1.The basis of *nix and shell

1.1 shell

```
system level -- connector -- user
core part --- command line
run command in command line --->shell 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 variables:   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 executable 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 # internal 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 regarged 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 continue, '>' --> PS2
[bobo@download] export PS2="#"
[bobo@download] echo "a"b"c
#


fmt: define variable with multiple 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 variable, 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 one-line
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
```

1.7 File Time & Permission

```
stat file
Acess time   # ls -lur
Modify time  # ls -ltr
Change time  #

mv file  # need modify permission for directory.
w  -- # modify on file/directory itself
```

2.Difference 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}
```

2. 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/du/df/comm

5.1 sort

5.1.1 basic sort rule

```
# *nix, input field seprator : IFS , default separator is continuous "space"(regared as one)
# by default , sort from the first column to the last column
# for two line, if Nst column is same, compare N+1 column
# ordered by ascii code(decimal), this is most common rule.
[bobo@sh] sort -n a.txt  # GNU linux, by default , --human-numeric-sort, differs with UNIX
_
0
a
aa
d
ocs@MTG8_OCS2_2:[/ocs/scripts]$sort sort.txt

0    # 48
_    # 95
a   # 97
aa  # 1st char 'a' same, compare '' with 'a', '' <'a'
d
# man ascii, check the decimal value
```

5.1.2 sort option

```
# sort -u   # sort and uniq, also supported in AIX
# equals xxx|sort|uniq
[bobo@sh] sort a.txt


_
0
a
a
aa
d
[bobo@sh] sort a.txt|uniq

_
0
a
aa
d

[bobo@sh] sort -u a.txt

_
0
a
aa
d

# sort -n   # numeric sort, compare number by decimal value instead of the ascii code
[bobo@sh] sort -n  a.txt


_
0
aa
d
2
10
[bobo@sh] sort  a.txt


_
0
10
2
aa
d

# sort -t "char"  # specify the 'char' as separator, usually used together with -k
[bobo@sh] sort a.txt
0,2,2
1,1,3
[bobo@sh] sort -t "," -k2,2 a.txt
1,1,3
```

```
0,2,2
[bobo@

# sort -o output_file # can output to the file itself
[bobo@sh] cat a.txt
0,2,2
1,1,3
[bobo@sh] sort -t "," -k2,2 -o b.txt a.txt
[bobo@sh] cat b.txt
1,1,3
0,2,2
[bobo@sh] cat a.txt
1,1,3
0,2,2
```

5.1.3 sort by column (sort -k)

```
sort -k [start_column.start_position],[end_column.end_position]
# usually only used column
# for each column , by default, sort from first position(1)/char to the last
[bobo@sh] sort a.txt
a1a 2 2
aa1 1 3
[bobo@sh] sort -k1.3,1.3 a.txt
aa1 1 3
a1a 2 2
[bobo@sh] sort a.txt
a1a a11 3
a2a ab2 2

[bobo@sh] sort -k1.3,2.2 a.txt  # sort by "a a1" and "a ab"
a1a a11 3
a2a ab2 2
[bobo@sh] cat a.txt
a2a ab2 2
a1a a11 3

# if not specify the end_position, by default, will compare the whole string

# -k start,end only specify the priority of sort column, sort command will still try to sort
until got a result
[bobo@sh] cat a.txt
a a b
a a 1

[bobo@sh] sort -k1,2 a.txt  # 1st & 2nd same, though we only specify "1,2", still sort 3rd
a a 1
a a b
```

5.2 grep

5.2.1 basic rule

```
# grep used BRE  (basic regular expression)
.  any char
*  the previous char recurr any time
\  escape '.' '*' to raw char
[bobo@sh] ls|grep ".xt"
a.txt
b.txt
sort.txt
[bobo@sh] ls|grep "\.xt"
[bobo@sh]

[bobo@sh] ls|grep ".sh"
ash
hw.sh  #both 'a' and '.' will match pattern '.sh'
[bobo@sh] ls|grep "\.sh"
hw.sh
```

5.2.2 options

```
# grep -E # equals 'egrep' command
# E ERP --extended regular expression, support much more patterns
a|b  a or b
{N,M}  the previous char recurrs N-M times
{N,}  the previous char recurrs more than N times
(xx)  treat as one string  . e.g. (a|b)c  match ac and bc
# no supported
\d


[bobo@sh] ls|grep -E "a{1,}sh"
aash
ash
[bobo@sh] ls|grep -E "a{2,}sh"
aash

[bobo@sh] ls|grep -E "a|b"
aash
ash
a.txt
bsh
b.txt

[bobo@sh] ls|grep -E "(a|b)sh"
aash
ash
bsh

# grep -F   # treat the pattern as raw string, instead of regular expression
       -F, --fixed-strings
              Interpret  PATTERN  as  a  list  of  fixed  strings (instead of
              regular expressions),
touch a*b
[bobo@sh] ls|grep "a*b"  # match 'b', 'ab', 'aa...b'
a*b
bsh
b.txt
[bobo@sh] ls|grep -F "a*b"
a*b

# grep -w   match word
[bobo@sh] ls|grep -w "ash"
ash  # aash excluded

# grep -c char  # count, can replace  `grep char|wc -l`
[bobo@sh] ls|grep ".txt"
a.txt
b.txt
sort.txt
[bobo@sh] ls|grep ".txt"|wc -l
3
[bobo@sh] ls|grep -c ".txt"
3
```

```
# grep -i  #ignore case
[bobo@sh] ls|grep "Ab"
Ab
[bobo@sh] ls|grep -i "Ab"
Ab
AB
[bobo@sh]
```

5.2.3 extra tips

```
# for '-' and '--', if any pattern starts with '-' or '--', need back slash to escape   ,
otherwise, grep will treat as option
[bobo@sh] echo > -a
[bobo@sh] ls|grep "\-a"
-a
```

5.3 cut

```
# cut -c start-end   # get start pos to end pos of a line
[bobo@sh] cat a.txt
a123 b c
[bobo@sh] cat a.txt|cut -c 4-9
3 b c
[bobo@sh] cat a.txt|cut -c 2-3
12
[bobo@sh] echo 20170302|cut -c 1-4
2017
[bobo@sh] echo 20170302|cut -c 5-6
03

