

# 13. Functions

在 shell 中，函数（function）是可以在脚本或交互式 shell 中调用的。函数可以接受参数，并返回一个值。函数可以简化代码，提高代码的可读性和可维护性。在 shell 中，函数通常以以下形式定义：

```
./library.sh
```

在 shell 中，函数通常以以下形式定义：

在 shell 中，函数通常以以下形式定义：

在 shell 中，函数通常以以下形式定义：

- 在 shell 中，函数通常以以下形式定义：
- 在 shell 中，函数通常以以下形式定义：
- **return** 命令用于返回函数的值。
- **echo** 命令用于输出到 **stdout**，例如 `c=`expr $a + $b`` 将 `a` 和 `b` 相加并输出结果。

在 shell 中，函数通常以以下形式定义：

在 shell 中，函数通常以以下形式定义：

```
#!/bin/sh
# A simple script with a function...

add_a_user()
{
  USER=$1
  PASSWORD=$2
  shift; shift;
  # Having shifted twice, the rest is now comments ...
  COMMENTS=$@
  echo "Adding user $USER ..."
```

```
###  
# Main body of script starts here  
###  
  
echo "Start of script..."  
  
add_a_user bob letmein Bob Holness the presenter  
add_a_user fred badpassword Fred Durst the singer  
add_a_user bilko worsepassword Sgt. Bilko the role model  
  
echo "End of script..."
```

```

add_a_user bob letmein Bob Holness
Start of script..." echo add_a_user bob letmein Bob Holness
add a user :

```

0000 00 00 0000 \$100 000000 \$100 000000 000000 00 000000 bob000 000000 .  
 0000 00 0000 '00 ' \$100 000000 0000 000000 00 0000 00 0000 0000 0000 :  
 A=\$100 00 0000 000000 00 0000 000000 0000 .00 00 00 0000 \$A00 0000 0 0000  
 .0000 0000 00 000000 \$30000 000000 \$@00 000000 .00 00 0 0000 0000  
 0000 000000 000000 .00 0000 00 0000 00 0000 000000 00 0000 00 00  
 0000 000000 .

???

[illegible]

```
#!/bin/sh

myfunc()
{
    echo "I was called as : $@"
    x=2 }

### Main script starts here

echo "Script was called with $@"
x=1

echo "x is $x"

myfunc 1 2 3

echo "x is $x"
```

scope.sh a b c

```
Script was called with a b c
x is 1
I was called as : 1 2 3
x is 2
```

```

$@ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
(global) , myfunc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

```

```

out.log"
tee
. , "myfunc 1 2 3 | tee
"x 1"
. myfunc()
. Astrid "| tee"
"ls | grep foo"
, grep
, ls
stdin ls stdout
tee
tee myfunc()
.
.
,
:
```

```
#!/bin/sh

myfunc()
{
    echo "\$1 is $1"
    echo "\$2 is $2"
    # cannot change $1 - we'd have to say:
    # 1="Goodbye Cruel"
    # which is not a valid syntax. However, we can # change $a:
    a="Goodbye Cruel"
}

### Main script starts here

a=Hello
b=World
myfunc $a $b
echo "a is $a"
echo "b is $b"
```

❏ ❏ ❏❏❏ ❏❏ \$a❏ ❏❏❏ "Hello World"❏ ❏❏❏ "Goodbye Cruel World"❏ ❏❏❏ .

## ??(Recursion)

❏❏ ❏❏❏ ❏ ❏❏❏ . ❏❏ ❏❏❏ ❏❏ ❏❏ ❏❏❏ :

```
#!/bin/sh

factorial()
{
    if [ "$1" -gt "1" ]; then
        i=`expr $1 - 1`
        j=`factorial $i`
        k=`expr $1 \* $j`
        echo $k
    else
        echo 1
    fi
}
```

```

while :
do
    echo "Enter a number:"
    read x
    factorial $x
done

```

```

[[[  [[  [[  [[  [[[[  [[  [[[[[[  [[[[  [[  [[  [[[[  [[[[[[  . [[
[[[[[[  [[  [[  [[[[  [[  [[  [[  [[  [[[[  .

```

## common.lib

```

# common.lib
# Note no #!/bin/sh as this should not spawn
# an extra shell. It's not the end of the world # to have one, but clearer not to.
#
STD_MSG="About to rename some files..."

rename()
{
    # expects to be called as: rename .txt .bak
    FROM=$1
    TO=$2

    for i in *$FROM
    do
        j=`basename $i $FROM`
        mv $i ${j}$TO
    done
}

```

## function2.sh

```

#!/bin/sh
# function2.sh
. ./common.lib
echo $STD_MSG
rename txt bak

```

## function3.sh

```
#!/bin/sh
# function3.sh
. ./common.lib
echo $STD_MSG
rename html html-bak
```

```
common.lib function2.sh function3.sh  
function1.sh  
mainlib
```

## Exit Codes

□□ □□ □□ □□ □□ □□□□ □□ □ □ □□ (14□) □ □□ □□ □□□□ . □□

□□ □□ □□ □□ □□□□□□ .

```
#!/bin/sh

adduser()
{
    USER=$1
    PASSWORD=$2
    shift ; shift
    COMMENTS=$@
    useradd -c "${COMMENTS}" $USER
    if [ "$?" -ne "0" ]; then
        echo "Useradd failed"
        return 1
    fi
    passwd $USER $PASSWORD
    if [ "$?" -ne "0" ]; then
        echo "Setting password failed"
        return 2
    fi
    echo "Added user $USER ($COMMENTS) with pass $PASSWORD"
}

## Main script starts here
```

```
adduser bob letmein Bob Holness from Blockbusters
if [ "$?" -eq "1" ]; then
    echo "Something went wrong with useradd"
elif [ "$?" -eq "2" ]; then
    echo "Something went wrong with passwd"
else
    echo "Bob Holness added to the system."
fi
```

□ □□□□ □ □ □ □ (useradd □ passwd)□ □□□ □□ □ □ □□□ □□□□ . □□  
□□ □ □□ □□ □□ □□ □ □ □ □ □ 1□ , □□□□ □□ □ □ □ □ □ 2□  
□□□□ . □□ □ □ □□□ □□ □□ □□ □ □ □□□ .

Revision #1

Created 17 January 2024 01:23:46 by Enigma

Updated 17 January 2024 01:41:46 by Enigma