PS command is used to see the currently running process on a unix host

-e : Selects all processes

-f : Full format listing

```
# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0  2011 ?        00:00:56 init [5]
root         2     1  0  2011 ?        00:00:01 [migration/0]
root       193     1  0  2011 ?        1-17:01:10 [kswapd0]
root       194    51  0  2011 ?        00:00:00 [aio/0]
root       384 27580  0 Apr03 ?        00:00:00 crond
oracle     387   384  0 Apr03 ?        00:00:00 /bin/sh -c bash
oracle   18660     1  0 Feb22 ?        00:00:00 script rman.sh
oracle   18661 18660  0 Feb22 pts/6    00:00:00 sh -i
oracle   23239 23232  0 04:26 ?        00:00:00 sshd: user1@pts/0
oracle   23240 23239  0 04:26 pts/0    00:00:00 -bash
root     27823     1  0  2011 tty1     00:00:00 /sbin/mingetty tty1
root     27824     1  0  2011 tty2     00:00:00 /sbin/mingetty tty2
```

UID - user id

PID - process id

PPID - parent process id

C - percentage CPU used throughout the life time of the process

STIME - start time of the process

TTY - terminal type associated with the process

TIME - cumulative CPU time in dd-HH:MI:SS format

CMD - name of the process of command

My own process id

`echo $$` will display your own process id

Files to know

/dev/null – Null Device / Black hole / Bit bucket

```
$ ls -ltr /dev/null
crw-rw-rw-  1 root root 1, 3 Oct 12 18:19 /dev/null
$ echo "this is junk " > /dev/null
$ cat /dev/null
```

It is a place for dumping unwanted information

/var/spool/mail/$USER – mail file

Used to receive emails for a user account from external email servers using POP3

Shell programmers use it to redirect messages from scripts to the mail file

```
$ echo "over to the mail file..." >> /var/spool/mail/user1
$ cat /var/spool/mail/user1
```

over to the mail file...

/etc/passwd – user information file

Authorized system users have login account entries in this file

Each entry in the /etc/passwd file contains seven fields, separated by a colon (:)

```
loginID:x:UID:GID:comment:home_directory:login_shell
$ more /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
```

/etc/shadow – user password information file

Contains encrypted passwords of the user accounts, each entry has nine fields separated by a colon (:)

Only the root user can read the /etc/shadow file

```
loginID:password:lastchg:min:max:warn:inactive:expire:
```

```
$ more /etc/shadow
```

```
root:5RiJS.yvdGBkU:6445::::::
```

```
daemon:NP:6445::::::
```

/etc/group – system group entries file

This file defines the default system group entries, each entry has four fields separated by colon (:)

Each user belongs to a group that is referred to as the user's primary group, however a user can also belong to 15 additional groups called secondary groups

```
groupname:group-password:GID:username-list
```

```
$ more /etc/group
```

```
root::0:root
```

```
other::1:
```

/etc/motd – message of the day

This is intended to provide greetings displayed whenever a user logs in. By default, this file is empty

You can use this file to notify users (at login) about system maintenance windows, reminders to clean up unwanted files etc

```
$ more /etc/motd
```

```
Please clean up your home directory ($HOME) frequently
```

/etc/oratab or /var/opt/oracle/oratab – oratab file

This file lists all the Oracle databases in a host along with their home directories, which helps the developers to set the appropriate Oracle home location for a particular database

```
$ cat /var/opt/oracle/oratab

PROD:/u01/oracle/product/8.1.7:N

DEV:/u01/oracle/product/9.2.0:Y

STG:/u01/oracle/product/10.1.0:N

TEST:/u01/oracle/product/10.2.0:Y

QA:/u01/oracle/product/11.2.0:Y
```

While true

The loop keeps executing even if the condition becomes false

```
$ while true

do

echo "Enter a value less than 5..."

read a

if [ "$a" -gt 5 ]

then

echo "I asked for a number less than 5"

else

echo "Good choice"

fi

done
```