# cut -d "separator" -f field_start-field_end
# get column value, default separaror/ IFS is one "space", separator can only specify one char
[bobo@sh] cat a.txt
a123 b c
[bobo@sh] cat a.txt|cut -d " " -f2  # column/field sequence number
b
[bobo@sh] cat a.txt|cut -d " " -f2-  # '2-' equals '2-last'
b c
[bobo@sh] cat a.txt|cut -d " " -f2-3
b c
[bobo@sh] cat a.txt|cut -d " " -f1-
a123 b c
[bobo@sh] cat a.txt|cut -d"," -f 2 a.txt
b
[bobo@sh] cat a.txt
a123,b,c,,
[bobo@sh]
```

5.4 date

```
date +%fmt_keyword  # %fmt_keyword can concact with string or other fmt_keyword, space should be escaped
%Y   # year in yyyy format
%m  # month in mm format
%d   # day in dd format
man date # check more options
%H  # hour in 24 hour format, left padding with zero, 03
%M  # minute  left padding with zero, 09
%S  # seconds left padding with zero, 09
[bobo@sh] date +"%Y abc %m"
2017 abc 03
[bobo@sh] date +%Y\ abc\ %m  # must escape space
2017 abc 03
[bobo@sh] date +%Y_abc_%m
2017_abc_03

date +%Y-%m-%d_%H:%M:%S
2017-03-30_11:14:21
```

5.5 ps

```
# ps aux  # ps '-aux' is different with 'ps aux', ps '-ef' also differs with 'ps ef'


# ps -ef
-e   #show all processes , qeual to -A
-f  # full format

# ps -eo "args,cmd"
-o "fmt_keyword1,fmt_keyword2" # specify output format
ps -eo "args,cmd"  # only list the arguments and commands, 'args' 'cmd' are the long keywords
of the builtin specification of ps, check manual page to view more
ps -eo "cmd,etime="
[bobo@sh] ps -eo "cmd,etime"
CMD                             ELAPSED
/sbin/init nosplash noquiet    02:14:46
[kthreadd]                     02:14:46

# elapsed time format, day_hour:min:se
# '=' follows the keyword, to suppress the  title
[bobo@sh] ps -eo "cmd,etime="|head
CMD
/sbin/init nosplash noquiet    02:16:02
[kthreadd]                     02:16:02
[bobo@sh] ps -eo "cmd=,etime="|head
/sbin/init nosplash noquiet    02:16:13

# ps -p pid
[bobo@sh] ps -p 3056
  PID TTY          TIME CMD
 3056 ?        00:00:00 evolution-addre
```

5.6 expr

```
# to manuplate the string, or calculate
[bobo@sh] expr 1 + 2   # fmt:  expr op_value (space needed) operator  (space needed) op_value
3
[bobo@sh] expr 1 \* 2
2
[bobo@sh] expr 4 / 2
2

# substr/match/index/length   # fmt: expr substr/match  "input_string" start-Pos  length,
similar to oracle/awk substr
[bobo@sh] expr substr "abced"  2 3
bce
# match use BRE
    match STRING REGEXP
              same as STRING : REGEXP
       substr STRING POS LENGTH
              substring of STRING, POS counted from 1

       index STRING CHARS
              index in STRING where any CHARS is found, or 0

       length STRING
              length of STRING

[bobo@sh] expr match "1abcd" ".[a-z]*bc.*" # match pattern with whole string, return the
length
5
[bobo@sh] expr index "1abcd" "ab"         # start position when 'ab' appears in '1abcd'
2
[bobo@sh] expr length "1abcd"
5

# another format to get substring, fmt : exprt "input_string" (space) : (space) "regular
expression with group", output result is the groups
[bobo@sh] expr "1abcd" : ".*\(ab\).*"
ab

(xx)  # stands for groups in regexp, in expr, need to escape
[bobo@sh] expr "1ab2cd" : ".*\([0-9]\).*"  # greedy match '.*' matches '1ab'
2

greedy match  # match as more as possible
minimum match  # match as more as possible
[bobo@sh] echo "c1ab2cd"|sed 's/\(.*\)\([0-9]\)\(.*\)/\1f\3/1'
c1abfcd
```

5.7 split

```
# split -N file  / split -l N file # split the file to parts, each part limit to lines,  part
name : x[aa-zz]
split -50 sp.txt  # got sp.txt(orig) , xaa, xab,xac
[bobo@sh] wc -l x* sp.txt
  50 xaa
  50 xab
  29 xac
 129 sp.txt
# split by lines, line content will not be truncated


# split by size, k, m, default-bytes
split -b N(unit) file # split file to parts, each part's size limited to maximum N bytes
split -b 400k sp.txt
[bobo@sh] du -sk x* sp.txt
400 xaa
400 xab
284 xac
1084    sp.txt

# potential risk: the text line may be truncated
/media/Data/Software/runtime_bin/jdk1.6.0_45_64bit/db/javadoc/jdbc3/org/apache/derby/vti/packa
ge-tree.html
/media/Data/Software/runtime_bin[bobo@sh]  # last line truncated


# specify the output filename
# fmt: split opion orig_filename output_filename_prefix

[bobo@sh] split -b 400k sp.txt out_
[bobo@sh] ls -l out_*
-rw-r--r-- 1 bobo dba 409600 3月  30 13:32 out_aa
-rw-r--r-- 1 bobo dba 409600 3月  30 13:32 out_ab
-rw-r--r-- 1 bobo dba 290816 3月  30 13:32 out_ac
[bobo@sh] split -b 400k sp.txt sp.txt
[bobo@sh] ls -l sp.txt*
-rw-r--r-- 1 bobo dba 1110016 3月  30 11:52 sp.txt
-rw-r--r-- 1 bobo dba  409600 3月  30 13:33 sp.txtaa
-rw-r--r-- 1 bobo dba  409600 3月  30 13:33 sp.txtab
-rw-r--r-- 1 bobo dba  290816 3月  30 13:33 sp.txtac
[bobo@sh]
```

5.8 diff/sdiff/comm (compare)

