[1.The basis of \*nix and shell](#header-c114)  
 [1.1 shell](#header-c6)  
 [1.2 environment](#header-c8)  
 [1.3 Application](#header-c10)  
 [1.3.1 crontab](#header-c11)  
 [1.4 magic word, the first line](#header-c26)  
 [1.5 check the help content//man xx](#header-c28)  
 [1.6 HOW-TO define variable , get user input](#header-c30)  
[2.Defference of different shell（ksh,bash,sh)](#header-c34)  
 [2.1 'sh', short for shell](#header-c35)  
 [2.2 bash vs ksh](#header-c41)  
 [2.2.1 characteristics of bash](#header-c46)  
[3 Foreground & Background](#header-c65)  
[4. Special variable & operator & statement](#header-c69)  
 [4.1 operator//used in test condition](#header-c71)  
 [4.2 condition in statement//use return value](#header-c75)  
 [4.2.1 Use true and false as condition](#header-c79)  
 [4.3 statements](#header-c81)  
 [4.1 flow control//loop control](#header-c83)  
[5.Most common commands:](#header-c87)  
[6.PIPE，HereDoc，STDIN/STDOUT，redirect，operator，special file(fifo,link)](#header-c90)  
[7.Function, arguments](#header-c91)  
[8.shell statements(while,for,if..)](#header-c92)  
[9.awk basis(variable，string，statements,pattern&action）](#header-c93)  
[10.sed basis(AIX)](#header-c94)  
[11.Regex/Glob/Wildcard](#header-c95)  
[12.Different between AIX/Linux commands](#header-c96)  
[13.Debug shell script](#header-c97)

# 1.The basis of \*nix and shell

## 1.1 shell

system level -- connector -- user  
core part --- command line  
run command in command line --->shelll interactive session  
  
#---------------view   
input --> command  
output ---> screen output  
  
#--------------c++  
no need compile  
shell environment :  
1. explain the code  
2. execute code  
3. return the result to output  
# ------------- shell script  
in a word, group the command into a file

## 1.2 environment

ksh, the environment means all initial/baic variables provided by ksh.  
---level 0 basic vairables: PWD LOGNAME PATH HOME  
---level 1 run profile, to set custom variables : '.' #source, means run file in current shell  
level0 + level 1 + ...  
Execution order:  
Login/run script/enter command line  
1. /etc/ # system profile /etc/bash\_profile /etc/profile  
2. $HOME/.profile # used to login , default profile  
3. shell specific profile, depends on the default shell, used for login shell  
$HOME/.bash\_profile # usually used in linux. bash  
4. shell specific script/profile, each time open a shell (non-login)  
.bashrc  
.kshrc

## 1.3 Application

### 1.3.1 crontab

The default environment is the level 0, HOME --- > $HOME of user PATH --> /usr/bin;/bin Initial path --> Will not execute .profile/.bash\_profile **Execute any file in crontab, need to set the profile in the script**

. ~/.profile ALL scripts deployed in crontab

## 1.4 magic word, the first line

#!/usr/bin/ksh #used for OS to identify the file type  
# text, linux/Unix will read the first line, to find a excutable file to run the file as batch command  
'#!' the magic word  
#! /bin/sh  
#! env python #env will search the first location of python executable  
#! /usr/bin/php  
./script.sh   
OS will find the first line, run  
ksh ./script.sh  
# the first line will be suppressed.  
script.sh:  
#! /bin/bash  
./script.sh --> load to bash  
ksh ./script.sh -->load to ksh  
#! `which python` # /ocs/tool/runtime/bin/python  
#! /bin/bash -->   
ksh -c "/path/to/file.sh"  
# crontab -l

## 1.5 check the help content//man xx

man cmd  
# default manaual page is shell/ command  
man test ## listed all operators for value compare  
man ksh ## listed all features for a shell, include statements/variables  
man N cmd # default N is 1 -- shell command  
man /etc/passwd # system profile  
man 2 time # for programmer  
man man # check manual page no  
  
# kill -L  
man select # if no result  
[bobo@download] which select  
[bobo@download] # not found, or the command is 'shell internal', so we need to check 'man ksh'  
  