Until false

The loop keeps executing even if the condition becomes true

```
$ until false

do

echo "enter a value greater than 5 "
```

```
read a
if [ "$a" -lt 5 ]
then
echo "I SAID GREATER THAN 5"
else
echo "Thanks for exiting me out"
fi
done
```

break

You use the ‘break’ command to terminate a continuously running statement/loop

```
until false
do
echo "enter a value greater than 5 "
read a
if [ "$a" -lt 5 ]
then
echo "I SAID GREATER THAN 5"
else
echo "Thanks for exiting me out"
break
fi
done
```

Continue

Continue will continue to process a loop even if the condition becomes invalid

```
$ more continu
echo "setting a=0"
a=0
while [ "$a" -lt 10 ]
do
if [ "$a" -eq 5 ]
then
echo "a is 5 now"
echo "$a"
sleep 1
continue
else
echo "a is not 5"
echo "$a"
sleep 1
fi
a=`expr $a \+ 1`
done
```

File Attribute

-f : to check if it's a file and it exists

-d : to check if it's a directory and it exists

-r : to check if the file exists and is readable

-w : to check if the file exists and is writable

-x : to check if the file exists and is executable

-s : to check if the file exists and is not empty

-z : to check if the string is of zero length

-n : to check if the string if of non-zero length

```
$ if [ -f "text1" ]
then
echo "true"
else
echo "false"
fi
true
```

id

id command is used to know your own UID and GID

```
$ id
uid=101(user1) gid=300(group1)
$ id root
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk)
```

chown

chown is used to change ownership of a file or directory but only the root user can do this

```
$ ls -l file7
-rw-r--r-- 1 user1 staff 672 Jun 1 15:11 file7

$ chown user2 file7

$ ls -l file7
-rw-r--r-- 1 user2 staff 672 Jun 1 15:12 file7

$ ls -l
total 12
drwxr-xr-x 2 user1 group1 4096 Mar 21 04:45 dir1

$ ls -l dir1
total 0
-rw-r--r-- 1 user1 group1 0 Mar 21 04:45 file1.1
-rw-r--r-- 1 user1 group1 0 Mar 21 04:45 file1.2

$ chown user2 dir1

$ ls -l
total 12
drwxr-xr-x 2 user2 group1 4096 Mar 21 04:45 dir1
```

```
$ ls -l dir1
total 0
-rw-r--r-- 1 user1 group1 0 Mar 21 04:45 file1.1
-rw-r--r-- 1 user1 group1 0 Mar 21 04:45 file1.2

$ chown -R user2 dir1

$ ls -l dir1
total 0
-rw-r--r-- 1 user2 group1 0 Mar 21 04:45 file1.1
-rw-r--r-- 1 user2 group1 0 Mar 21 04:45 file1.2
```

Groups

Groups command gives you information about you primary and secondary groups

```
# groups
root bin daemon sys adm disk

# groups user1
user1 : group1

# groups user1 user2
user1 : group1
user2 : group2
```

chgrp

chgrp is used to change the group privileges of a file/directory

```
# ls -l file4
-rw-rw-r-- 1 user1 staff 874 Jun 1 15:08 file4

# chgrp class file4

# ls -l file4
-rw-rw-r-- 1 user1 class 874 Jun 1 15:09 file4

# ls –ltr
drwxr-xr-x  2 user1 group1 4096 Mar 20 05:01 dir1

# ls -ltrR dir1
dir1:
total 0
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.2
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.1

# chgrp group2 dir1

# ls –ltr
drwxr-xr-x  2 user1 group2 4096 Mar 20 05:01 dir1
```

```
# ls -ltrR dir1
dir1:
total 0
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.2
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.1

# chgrp -R group2 dir1

# ls -ltrR dir1
dir1:
total 0
-rw-r--r--  1 user1 group2 0 Mar 20 05:01 file1.2
-rw-r--r--  1 user1 group2 0 Mar 20 05:01 file1.1
```