```
# diff a b
# sdiff a b   # compare two file , each compare result shown in one line
[bobo@sh] cat a.txt
bobo  29
ca  20
same line
dbc 21
[bobo@sh] cat b.txt
bobo male
same line
dbc female
[bobo@sh] diff a.txt b.txt
1,2c1
< bobo  29
< ca  20
---
> bobo male
==========================#
fmt: (from,to line in file 1)   result(c/..)  (from,to line in file 2)
1,2c1
content  file 1
---
content file 2
==========================#
4c3
< dbc 21
---
> dbc female

'a' adding
'c' change
'd' deleted, if deleted, file will be same
'>'  lines from file two
'<'  lines from file 1
< - denotes lines in file1.txt
> - denotes lines in file2.txt

# --------------------------
[bobo@sh] sdiff a.txt b.txt
bobo  29                                | bobo male  # '|' different line
ca  20                                  <            # '<' left exists , right not exist
same line                              same line
dbc 21                                  | dbc female
[bobo@sh]

# test delete one line
[bobo@sh] cat c.txt
bobo  29
ca  20
dbc 21
[bobo@sh] cat a.txt
bobo  29

ca  20
```

```
same line
dbc 21

[bobo@sh] diff a.txt c.txt
3d2 #     # if delete the 3rd lines in the  file 1, or put this line after the 2nd of the file 2, two files will be same
< same line

[bobo@sh] head a.txt c.txt
==> a.txt <==
bobo  29
ca  20
same line
dbc 21

==> c.txt <==
bobo  29
same line
dbc 21
[bobo@sh] diff a.txt c.txt
2d1
< ca  20
[bobo@sh]
# use file 2 for reference
[bobo@sh] diff c.txt a.txt   # if add the content after 1st line of file 1, or delete in the 2nd of file 2, two file will be the same
1a2
> ca  20
[bobo@sh]

# changes
[bobo@sh] diff a.txt c.txt  # show the changed line
1,2c1,2
< bobo  39
< ca  20
---
> bobo  29
> ca  22

# both change and delete will be regarded as change
[bobo@sh] diff a.txt c.txt
1c1,2      # change the 1 line of file 1 as <the content of file 2>, or change the 1,2 lines of file 2 with <the content of file 1>, two file will be the same
< bobo  39   # <the content of file 1>
---
> bobo  29
> ca  22   # <the content of file 2>
[bobo@sh] cat a.txt
bobo  39
same line
dbc 21
[bobo@sh] cat c.txt

bobo  29
```

```
ca  22
same line
dbc 21
[bobo@sh]
```

5.8.1 generate simple patch, apply patch

```
diff -c /diff -u file1 file2  > patch_apply_on_file1
patch <patch_apply_on_file1   # apply patch on file1

[bobo@sh] cat a.txt
bobo  29
ca  22
same line
dbc 21
[bobo@sh] cat b.txt
bobo male
same line
dbc female
[bobo@sh] diff -c a.txt b.txt
*** a.txt    2017-03-30 13:54:42.384234581 +0700
--- b.txt    2017-03-30 13:35:56.968254735 +0700
***************
*** 1,4 ****
! bobo  29
! ca  22
  same line
! dbc 21
--- 1,3 ----
! bobo male
  same line
! dbc female
[bobo@sh] diff -c a.txt b.txt >patch_apply_to_a
[bobo@sh] patch <patch_apply_to_a # a will be the same with b
patching file a.txt
[bobo@sh] cat a.txt
bobo male
same line
dbc female
[bobo@sh] cat b.txt
bobo male
same line
dbc female
[bobo@sh]

# e.g update any version of script
[bobo@sh] cat hw.sh
echo "Script Name: $0"
echo "Last parameter: ${@:$#} "
[bobo@sh] cat hw.update.sh
echo "Script Name: $0"
echo "parameter 1st: $1"
echo "end"
[bobo@sh]
diff -c hw.sh hw.update.sh >patch_to_orig_hw.sh_for_rbk
# update hw.sh to hw.update.sh
mv hw.sh hw.sh.orig
mv hw.update.sh hw.sh
# rollback

[bobo@sh] cat hw.sh
```

```
echo "Script Name: $0"
echo "parameter 1st: $1"
echo "end"
[bobo@sh] patch <patch_to_orig_hw.sh_for_rbk
patching file hw.sh
Reversed (or previously applied) patch detected!  Assume -R? [n] y
[bobo@sh] cat hw.sh
echo "Script Name: $0"
echo "Last parameter: ${@:$#} "
[bobo@sh]

# patch -lN <patch_file
--/path/to/file

# comm -[123] file1 file2, apply on ordered files, otherwise , warn
# comm command regarding one file into structures as blow:
uniq content in file1
            same content
                   uniq content in file2
-1  suppress the uniq content in file1
-2  suppress the uniq content in file2
-3  suppress the same content both in file1 and file2
# check the same line
comm -12 a.txt b.txt

[bobo@sh] comm -12 a.txt b.txt
same line
# check only the difference
[bobo@sh] comm -3 a.txt b.txt
bobo  29
    bobo male  # indent to show content only exists in file2
ca  22
dbc 21
    dbc female
[bobo@sh]
```

5.9 tar/gzip (compress)/zip

```
# compress /uncompress is AIX command, compress file to file.Z
# tar will not compress, just put files together in a solid file, original files still exists
tar -cf test.tar test    # '-cf' equals 'cf' , '-' is not essential
# fmt:  tar [options] tarball_name original_files/folder_list ...
tar -cf test.tar test (folder) davmail.log (file) ...
c  create
f  use file as compress target
v  verbose, print log on screen
x uncompres
# list the content in tarball without uncompress
t  list
tar -tf file.tar

[bobo@test] tar -tf file.tar|head
./
./tmp/
./tmp/2_def_a
./tmp/cc
./tmp/bc_1
./sql/
./sql/test_elsif.sql
./sql/test_no_data_found.sql