# print internal  
echo # inerternal or not  
printf # external command , --- shell

## 1.6 HOW-TO define variable , get user input

fmt: var=value  
Remarks: '=' , no space is allowed in left or right  
var =2 # var will be regarded as command  
var= 2 # var= will be regarded as command  
  
fmt: number , by default, shell will treat all variables as string.  
 Number type, only valid in expression  
 [bobo@download] echo $((var\*5))  
10  
  
fmt: define string withe space, use double quote or single quote  
var="a b "  
var='a b '  
double vs single  
double: some special chars still got their special usage, '\',``,$()  
single: all special chars be reargded as normal/raw string  
'\' ---> use backslash to escape the next char  
\\ -->'\'  
---------------  
distinguish between 'escape' and '\magic'  
'\n' # magic word in command, echo /print/sed/grep # newline  
'\t' # tab  
If we don't want the magic word or special char act as it's real meaning, we use '\' to escape  
"\\n" -->r'\n'  
"\\*" -->r'\*''  
  
[bobo@download] print 'ab\ncd'  
ab  
cd  
[bobo@download] print "ab\ncd"  
ab  
cd  
[bobo@download]   
  
escape from left to right, the quote matches from left to right  
  
Double quot total number must be even number, or shell will consider the quote not finished  
[bobo@download] echo "a"b"c  
> # prompt to line to contine, '>' --> PS2  
[bobo@download] export PS2="#"  
[bobo@download] echo "a"b"c   
#  
  
  
fmt: define varibale with mutiple lines  
var="a  
b # each enter will be regarded as new line  
c"  
var="b^Jb^Jc" # '^J' -->when we trace back the command history, indicates for new line  
  
Remarks:  
1. If we want to view the real value for multiple line varibale, we must use "$var"  
[bobo@download] echo "$var"  
b  
b  
c  
2. Otherwise, 'echo $var', the new line will be suppressed,  
[bobo@download] echo $var  
b b c  
[bobo@download]   
3. Furthermore , for command result `cmd`, sometimes it is multiple-line value  
echo `cmd` ## transform to oneline  
echo "`cmd`" ## real value  
[bobo@test] echo "`ls`"|wc -l  
12  
[bobo@test] echo "`ls`"  
algorithm  
backup  
cpp  
  
[bobo@test] echo `ls`  
algorithm backup cpp graphviz oracle perl python sh shell sql tmp topic  
[bobo@test] echo `ls`|wc -l  
1  
[bobo@test]   
  
  
fmt: var="a\ # add '\' in the end(no space )  
b\ # as one line  
c  
"  
[bobo@test] var="a\  
#b  
#c\  
#d"  
  
 [bobo@test] echo "$var"   
ab  
cd  
[bobo@test]   
  
fmt: use 'let' to define or calculate/assignment #universal for AIX/linux, ksh/bash ..., not valid for string, only calculate/definition  
let var=value # assign, define  
[bobo@test] let a=2  
[bobo@test] echo $a  
2  
[bobo@test]   
[bobo@test] let b=3  
[bobo@test] echo $b  
3  
[bobo@test] let c=a+b #calculate, c=$((a+b)), c=$a+$b  
[bobo@test] echo $c  
5  
  
[bobo@test] typeset c="abc" # c="abc"  
[bobo@test] echo $c  
abc

# 2.Defference of different shell（ksh,bash,sh)

## 2.1 'sh', short for shell

sh is the oldest shell,

In modern UNIX, it's not the original oldest shell, it's a link to a default shell

[bobo@sh] ls -l /bin/sh  
lrwxrwxrwx 1 root root 4 3月 11 16:41 /bin/sh -> bash  
# on AIX  
link to ksh  
# on Linux  
link to bash  
#!/bin/sh -- >#! ksh or bash

## 2.2 bash vs ksh

Mostly used in modern linux. For AIX , use ksh AIX, bsh is not bash,

ocs@MTG8\_OCS1\_1:[/ocs]$cat /etc/shells  
  
/bin/csh  
/bin/ksh  
/bin/psh  
/bin/tsh  
/bin/bsh  
/usr/bin/csh  
/usr/bin/ksh  
/usr/bin/psh  
/usr/bin/tsh  
/usr/bin/bsh

### 2.2.1 characteristics of bash

1. expression

* File name:   
  [^0-9] ^--exclude [0-9]:0 1 ...  
  {a,b}

1. child-shell / sub-shell

* in a shell script, when we run /call a command , we run it in the current shell.  
  When we run `date` $(date) date|wc -l, we spawn a sub-shell, run the command in the child-shell:  
  1. Inherent all the variables from the parent/current shell  
  PWD  
  HOME  
  LOGNAME # username  
  bobo@ubuntu:/media/Data/Work/CodeLib/test/sh$ abc=123 # defined in current shell  
  bobo@ubuntu:/media/Data/Work/CodeLib/test/sh$ echo `echo $abc` # can be accessed in sub-shell  
  123  
  2. The output of child-shell will redirect to parent  
  The variable value cannot pass from child to parent  
  [bobo@sh] echo `dde=abc`  
    