Chown

Chown allows you to change both the ownership and group privileges of a file/directory

```
# ls –l
drwxr-xr-x  3 user1 group1 4096 Dec  7 00:11 dir1
-rw-r--r--  1 user1 group1   32 Dec  7 00:25 file1

# chown user2:group2 file1
```

```
# ls -l
drwxr-xr-x  3 user1 group1  4096 Dec  7 00:11 dir1
-rw-r--r--  1 user2 group2    32 Dec  7 00:25 file1

# ls -l dir1
drwxr-xr-x  3 user1 group1  4096 Dec  7 00:11 dir2
-rw-r--r--  1 user1 group1     0 Dec  7 10:50 file2.1

# chown -R user2:group2 dir1

# ls -ld dir1
drwxr-xr-x  3 user2 group2 4096 Dec  7 10:50 dir1

# ls -l dir1
drwxr-xr-x  3 user2 group2 4096 Dec  7 00:11 dir2
-rw-r--r--  1 user2 group2    0 Dec  7 10:50 file2.1
```

Permissions

File / Directory permissions tell unix what can be done/not done by someone on a particular file/directory.

There are three things that you can/cannot do with a file or directory

- read
- write
- execute

- rwx r-x r-x

  123 456 789

1,2,3 read, write, execute permission for User (Owner) of file

4,5,6 read, write, execute permission for Group

7,8,9 read, write, execute permission for Other equivalent

Octal Digit	Text Equivalent	Binary Value	Meaning
0	---	000	All types of access are denied
1	--x	001	Execute access is allowed only
2	-w-	010	Write access is allowed only
3	-wx	011	Write and execute access are allowed
4	r--	100	Read access is allowed only
5	r-x	101	Read and execute access are allowed
6	rw-	110	Read and write access are allowed
7	rwx	111	Everything is allowed

```
# ls -l
-rw-r--r--  1 user1 group1    0 Mar 20 05:01 file2

# chmod 755 file2

# ls -ltr
```

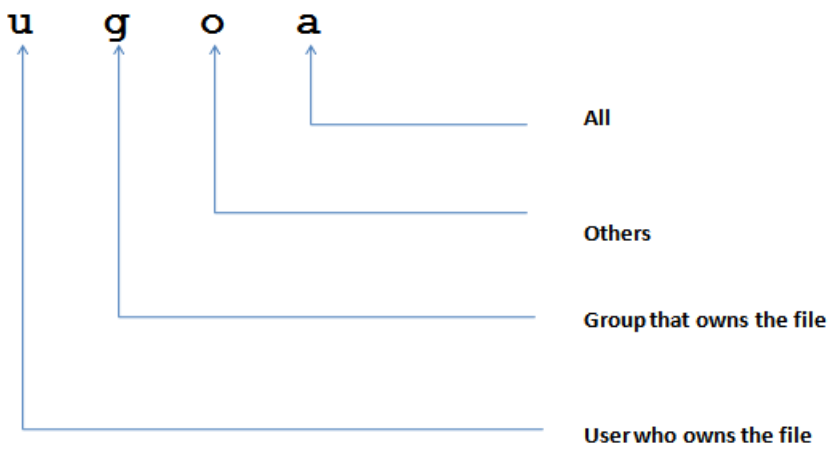
```
-rwxr-xr-x  1 user1 group1    0 Mar 20 05:01 file2

# ls –l
drwxr-xr-x  2 user1 group1 4096 Mar 20 05:01 dir1

# ls -l dir1
total 0
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.1
-rw-r--r--  1 user1 group1 0 Mar 20 05:01 file1.2

# chmod -R 444 dir1

# ls –l
dr--r--r--  2 user1 group1 4096 Mar 20 05:01 dir1
# ls -l dir1
total 0
-r--r--r--  1 user1 group1 0 Mar 20 05:01 file1.1
-r--r--r--  1 user1 group1 0 Mar 20 05:01 file1.2
```




```
# chmod -R u+wx dir1

# ls -l
drwxr--r--  2 user1 group1 4096 Mar 20 05:01 dir1

# ls -l dir1
total 0
-rwxr--r--  1 user1 group1 0 Mar 20 05:01 file1.1
-rwxr--r--  1 user1 group1 0 Mar 20 05:01 file1.2

# chmod -R o-r dir1

# ls -l
drwxr-----  2 user1 group1 4096 Mar 20 05:01 dir1

# ls -l dir1
total 0
```

```
-rwxr-----  1 user1 group1 0 Mar 20 05:01 file1.1
-rwxr-----  1 user1 group1 0 Mar 20 05:01 file1.2
```