## uncompress only one file, by default, it will uncompress to the original path, paths will
be created recursively
tar -xf file.tar  /path/to/file/in/tarball
[bobo@test] ls -l
总用量 35416
drwxr-xr-x 15 bobo dba     4096 3月  30 14:17 bak
-rw-r--r--  1 bobo dba 36259840 3月  30 14:14 file.tar
[bobo@test]
[bobo@test] tar -xf file.tar ./sql/test_complex_statment.sql  # can add more paths
[bobo@test] ls -l
总用量 35420
drwxr-xr-x 15 bobo dba     4096 3月  30 14:17 bak
-rw-r--r--  1 bobo dba 36259840 3月  30 14:14 file.tar
drwxr-xr-x  2 bobo dba     4096 3月  30 14:19 sql   # folders auto-created
[bobo@test]
[bobo@test] find sql
sql
sql/test_complex_statment.sql
[bobo@test]


# gzip  file  # output to file.gz, original files deleted
gunzip  / gzip -d   file.tar  # uncompress , original file.gz deleted
# check the content list
gzip -l / gunzip -l

[bobo@test] gzip -l file.tar.gz
         compressed        uncompressed  ratio uncompressed_name
            2355547            36259840  93.5% file.tar
```

```
[bobo@test]

# zip
# usage: zip [a/x] zip_name  orig_file
[bobo@test] zip a file.tar.z file.tar
    zip warning: name not matched: file.tar.z
  adding: file.tar (deflated 94%)
[bobo@test]

# unzip file
## check content
unzip -l file.z
[bobo@test] unzip -l a.zip
Archive:  a.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
 36259840  2017-03-30 14:14   file.tar
---------                     -------
 36259840                     1 file
[bobo@test]
```

5.10 du/df/etc

```
# df -g   # unit GB
# df -m  # unit MB
# df -h  #human reable, only linux

# du  or du dir #By default, it will crawl/count all subs- directories,
124 ./shell/modules
20  ./shell/split_file_by_page
284 ./shell
44692   .

# du -s dir # only count the folder, will not go deep into sub folder
[bobo@test] du -s .
44692   .
[bobo@test] du -s shell
284 shell

#  count all current listed sub directories , but not go into sub-sub diretories
[bobo@test] ls|xargs du -s
68  algorithm
7196    backup
56  cpp
28  graphviz
72  java
24  oracle
27100   perl
2364    python
32  sh
284 shell
28  sql
4   tmp
16  topic

# paste # paste two or more files, just like join in oracle, join by line number, concact
context in the second/... file to the previous file
[bobo@sh] cat a.txt
bobo  29
abc 21
[bobo@sh] cat b.txt
male
female
[bobo@sh] cat c.txt
100
200
[bobo@sh] paste a.txt b.txt c.txt
bobo  29    male    100
abc 21  female  200

# join # concact two sorted files, join by the first column, if match ,then concact the
content from the second to the last columns
[bobo@sh] join a.txt b.txt
bobo 29  male
dbc 21 female

[bobo@sh] cat a.txt
```

```
bobo  29
ca  20   # different, this line will be omitted
dbc 21
[bobo@sh] cat b.txt
bobo male
dbc female
[bobo@sh]

# tr "string list"  "translate list"  <innput >output # translate
[bobo@sh] echo abc|tr "a" "A"  # input is stdin, output stdout
Abc
[bobo@sh] cat a.txt
a 2
a 3
b 6
b 5
[bobo@sh] tr "a" "A" <a.txt
A 2
A 3
b 6
b 5
[bobo@sh]
[bobo@sh] tr "ab" "bA" <a.txt   # translate 'a' to 'b', 'b' to'A', similar to oracle
translate() , original a.txt not change, just output to screen
b 2
b 3
A 6
A 5
[bobo@sh]
```

6.PIPE，HereDoc，STDIN/STDOUT，redirect，operator，special file(fifo,link), send option to command

```
# '|'
cmd1| cmd2|cmd3
# all commands run at same time, each cmd have STDOUT/STDERR
cmd1 2>&1 |cmd 2>/dev/null|cmd 3

# heredoc/inline input
cmd  << finish_iden
...
finish_iden   # any string, just start with string, and finish with end of line

[bobo@bobo] cat <<mose
> a
> b
>  mose
> mose
> mose
a
b
 mose
mose # trailing space
[bobo@bobo]


cat <<EOF
line 1
line 2
EOF
# output
line 1
line 2


# content still workds in heredoc
`cmd`
$(cmd)
$VAR

[bobo@bobo] cat <<abc
> echo $HOME
> abc
echo /home/bobo
[bobo@bobo]

[bobo@bobo] cat <<!
> `date
> `
> !
2017年 03月 30日 星期四 14:30:44 +07
[bobo@bobo]

# STDIN/STDOUT
cat < file    #file acts as stdin

[bobo@sh] cat <a.txt
```

```
bobo  29
ca  22
same line
dbc 21

[bobo@sh] echo abc|cat -  # '-' stands for the stdin/stdout, it can only be used one time
abc
[bobo@sh]

[bobo@sh] tar -cf - *.sh |gzip >file.gz

# '-' stands for the stdout, tar generate the xx.tar to stdout, '|', stdout in the left of
pipe, equals stdin in the right side of the pipe, then gzip read from stdin

# fifo
fifo - first in first out, pipe file
[bobo@sh] ls -l ff
prw-r--r-- 1 bobo dba 0 3月  30 14:39 ff
[bobo@sh]
mkfifo  file # create to pipe file
cmd1 | cmd2 --->
cmd1 >ff
cmd2 <ff
# example
mkfifo /path/PIPE   # PIPE is a file
chmod 777 /path/PIPE
tar -cf PIPE  folder  &  # run in bg, output tar data fo PIPE
compress <PIPE  > targetfile.tar.Z   # PIPE will always get input from tar, output to
compress, untill tar finished, or compress finished
# equals to
tar -cf targetfile.tar folder
compress targetfile.tar
# advantage
1. fifo will not occupy disk space
prw-r--r-- 1 bobo dba 0 3月  30 14:39 ff
2. some command cannot connect by '|' (cannot read from stdin), and we want to connet these
two commands -- to run at same time

# redirect
>/>>   &N  file description
&1  &2, file description from 1 to N
>/dev/null 2>&1
2 -- stderr
# others are hidden, normally not open
# if we want to redirect to more target, except for 1(stdout) 2(stderr)
[bobo@sh] mkfifo ff
[bobo@sh] exec 5<>ff   # open the channel for NO.5
[bobo@sh] echo abc >&5  # output to channel No.5
[bobo@sh] cat <&5   # read from channel No.5
exec 5<&-  # close
# these channel means output target, but the target is not normal file