  [bobo@sh] echo $dde  
    
  [bobo@sh]   
    
  3. cmd1 | cmd2 | cmd3  
  cmd1, cmd2, cmd3 run parallel in 3 child-shell  
  input-->cmd1 -->output <--cmd2 -->output <---cmd3-->output

### 2.2.2 characteristics of ksh

* cmd1 | cmd2 | cmd3  
  The last command cmd3 run on current shell  
  cmd1 and cmd2 run in chld-shell  
  ls ~|grep ".sh"|while read line  
  do  
   var=$line  
  done  
  echo "last line: $var"  
    
  [bobo@sh] bash test.sh  
  last line:   
  [bobo@sh]   
    
  [bobo@sh] ksh test.sh  
  last line: userenv.sh  
  [bobo@sh]

# 3.Start from hello world  
  
Main entry --> start from the first command  
'#' comment line   
echo "String" /String  
  
[bobo@sh] which print  
/usr/bin/print  
[bobo@sh] which printf  
/usr/bin/printf  
[bobo@sh]   
  
```shell  
printf "format description, %8d%s" "first string/param" "secdond string/param"  
[bobo@sh] printf "This is %2d number %#8s format\n" 0.22222 test   
This is 00 number test format  
d- decimal  
s- string  
f-float  
%xxd/s/f  
print a bit different with echo

[bobo@sh] hw.sh  
Hellow World! # screen == STDOUT, standard output == standard normal output   
[bobo@sh]  
## STDOUT -- file description 1  
hw.sh # Default output --> STDOUT -->screen  
hw.sh 1>logfile  
'>' rediect output to file  
  
# append the 'rediect' identifier to comamnd  
  
## STDERR -- file description 2, met some error, will output to STDERR, default location is also the screen, as default, 1 and 2 will both print on screen  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$(pwd;mkdir -p /root) (cmd1; cmd2) # put commands in a group, delimiter is ';'  
/oracle/report\_scripts # standard output/normal output  
mkdir: 0653-357 Cannot access directory /. # err output  
/: The file access permissions do not allow the specified action. # err output  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$  
  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$(pwd;mkdir -p /root) 1>normalout 2>errout  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$cat normalout   
/oracle/report\_scripts  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$cat errout   
mkdir: 0653-357 Cannot access directory /.  
/: The file access permissions do not allow the specified action.  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$  
  
Note:  
1. Be careful with Number and '>'  
N> # redirect  
N > # N--command or argument, '>' equals to '1>'  
1 >FILE  
'>>' # Append to file  
e.g.  
/path/to/script 1>/dev/null # 1>>/dev/null , equal  
/path/to/script 2>/dev/null  
/path/to/script 1>/dev/null 2>/dev/null  
/path/to/script 1>/dev/null 2>&1 # short for 2>STDOUT  
/path/to/script 1>&2 2>file # short for 2>STDOUT  
  
'&N' # stand for the file description Number &1 STDOUT, &2 STDERR  
&4 &5 ... # file description not opened. # exec 5 <> file  
  
/dev/null # special device , accept any input, nothing output  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$ls -l /dev/null  
crw-rw-rw- 1 root system 2, 2 Mar 13 15:46 /dev/null  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$ls -l /dev/zero  
crw-rw-rw- 1 root system 2, 3 Jul 23 2015 /dev/zero  
[oracle@MTG8\_OCSDB2\_1/oracle/report\_scripts]$

chmod +x hw.sh ## add the file to the PATH search list  
# AIX: chmod -x hw.sh, when we want to call 'hw.sh', canot find the file,  
# system only try to find the 'executable' file in PATH  
-rw-r--r-- 1 bobo dba 34 3月 13 15:25 hw.sh  
Basic: 10 digits  
- # file  
d # direcotry  
p # pipe file  
l # link   
owner|group user|others  
rwx|rwx|rwx  
w # writable 4   
r # read 2  
x # execute 1  
777 rwx|rwx|rwx  
When we crate a file, default permission # mask/umask   
[root@MTG8\_OCS1\_1/]#umask  
022 # (7-0)(7-2)(7-2) default folder permission   
drwxr-xr-x 2 root system 256 Mar 13 15:32 motest  