Pipes

A pipe is a mechanism which takes the output of a command as its input for the next command

```
$ who
root     :0           Oct 12 18:20
user1    pts/0        Sep 14 12:13 (172.16.203.213)
user1    pts/4        Mar  2 16:11 (172.16.245.45)

$ who|wc -l
3

$ ls -ltr
total 288
-rwxr-xr-x   1 user1 group1     0 Jun  7  2005 root.sh.old
drwxr-x---   3 user1 group1  4096 Nov  2  2010 wwg
drwxr-x---   4 user1 group1  4096 Nov  2  2010 uix
drwxr-x---   3 user1 group1  4096 Nov  2  2010 tg4tera
drwxr-x---   3 user1 group1  4096 Nov  2  2010 tg4sybs
drwxr-x---   3 user1 group1  4096 Nov  2  2010 tg4ingr
drwxr-x---   3 user1 group1  4096 Nov  2  2010 tg4ifmx

$ ls -ltr|wc
     66     587    3348
```

```
$ ls -ltr |wc –l
66

$ ls -ltr| wc -w
587

$ ls -ltr|wc -c
3348

$ ls -tlr |wc
     66     587    3348

$ ls -tlr|wc|wc –l
1

$ ls -tlr|wc|wc
 1       3      24
```

Which

'which' command is used to know the absolute path of a binary

```
$ which wc
```

```
/usr/bin/wc
```

```
$ which ksh
/bin/ksh
```

```
$ which who
/usr/bin/who
```

```
$ which whoami
/usr/bin/whoami
```

```
$ which date
/bin/date
```

```
$ which mv
/bin/mv
```

PATH

The PATH lists all the locations of the unix binaries – so when a command is entered, the shell quickly searches through all these locations until it can find a directory where the executable exists

```
$ echo $PATH
```

```
$ ls
-bash: ls: No such file or directory
```

```
$ export PATH=/bin

$ echo $PATH
/bin

$ ls
file1  file2  dir1  dir2

$ who
-bash: who: command not found

$ which who
/usr/bin/which: no who in (/bin)

$ export PATH=$PATH:/usr/bin

$ echo $PATH
/bin:/usr/bin
$ which who
/usr/bin/who

$ which ifconfig
/usr/bin/which: no ifconfig in (/bin:/usr/bin)

$ export PATH=/bin:/usr/bin:/sbin

$ echo $PATH
```

```
/bin:/usr/bin:/sbin


$ which ifconfig
/sbin/ifconfig


$ ifconfig
bond0   Link encap:Ethernet  HWaddr 84:2B:2B:08:2A:BD
          inet addr:192.168.84.84  Bcast:172.16.225.255  Mask:255.255.255.0
          inet6 addr: fe80::862b:2bff:fe08:2abd/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:13093025243 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34582125408 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3332059057864 (3.0 TiB)  TX bytes:50564046628179 (45.9 TiB)
```

Binary calculator

bc is the command to invoke the binary calculator

```
$ bc
2+9
11
3*5
15
3-4
Ctrl+d
$
```

```
$ echo "2.3*2.1"|bc
4.8

$ echo "4-5"|bc
-1

$ echo "4/3"|bc
1
```