# use pipe to send option to command
```

```
(sub command; sub command)| cmd  # cmd canread from stdin
(echo y)|ostool send PNO 65006
echo y|ostool send PNO 65006

# telnet to svr, non-interactive
(sleep 2; echo usr #username
; sleep 2; echo pwd #passwd
; sleep 2; echo command) |telnet ip

# sftp /rm .. some command cannot read from stdin
sftp -b - # can read from stdin

# special files
ln -s b.txt b.link  # soft link
ln b.txt c.link  # hard link , is the same file to the original file
[bobo@sh] ll b.link c.link
-rw-r--r-- 2 bobo dba 31 3月  30 13:56 c.link
lrwxrwxrwx 1 bobo dba  5 3月  30 14:58 b.link -> b.txt

[bobo@sh] ls -li *.link b.txt
27921550 lrwxrwxrwx 1 bobo dba  5 3月  30 14:58 b.link -> b.txt #link file
27921553 -rw-r--r-- 2 bobo dba 31 3月  30 13:56 b.txt
27921553 -rw-r--r-- 2 bobo dba 31 3月  30 13:56 c.link  # c.link and b.txt is the same file,
occupy one piece on hard disk, if we delete one name, another name exists, two file same and
equals
cp b.txt d.link  # create different file with same content

27921553 -rw-r--r-- 2 bobo dba 31 3月  30 13:56 b.txt
27921553 -rw-r--r-- 2 bobo dba 31 3月  30 13:56 c.link  # hard link, inode is the same with
b.txt
27921558 -rw-r--r-- 1 bobo dba 31 3月  30 15:03 d.link
```

7.Function, arguments, alias, return,exit

```
# define
foo(){

}

function foo {  # func name follows with a space  # bash/ksh compatible, act different in ksh

}
foo
`foo`

[bobo@sh] foo(){
> echo $0
> echo hw
> }
[bobo@sh] function bar {
> echo $0
> echo bar
> }
[bobo@sh] echo `foo`
/usr/bin/ksh hw

[bobo@sh] foo
/usr/bin/ksh
hw

[bobo@sh] bar
bar   # $0 in ksh function , stands for function name
bar

# function output vs return value
output : result of all 'echo/xx' command output
return value:  execute success or not (0, non-zero) of the last command in the function

foo   # 1. get output  2. get the return value of the last command
[bobo@sh] foo
/usr/bin/ksh
hw
[bobo@sh] echo $?
0
[bobo@sh] foo(){
> echo abc
> mkdir -p /root/abc 2>/dev/null  # return value : 1, failed
> }
[bobo@sh] foo
abc
[bobo@sh] echo $?
1

# obvious return value
return [N]  # anywhere return, function end, N: default 0
[bobo@sh] foo(){
> echo abc
```

```
> return
> echo def  # function finished, will not run
> }
[bobo@sh] foo
abc
[bobo@sh]

[bobo@sh] foo(){
> echo OK
> return 2
> }
[bobo@sh] foo
OK
[bobo@sh] echo $?
2
[bobo@sh]

# arguments, same with script
foo $1 $2 $3
# $0 stands for script name(bash/ksh..) or function name(ksh function xx {..})
[bobo@sh] foo(){
> echo "first argument" $1
> echo "argument count" $#
> }
[bobo@sh] foo a b c
first argument a
argument count 3
[bobo@sh]

## exit  [N] exit the script, specify the exit code
[bobo@sh] cat foo.sh
echo abc
exit 10

[bobo@sh] chmod +x foo.sh
[bobo@sh] foo.sh
abc
[bobo@sh] echo $?
10
[bobo@sh]

[bobo@sh] cat foo.sh
foo(){
    echo func
    return 2
}
foo  # return, function finished
echo "return of func" $?
echo abc
exit 10
echo "return of script" $?  # script exited, will not run


[bobo@sh] foo.sh
```

```
func
return of func 2
abc
[bobo@sh]

# alias, not used too much in script, not recommended to use in script
alias name='command string'
alias name='foo(){ cmd $1; }; foo'
```

8.String manipulation, xargs, eval

```
# operate on text/string
cut  # get substring
expr  # get substring
echo "string"|sed/awk/cut ...  # operate on string
var="string"
echo ${#var}  # fastest way to get the length of string
basename /path/fo/file.sh .sh    # basename path_to_file remove_suffix

[bobo@sh] basename /home/bobo/test/sh/hw.sh
hw.sh
[bobo@sh] basename /home/bobo/test/sh/hw.sh .sh
hw
[bobo@sh]

# variable expand to get string
# '#' if substring matches the pattern from beginning of the string , remove it
# '%' if substring matches the pattern from the end of the string , remove it
# '##'/'%%' maximum match

[bobo@sh] name=/home/bobo/test/sh/hw.sh
# get path
[bobo@sh] echo ${name%/*.sh} # matched '/hw.sh', remove it
/home/bobo/test/sh
# get name
[bobo@sh] echo ${name##*/}  # matched '/home/bobo/test/sh/', remove it
hw.sh

# xargs,   transfer input lines to one line
xargs # transfer all input lines to one line
[bobo@sh] echo "A
> b
> c"|xargs
A b c

xargs -n N  # transfer all input lines to line, each output line contains N lines of input
[bobo@sh] echo "A^Jb^Jc"|xargs -n2
A b
c

rm *  ---> # sometimes, OS litmited the maximum param numbers/param  length, we need to use
xargs
ls|xargs rm
find . -type f -exec -rm {} \;  # may meet exception when param too long -->
find . -type f |xargs rm

xargs -I {} echo {} # fmt: xargs -I rep_str  command, 'echo {} ' is the command
[bobo@sh] ls|xargs -I {} echo "content: "{}
content: 5
content: -5
content: =a
content: -a
[bobo@sh] ls|xargs -I abc echo "content: "abc  # here 'abc' is the replace string keyword
content: 5
```