 # 644 default file permission, default not allow file to execute  
-rw-r--r-- 1 root system 0 Mar 13 15:30 motest

# 3 Foreground & Background

Next command need to wait -- run script foreground  
Run command in background ( Append '&' to the last )--- run in background  
[bobo@sh] sleep 100 &  
[1] 31836  
[bobo@sh]   
[bobo@sh] echo $!  
31836  
[bobo@sh] sleep 200 &  
[2] 31870  
[bobo@sh] echo $!   
31870  
[bobo@sh] jobs -l #detailed, include PID  
[2] + 31870 Running sleep 200 &  
  
1. Got the background process id, '$!' stands for the previous background process' PID  
2. Use 'jobs' command to view all bg process  
'fg %N' to bring background process to foreground  
[bobo@sh] fg %2  
sleep 200   
'kill %N' to kill background process  
[bobo@sh] jobs  
[2] + Running sleep 100 &  
[bobo@sh] kill %2   
[2] + Terminated sleep 100 &  
[bobo@sh]   
  
'Ctrl +Z ' to bring foreground to stopped at background  
^Z[2] + Stopped sleep 200 &  
[bobo@sh]   
'bg %N' to start background process, still in background  
[bobo@sh] bg %2  
[2] sleep 200 &  
[bobo@sh] jobs -l  
[2] + 31870 Running sleep 200 &  
[bobo@sh]   
  
3. Background process will exit after parent shell terminated.  
[bobo@sh] ps -ef|grep 31503  
bobo 31503 31493 0 16:01 pts/0 00:00:00 /usr/bin/ksh  
bobo 31966 31503 0 16:16 pts/0 00:00:00 /usr/bin/ksh # 31966 for sleep  
bobo 32022 31977 0 16:16 pts/1 00:00:00 grep --color=auto 31503  
[bobo@sh]   
[bobo@sh] ps -ef|grep 31503  
bobo 32063 31977 0 16:17 pts/1 00:00:00 grep --color=auto 31503 #parent shell terminated, all bg processeds exit  
[bobo@sh]   
  
nohup cmd '&' --> parent terminated, child keep running  
nohup # not work  
  
parent PID  
----child PID 1   
----child PID 2  
1--- init process  
--------child PID 1  
  
Note for nohup:  
1. Default output file nohup.out  
2. Usually we rediect to other files or /dev/null  
bobo 32173 31977 0 16:21 pts/1 00:00:00 sleep 100  
bobo 32173 1 0 16:21 ? 00:00:00 sleep 1000  
  
4. As default, the output of background process will be passed to the parent/current shell  
[bobo@sh] cat hw.sh  
i=1  
while [ $i -le 10 ]  
do  
 echo "output"  
 mkdir /root/bula  
 let i=i+1  
done  
  
[bobo@sh] hw.sh &i  
[1] 32335  
output  
/usr/bin/ksh: i: not found [没有那个文件或目录]  
mkdir: 无法创建目录"/root/bula": 权限不够  
output  
mkdir: 无法创建目录"/root/bula"[bobo@sh] : 权限不够  
output  
  
  
5. Use 'wait' to wait all background process exit  
[bobo@sh] time (sleep 5 &; sleep 10 & ; wait;echo OK)  
[1] 32450  
[2] 32451  
OK #show after 10 s  
  
real 0m10.05s  
user 0m0.00s  
sys 0m0.00s

# 4. Special variable & operator & statement

$! # previous backgorund process PID  
$? # return value of "previous" command  
# similar to C++ 'return'  
Sucess: 0  
Fail: non-zero, 1 2 255  
return to main process/current shell  
[bobo@sh] echo OK  
OK  
[bobo@sh] echo $?  
0  
[bobo@sh] mkdir -p /root/avc  
mkdir: 无法创建目录"/root": 权限不够  
[bobo@sh] echo $?   
1  
[bobo@sh]   
Note: The command to check return value must follow the previouse command, 'enter' not cound  
'echo $?' is also a command  
$1 $2 $3 $4 $5  
a b c d e  
var=$1 # var=a  
shift #   
$1 $2 $3 $4 # maximum location reduced from $5 ->$4  
b c d e # argument reduced from left  
var=$1 # var=b  
shift 2  
$1 $2 # maximum location reduced from $4 ->$2  
d e # argument reduced from left for 2   
var=$1 # var=d  
  
$0 argv[0] script file name  
$1 .. $N script parameter, correpdonding to it's location  
$$ script processs ID  
  
[bobo@sh] hw.sh a b c  
Script Name: hw.sh  
Script PID: 407  
First parameter: a  
Second parameter: b  
  
$\* parameter list as one string  
$@ parameter list as a list  
  
[bobo@sh] hw.sh a b c  
Script Name: hw.sh  
Script PID: 508  
parameter list: a b c  
parameter list: a b c  
  
\*\*shell parameter delimiter is space, and quoted string will be regarded as one string/parameter\*\*  
