3]=0x7ffef07a830c
```

```
&dizi[4]=0x7ffef07a8310
```

Plain Text ▾

Tab Width: 8 ▾

Ln 1, Col 1 ▾

INS

Shell redirection

```
Terminal File Edit View Search Terminal Help
bilg:8_Sh$ ./cikti 3>&1
&dizi[0]=0x7ffeaeedc250
dizi[0]=83
&dizi[1]=0x7ffeaeedc254
dizi[1]=86
&dizi[2]=0x7ffeaeedc258
dizi[2]=77
&dizi[3]=0x7ffeaeedc25c
dizi[3]=15
&dizi[4]=0x7ffeaeedc260
dizi[4]=93
bilg:8_Sh$
```