```
content: -5
content: =a
content: -a

# eval  # translate the shell comands for the second time
[bobo@sh] echo $HOME  # shell translate $HOME to /home/bobo
/home/bobo
[bobo@sh] echo \$HOME  # shell translate '\$' to raw token '$'
$HOME
[bobo@sh] eval echo \$HOME  # 1. shell translate '\$' to raw token '$' 2. eval translate for
the second time, translate '$HOME' to '/home/bobo
/home/bobo

[bobo@sh] var=a
[bobo@sh] a_1=b
[bobo@sh] eval echo \$${var}_1 # 1. shell translate '\$' to raw token '$', translate '${var}'
to 'a', command as 'eval echo $a_1' 2. eval translate '$a_1' to value b
b
[bobo@sh]
```

9.awk basis(AIX-based,variable，string，statements,pattern&action）

```
# awk language, gawk   GNU awk, only shipped with linux
# awk process line by line, treat continuous space as one
# $1 stands for the column values, $NF stands for the last column value, $(NF-1) for the column
before last column, $0 stands for the whole line
[bobo@sh] awk '{print $NF}' a.txt
29
22
line
21

# string in awk , need to be quote with "string", otherwise, awk treat as column/variable
# variables (include $0) concact directly
awk '{a="content ";print a$0 }' a.txt
[bobo@sh] awk '{a="content ";print a$0 }' a.txt  # a-->variable a, concact with $0
content bobo  29
content ca  22
# print equals to 'print $0'
[bobo@sh] awk '{print}' a.txt
bobo  29
ca  22


[bobo@sh] awk 'BEGIN{print "Start"}{print}END{print "end"}' a.txt
Start
bobo  29
ca  22
same line
dbc 21
end

# form 1
awk 'BEGIN{cmd}   # comands executed before processing the first line
{
  if(xxx) {xxx};  # if($1=="xx") / if ($2~ /pattern/) '~' like /pattern/ is regexp
  else if(xxx) {xxx};  # commands end with ';',  or new line
  for(i=0;i<10;i++) {xxx}
  ar[0]=1
  ar[1]=2
  for( x in ar) {print ar[x]};
  substr/instr/length/match/split/index/.....


}
END{cmd}  # comands executed after processed the last line
'
# form 2
awk '/pattern/{action};/pattern/{action}'
# equals
awk '{
if(/pattern/){action};
if(/pattern/){action};
}'
```

```
# form 3
awk -f file.awk  # write commands to file
[bobo@sh] cat file.awk
BEGIN{print "Start"}{print}END{print "end"}
[bobo@sh] awk -f file.awk a.txt
Start
bobo  29
ca  22
same line
dbc 21
end

# ++
[bobo@sh] cat a.txt
a 2
a 3
b 6
b 5
[bobo@sh] awk '{rec[$1]+=$2}END{for(x in rec){print x" "rec[x]}}' a.txt
a 5
b 11

# rec[$1]+=$2   # define array named 'rec', rec[a]=2, rec[a]+=3, i.e.   rec[a]=rec[a]+3,
rec[b]=6, rec[b]+=6 -->rec[b]=11
# array --> similar to a dict {key:value,key1:value1}

# get environment variable
[bobo@sh] echo abc|awk '{print ENVIRON["HOME"]}'  # environ["variable_name"]
/home/bobo
[bobo@sh] echo abc|awk '{print "'$HOME'"}'
/home/bobo

# get host command output
[bobo@sh] echo abc|awk '{system("ls")}'   #sytem("command")
-5  -a  a*b   ash  b.link  b.txt  d.link  file.gz   hw.sh   patch_apply_to_a
patch_to_orig_hw.sh_for_rbk
# get one line result of host command
[bobo@sh] echo abc|awk '{"ls"|getline a;print a}'  # "command"|getline var , put the command
output(one line) to var
5


# variables  '-v'
[bobo@sh] echo |awk -v var=name -v var2=name2 '{print var, var2}'
name name2

# special variable in awk, must define in BEGIN block
FS   # field spliter, default value 'continuous space'
RS  # Record spliter, default value '\n'
OFS  # output field spliter, default value equals FS
ORS # output record spliter, default value equals RS
[bobo@sh] echo a,b |awk 'BEGIN{FS=","}{print $2}'
b
```

```
# equals to '-F'  to specify spliter char/string
[bobo@sh] echo a,b |awk -F"," '{print $2}'
b
[bobo@sh] echo a,,b |awk -F",," '{print $2}'
b
# change output spliter, the column must be modified, so here we use '$1=$1'
[bobo@sh] echo a,,b |awk -F",," 'BEGIN{OFS="+"}{$1=$1;print $1,$2 }'
a+b
[bobo@sh] echo a,,b |awk -F",," 'BEGIN{OFS="+"}{print $1,$2 }'
a+b
[bobo@sh] echo "a,b
c,d"|awk -F"," 'BEGIN{ORS="--"}{print $0}'
a,b--c,d--[bobo@sh]
[bobo@sh] echo "a,b^Jc,d"|awk -F"," 'BEGIN{}{print $0}'
a,b
c,d
```

10.sed basis(AIX-base)

```
# replace string
s/pattern_need_to_matched_and_be_replaced/replace_str/
s/pattern_need_to_matched_and_be_replaced/replace_str/[N,g]  # N>=1 the Nth occurence, g --
replace all occurences
[bobo@sh] echo abcb|sed 's/b/d/1'
adcb
[bobo@sh] echo abcb|sed 's/b/d/2'  # replace the 2nd occurence
abcd
[bobo@sh] echo abcb|sed 's/b/d/g'  # replace all occurences
adcd

# sed "s/str/rep/g" or sed 's/str/rep/g', if want to use shell variable, use ""
[bobo@sh] echo abcb|sed "s/b/$LOGNAME/g"
abobocbobo
[bobo@sh] echo abcb|sed 's/b/$LOGNAME/g'
a$LOGNAMEc$LOGNAME
[bobo@sh]

[bobo@sh] echo abcb|sed "s#b#$HOME#g"  # when the replace string contains '/', conflict with
's/pat/str/g'
a/home/boboc/home/bobo
[bobo@sh]