Calendar

This is used to display the current month, current year or a month or year from the past or future.

```
$ cal
 February 2009
Su Mo Tu We Th Fr Sa
 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

$ cal -3
 March 2012            April 2012            May 2012
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7         1  2  3  4  5
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
```

```
18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
25 26 27 28 29 30 31  29 30                 27 28 29 30 31
```

```
$ cal 3 2010
 March 2010
Su Mo Tu We Th Fr Sa
    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
```

```
$ cal 2009
```

```
$ cal –y 2009
```

```
$ cal -j 2 2009
 February 2009
Sun Mon Tue Wed Thu Fri Sat
 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
```

Modes of Debugging

-x : Display commands and their arguments as they are executed. (displays a (+) symbol before every output and input statement)

-v : Display shell input lines as they are read

Normal execution of a script

```
$ ksh calci
enter 1st number :
3
enter 2nd number :
5
Select one of the below choices
a or A for addition
s or S for subtraction
m or M for multiplication
d or D for division
m
the result is : 15
```

using –x mode of debugging

```
$ ksh -x calci
+ echo enter 1st number :
enter 1st number :
+ read a
4
+ echo enter 2nd number :
```

```
enter 2nd number :
+ read b
3
+ echo Select one of the below choices
Select one of the below choices
+ echo a or A for addition
a or A for addition
+ echo s or S for subtraction
s or S for subtraction
+ echo m or M for multiplication
m or M for multiplication
+ echo d or D for division
d or D for division
+ read d
a
+ expr 4 + 3
+ c=7
+ echo the result is : 7
the result is : 7
```

using –v mode of debugging

$ ksh -v calci

```
#!/bin/ksh
echo "enter 1st number : "
enter 1st number :
read a
```

```
5
echo "enter 2nd number : "
enter 2nd number :
read b
2

echo "Select one of the below choices"
Select one of the below choices
echo "a or A for addition"
a or A for addition
echo "s or S for subtraction"
s or S for subtraction
echo "m or M for multiplication"
m or M for multiplication
echo "d or D for division"
d or D for division
read d

case $d in

        a|A) c=`expr $a \+ $b`
                echo "the result is : $c"
                ;;

        s|S) c=`expr $a \- $b`
                echo "the result is : $c"
```

```
                ;;

        m|M) c=`expr $a \* $b`
                echo "the result is : $c"
                ;;

        d|D) c=`expr $a \/ $b`
                echo "the result is : $c"
                ;;


esac
the result is : 3
```

Escape characters

escaping a character with a backslash(\) cancels the special meaning of that character

```
$ cat escap
echo "enter a number"
read a
echo "the number that you entered is $a"
echo "the return value of $a is : $a"


$ ksh escap
enter a number
4
the number that you entered is 4
```

```
the return value of 4 is : 4

$ cat escap
echo "enter a number"
read a
echo "the number that you entered is $a"
echo "the return value of \$a is : $a"

$ ksh escap
enter a number
4
the number that you entered is 4
the return value of $a is : 4
$ cat escap
echo "enter a number"
read a
echo "the number that you entered is $a"
echo "the return value of $a is : $a"
echo "the return value of \$a is : $a"
echo "be courteous by saying "Please" "
echo "be courteous by saying \"Please\""

$ ksh escap
enter a number
3
the number that you entered is 3
the return value of 3 is : 3
```

```
the return value of $a is : 3
be courteous by saying Please
be courteous by saying "Please"

$ cat escap
echo "i am copying an "old_file" to a "new_file""

$ ksh escap
i am copying an old_file to a new_file

$ cat escap
echo "i am copying an \"old_file\" to a \"new_file\""
$ ksh escap
i am copying an "old_file" to a "new_file"
```

\n – Inserts a new line

\t – Horizontal tab

\v – Vertical tab

\c – Suppresses a new line

\a – System beep

```
$ cat escap
echo "this is \t horizontal tab"
echo "this is \n newline"
echo "this is \a system beep"
echo "this is \v vertical tab"
```

```
echo "this is not new line \c"


$ ksh escap
this is          horizontal tab
this is
 newline
this is  system beep
this is
         vertical tab
this is not new line $
```