[bobo@sh] hw.sh a "b c"  
Script Name: hw.sh  
Script PID: 637  
parameter 1st: a  
parameter 2nd: b c

## 4.1 operator//used in test condition

man test # get detailed expression/operator  
# for Decimal   
a -le b # less or equal(e)  
a -gt b # greater than (t)  
-ge # greater than or equal  
-eq # equal  
-lt # less than  
-ne # not equal  
  
# for string  
"str1" == "str2" # equal, in bash, '='/'=='  
!= # not equal  
-n "$Str" # length not(n) zero  
-z "$Str" # length zero (z)  
  
# for file  
-x file # 1. file should exists. 2. the file should has executable permission  
-f file # file exists  
-d dire # directory exists  
-p pipefile # file is a pipefile  
  
'!' identify reverse the test result  
! -x # if the file has not executable permission, return 0  
! -d # if the directory not exists, return 0  
  
# for expression, variable  
+ - \*  
$((a+b)) # variable a + variable b  
[bobo@sh] echo $((9+8))  
17  
  
[bobo@sh] var=2  
[bobo@sh] echo $((var+8))  
10  
  
[bobo@sh] expr 1 + 2  
3

## 4.2 condition in statement//use return value

if cmd; then xxx ;fi

if [ 0 -eq 0 ];then  
 echo equal  
fi  
 [ 0 -eq 0 ] is a command, the pass condition is the command return value 0  
 [ 0 -eq 0 ] equals command 'test 0 -eq 0'  
[bobo@sh] test 0 -eq 0  
[bobo@sh] echo $?  
0  
[bobo@sh]   
[bobo@sh] if mkdir /root/abc 2>/dev/null;then  
> echo can  
> fi  
[bobo@sh]

### 4.2.1 Use true and false as condition

true/false equals as return value   
command true equals reutrn 0  
command false equals reutrn 1  
[bobo@sh] if true;then  
> echo OK  
> fi  
OK  
[bobo@sh]   
  
var=true  
if $var;then  
 echo OK  
fi  
var=false  
if $var;then  
 echo OK  
fi

## 4.3 statements

if .. ;then  
 do\_something;  
 do\_something  
fi  
if ..   
then  
 do\_something;  
 do\_something  
fi  
';' # end of line, by default, 'enter' equals ';'   
If we want to write several command in one line, we need join ';' with commands  
for x in a b c # list delimter is space, the list can be the output of some command  
for x in `ls`  
do  
 do\_somethind  
done  
  
while cmd # cmd return true  
while true  
while : # ':' equals to true  
do  
 do\_somethind  
done  
  
until cmd  
do  
 xxx  
done  
  
case "input" in  
1)  
 ;;  
2)  
 cmd;  
 cmd  
 ;;  
\*) # all rest  
 ;;  
esac  
  
# judge one time  
  
select variable in [a b .. ];  
do  
 do\_some\_thing  
done  
# interactive control  
# e.g. let user to choose option  
# if value not match, loop will continue, always, except for 'Ctrl+D' pressed  
# 'Ctrl+D' is universal key stroke to exit, exitshell/session/interactive prompt  
[bobo@download] select option in 1 2 3  
> do  
> echo "You choose option $option"  
> done  
1) 1  
2) 2  
3) 3  
#? 2  
You choose option 2  
# once select statement set, the variable is active.  
# similar to for loop; for var in xxx  
# until break

## 4.1 flow control//loop control

case ... esac, no need to control, it only match one time  
for ... do ... done #   
1. continue -->stop execute current  
for xxx  
do  
 continue #  
 echo will not execute  
done  
2. break --> equals break 1 -->break one loop layer, break N  
while xx  
do  
 while xx  
 do  
 break   
 break 2  
 done  
 echo outer loop  
done  
  
  
## script.sh -cp xx -mv xx  
while [ -n "$1" ]  
do  
case "$1" in  
-cp)  
 shift # equals to shift 1, argument number will be reduced 1  
 ;;  
-mv)  
...  
esac

# 5.Most common commands:

sort/grep/cut/date/ps/expr

# 6.PIPE，HereDoc，STDIN/STDOUT，redirect，operator，special file(fifo,link)

# 7.Function, arguments

# 8.shell statements(while,for,if..)

# 9.awk basis(variable，string，statements,pattern&action）

# 10.sed basis(AIX)

# 11.Regex/Glob/Wildcard

# 12.Different between AIX/Linux commands

# 13.Debug shell script