# match pattern '/pattrn/[action]'  # action include 's/pat/str/g'
[bobo@sh] ls|sed -n '/.sh$/p'   # 'p' to print the matched result, by default, sed will print
"original line+processed result", if we only want the result, use '-n' option to suppress the
original line output
aash
ash
bsh
foo.sh
hw.sh

[bobo@sh] ls|sed -n '/.sh$/p'
aash
ash
bsh
foo.sh
hw.sh
[bobo@sh] ls|sed -n '/.sh$/p'|sed '/foo/p'
aash
ash
bsh
foo.sh   # original output
foo.sh   # result of 'p' action
hw.sh
[bobo@sh] ls|sed -n '/.sh$/p'|sed -n '/foo/p'
foo.sh
[bobo@sh]
[bobo@sh] ls|sed '/.sh$/s/foo/bar/g'|grep -E "foo|bar" # repace foo with bar, s/pat/rep/g does
not have any output, it applies on the original output
bar.sh

[bobo@sh]
```

```
# sed '/b/s/b/d/g'
a
b
-->
a
d  # original line, changed from b-d
sed -n '/b/s/b/d/g'  # cannot get modified result
sed  '/b/s/b/d/g'  # can get modified result, show original lines

# add one line, command/action 'a'
[bobo@sh] echo abc|sed 'adef'  # 'a' is the action, 'def' is the newline appended
abc
def
[bobo@sh]

[bobo@sh] echo "abc
> def"|sed '/abc/anewline'
abc
newline
def

# replace/add with multiple lines, use '\' to renew two lines
[bobo@sh] echo abc|sed 's/b/new\
> li\
> ne/g'
anew
li
nec
[bobo@sh]

# multiple commands
[bobo@sh] echo abc|sed 's/a/A/1;s/c/C/g'    # use ';'
AbC
[bobo@sh] echo abc|sed  -e 's/a/A/1' -e 's/c/C/g'  # use '-e'
AbC
[bobo@sh] echo abc|sed  '/bc/{p;s/c/C/;p}'       # /bc/ is the regular expression, match the
lines that 'bc' appears', only process with these lines;  use {cmd1; cmd2;} , do different
actions to the same line
abc  # first 'p'
abC  # s/c/C/, original line changed
abC  # second 'p', original line changed
[bobo@sh] echo "abc
> dcc
> def"|sed '/.c/{s/c/C/g}'
abC
dCC
def
[bobo@sh]
```

11.Regex/BRE/ERE/Wildcard

```
# BRS  # basic, only '.', '*', '[a-z]
# ERE  # {1,} use (group)
# regexp  # a concept, python regexp/perl regexp, include much more special tokens
# wildcar/glob   # shell translate, '*' matches any char,[a-z], {ab,cd} matches ab and cdr
'+' previous char recurs more than one time
[bobo@sh] echo aabc|grep -E "a+"
aabc  # 'aa' matches 'a+'
[bobo@sh] echo baaabc|grep -E "a+"
baaabc # 'aaa' matches 'a+'
'{n}' # recurr exact times
[bobo@sh] echo babc|grep -E "a{2}"
[bobo@sh]
[bobo@sh] echo baaabc|grep -E "a{2}"
baaabc  #'aa' matches 'a{2}

'$' the end of line
'^'  the beginning of line

[bobo@sh] echo babc|grep -E "^c"    #'b' not match '^c'
[bobo@sh] echo babc|grep -E "c$"
babc   # 'c' matches 'c$'
[bobo@sh]

[^a-z]  # not match all in the list a-z
[bobo@sh] echo babc|grep -E "[^a-z]"  # 'babc' all belongs in '[a-z]', [^a-z] exclude all
[bobo@sh] echo babc|grep -E "[^b-z]"  # 'a' not in the exclude list [b-z], so matches
babc

[:upper:]  # for all upper case char, standard regexp
[[:upper:]]  # some command only support [[:upper:]]

[bobo@sh] echo babc|tr [:lower:] [:upper:]
BABC
[bobo@sh] echo babc|tr [[:lower:]] [[:upper:]]
BABC
[bobo@sh]
```

12. Watch the differences between AIX/Linux commands

```
# for most commands, linux command have much more options and modern features
# find

# date
[bobo@sh] date  --date="2 days ago"
2017年 03月 28日 星期二 16:57:00 +07
ocs@MTG8_OCS1_2:[/ocs]$date  --date="2 days ago"
date: Not a recognized flag: -
Usage: date [-u] [+"Field Descriptors"]
ocs@MTG8_OCS1_2:[/ocs]$

# sort
# linux sort commands may act different
# split
```

13.Debug shell script

```
# set -x to enable debug
[bobo@sh] cat hw.sh
set -x
echo "Script Name: $0"
echo "Last parameter: ${@:$#} "
[bobo@sh] ./hw.sh
+[2] echo 'Script Name: ./hw.sh'
Script Name: ./hw.sh
+[3] echo 'Last parameter: ./hw.sh '
Last parameter: ./hw.sh
[bobo@sh]

## sh -x
sh -x hw.sh
[bobo@sh] sh -x hw.sh
+[1] echo 'Script Name: hw.sh'   # command translate and execute process, start with '+[1]'
Script Name: hw.sh              # output
+[2] echo 'Last parameter: hw.sh '
Last parameter: hw.sh
[bobo@sh]
'+[line_no]'  --> PS4
[bobo@sh] echo $PS4
+[$LINENO]

[bobo@sh] export PS4="==debug line=="
[bobo@sh] ./hw.sh
==debug line== echo 'Script Name: ./hw.sh'
Script Name: ./hw.sh
==debug line== echo 'Last parameter: ./hw.sh '
Last parameter: ./hw.sh
[bobo@sh]

# control debug mode by flag
flag_debug=true
$flag_debug && set -x ||set +x

# pause the script, wait user input
read var?prompt
[bobo@sh] cat hw.sh
echo "Last parameter: ${@:$#} "
read var?"please confirm "
echo "continue"
[bobo@sh] ./hw.sh
Last parameter: ./hw.sh
please confirm somewords   # type to continue
continue
[bobo@sh] ./hw.sh
Last parameter: ./hw.sh
please confirm   # ctrl+c to exit
[bobo@sh]

[bobo@sh] cat hw.sh
echo "Last parameter: ${@:$#} "
```

```
read var?"please confirm "
if [ "$var" != "y" ];then
    exit
fi
echo "continue"
[bobo@sh] ./hw.sh
Last parameter: ./hw.sh
please confirm n
[bobo@sh] ./hw.sh
Last parameter: ./hw.sh
please confirm y
continue
[bobo@sh]

# debug in function
# bash/ksh93(linux ksh, or modern ksh) , by default , set 'debug in function' open, ksh/AIX,
default off
ocs@MTG8_OCS2_2:[/ocs/scripts]$sh -x hw.sh
+ export PS4=+[$LINENO]
+[10]foo
Last parameter:
please confirm y
continue
ocs@MTG8_OCS2_2:[/ocs/scripts]$cat hw.sh
export PS4='+[LINENO]'
foo(){
echo "Last parameter: $1 "
read var?"please confirm "
if [ "$var" != "y" ];then
    exit
fi
echo "continue"
}
foo

ocs@MTG8_OCS2_2:[/ocs/scripts]$
ocs@MTG8_OCS2_2:[/ocs/scripts]$sh -x hw.sh
+ export PS4=+[$LINENO]
+[11]foo
+[2]echo Last parameter:
Last parameter:
+[3]read var?please confirm
please confirm y
+[4][ y != y ]
+[7]echo continue
continue
ocs@MTG8_OCS2_2:[/ocs/scripts]$

ocs@MTG8_OCS2_2:[/ocs/scripts]$cat hw.sh
export PS4='+[$LINENO]'
foo(){
set -x   # open debug in function, AIX need , linux noneed

echo "Last parameter: $1 "
```

```
read var?"please confirm "
if [ "$var" != "y" ];then
    exit
fi
echo "continue"
}
foo
```

14. Application: Basic shell application

14.1 write log

```
logfile=a.log
plog(){
  echo "`date +%Y-%m-%d_%H:%M:%S`|$$|$*" >>$logfile
}

# usage
ocs@MTG8_OCS2_2:[/ocs/scripts]$plog this is log
ocs@MTG8_OCS2_2:[/ocs/scripts]$cat a.log
2017-03-30_17:13:08|6422896|this is log
ocs@MTG8_OCS2_2:[/ocs/scripts]$plog second line
ocs@MTG8_OCS2_2:[/ocs/scripts]$cat a.log
2017-03-30_17:13:08|6422896|this is log
2017-03-30_17:13:18|6422896|second line  # append
ocs@MTG8_OCS2_2:[/ocs/scripts]$
```

14.2 move cdr, backup cdr

```
find . -type f -name "in*_4_G_*.s" |xargs -I {} mv {} dest_dir
find . -type f -name "in*_4_G_*.s" | while read name
do
mv $name $newname
done
cp $name $newname
```

14.3 interact with remote file server (sftp)

```
# transfer file
sftp user@host <<EOF
put file
EOF

# check remote path
(echo cd $path) |sftp -b - user@host
if [ $? -ne 0];then
    # error handler
fi
```

14.4 check system resource/information (refer to monperf.sh/DailyInspect)

```
# df  # check filesystem
# svmon # check memory
# iostat 1 10  # check interval 1sec, repeat 10 times, check disk io
```

14.5 interact with DB

```
sqlplus -s user@inst <<EOF    # '-s' suppress the welcome output
set heading off pagesize 0
select count(1) from user tables
EOF

(echo select first 1 * from bal;)|ttisql -v 1 "dsn=ocs;user=ocs;pwd=ocs"  # '-v 1' minium
output, only output result

(echo select * from xx)|mdbSQL

mdbSQL <<EOF
select * from  xx
EOF
```

14.6 analyze CDR (text, zte format cdr)

```
# awk
{
  1=222
  2=333
}

awk -F'=' -v FN="$F" '{
if($0 ~ /^\{/){s=1}   # if match '{', the first line
else if($0 ~ /^\}/){   # if matches '}', the last tline
    if(CDR[950]==1){amount=amount+CDR[42];MSISDN=CDR[1];event=CDR[21]};
    if(CDR[951]==1){amount=amount+CDR[43];event=CDR[21]};
    if(CDR[952]==1){amount=amount+CDR[44];event=CDR[21]};
    if(CDR[953]==1){amount=amount+CDR[45];event=CDR[21]};
    print substr(CDR[3],1,8)","event","amount","MSISDN; s=0;
    for (key in CDR) {delete CDR[key]};
    amount=0
    }
else{                    # the content (multiple lines), store lines to array, CDR[1]=222,
CDR[2]=333
    if(s==1){CDR[int($1)]=$2}
    else{print "No start bracket"}
    }
}' $file
```

14.7 write service script(sysv-init && upstart compatible,)

```
# must write 'start|stop|status|forcestop' actions as input parameter
case "$1" in
'start')
start
;;
'check')
check
;;
'stop')
stop
;;
'status')
status
;;
forcedstop )
forcedstop
;;
*)
echo "Usage: myservice.sh { start | stop | status |forcedstop }"
exit 1
esac

# service appsvr status --> call app_service_script 'status'

# service appsvr start --> call app_service_script 'start'
```

14.8 monitor process status

```
ps -ef|grep "PROC_NAME"
ps -p $PID
ps -fu $LOGNAME|grep  -F "proc_name"  # LOGNAME is the login user name, e.g. 'ocs'
```

14.9 judge active and standby server

```
# judge by ip or file system '/ocs,/bill' exists or not
/ocs/scripts/judge_node.sh  # on mtg8
active_ip=xx.xx.xx.xx
standby_ip=xx.x.x.x
netstat -in|grep -c "xx.xx.xx.xx"  # if result 1, expected ip exists
ifconfig -a|grep -c "$active_ipe"
df -P|grep -c "/bill" #
```

14.10 detect processes ran in background (cycle cdr script)

```
Recurrate xx -M 3 -i 1 &
Recurrate xx -M 3 -i 2 &
Recurrate xx -M 3 -i 3 &
ps -ef|grep -c "Recurrate xx -M 3 " # by checking the count, we known how many bg process
running
```