10 Communication with other users

10.1 Introduction

Unix supports two types of communication commands: online communication and offline communication commands.

Online command enable user to communicate with other users who are currently logged in to the Unix system i.e. they provide online communication between the users. Here the communication is established only when user is logged in.

Offline communication command enable user to communicate with other users who are not currently logged in to the Unix system.

10.2 write and mesg

The **write** command allows user to communicate with other users whereas the **mesg** command allow or disallow write access to the terminal.

(a)write

It is on-line communication command. It sends a message to another user who is currently connected to the Unix system. The general syntax of **write** command is:

Syntax:

write user [ttyname]

The **write** command provides a facility to communicate with other users, by copying lines from his terminal to other user's terminal.

When a user runs **write** command, the other user on the other hand gets a message on his terminal in the following form:

Message from yourname@yourhost on yourtty at hh:mm ...

- ✓ For example, a user *bca1* writing on his terminal as follow:

  $write bca10 <enter>

- ✓ Instantly, the following message appears on user *bca10* terminal as follow:

  Message from bca1@localhost on pts/2 at 12:34 ...

- ✓ Any further lines type by user *bca1* will be copied to the terminal of user *bca10*. When user *bca1* presses interrupt character *(ctrl + d)* then the conversation is over which is indicated on other user's terminal by EOF.
- ✓ Following is the complete message display on terminal of user *bca10*.

  Message from bca1@localhost on pts/2 at 12:34 ...
  hello
  how are u?
  EOF

If the user is logged in on more than one terminal with same username then you have to specify the terminal name as the second argument to the **write** command.

For example, a user *bca10* is logged in at two different terminals *pts/3* and *pts/6*. Now, user *bca1* wants to send message to *bca10* then he types specific terminal name with username on his terminal as follow:

```
$write bca10 pts/6
messages
<ctrl-d>
```

In this case, if you do not specify terminal name then **write** command find solution to itself i.e. who has to send message. In Linux, it sends to user whose login time is latest.

There are two requirements for smooth **write** operation:
(i) The receiver must be logged in to the system
(ii) The receiver terminal is writable.

There are mainly two alternative to fulfill these requirements:

(1) **finger** command tells you that who are currently connected to the system and who can receive messages on their terminals. The finger displays a list of all those user who are logged in to the system and the list also places a * before the terminal-name where **mesg** is set to **–n**.

```
$finger <enter>
Login          Tty        Idle    Login Time
bharat         *pts/0     24      Oct 1 12:21
bac1           pts/1      .       Oct 1 12:31
```

The result shows that terminal *pts/0* is write protected and *pts/1* is writable.

2) The **who –T** command lists all the users who are currently logged in and places a '+' next to the username who have allowed messages and a '-' sign does not allowed.

```
$ who -T
bharat  -  pts/0   Oct 1 12:21
bca1    +  pts/1   Oct 1 12:31
```

Following are the situation in which **write** command do not work:

(i) Sender's terminal is write protected and try to send message. For example, the terminal of a user *bca1* is write-protected and he tries to send message to user *bca10* then he gets message on his terminal as follow:

```
$write bca10    #bca1 terminal is write-protected
You have write permission turned off
$
```

(ii) Receiver is not logged in. For example, user *bca1* writes a message to user *bca10* who is not logged in then user *bca1* shows a message on his terminal as follow:

```
$write bac10    #bca10 is not logged in
bca10 is not logged in
$
```

(iii)Receiver's terminal is write-protected. For example, user *bca1* writes a message to user *bca10* whose terminal is write-protected then user *bca1* gets following message on his terminal.

```
$write bca10    #bca10 terminal is write-protected
bca10 has message disable
$
```

NOTE: The super user can write to any terminal, irrespective of whether *mesg* has been set to *–y* or *–n*.

(b)mesg

It controls write access to your terminal by others. It is used to allow or disallow other users to write to your terminal.

The general syntax is:

syntax:

mesg [y|n]

✓ Without argument, it displays the write access state of your terminal.

```
$ mesg
is y
$
```

The output y indicates that your terminal get access to write by other users.
It has two options which determine the write access to your terminal. Table-(a.10) shows option used with **mesg** command.

Table-(a.10): options used with mesg command

Option	Meaning
Y or y	It allows write access to your terminal.
N or n	It disallows write access to your terminal.

✓ If a user wish to disallow write access to his terminal then he issued a command as follow:

```
$mesg n                    #disable write access permission to this terminal
$
```

10.3 wall

It sends a message on the terminal of all the users who are currently logged in to the Unix system. This command is specially designed for the super user. The **wall** command enables the super user to 'write to all' irrespective of whether the users have given write permission to their terminals or not. The general syntax of **wall** command is as follow:

Syntax:

wall [message]

✓ The *message* can be given as an argument to **wall** like this:

```
$wall "Good morning students"
Broadcast message from bca63 (pts/2) (Thu Sep  5 10:28:26 2013):

Good morning students
$
```

It sends message to all users who are currently connected to the system.

✓ The *message* can also be sent to **wall** using pipe like this:

```
$ echo "Good morning students"|wall
Broadcast message from bharat (Thu Sep  5 16:30:47 2013):

Good morning students
$
```

✓ The message can be also sent as the standard input from a terminal. In this case, the message should be terminated with the EOF key (usually Control-D).

```
$ wall
hello
There is an urgent.
Please contact to TMTBCA
<ctrl-d>
```

The **wall** program resides in */etc* directory (*/usr/bin* in linux).

10.4 motd

It is a special file resides in */etc* directory. If a super user wishes to put an off-line message to the user then he typed message in the *motd* file. The content of this file is displayed on user's terminal as soon as any of the user logged in to the system. The *motd* stands for the 'message of the day'. i.e. a system user puts message of the day to this file. It is only the super user who can change the contents of the file */etc/motd.*

As soon as a user login to the Unix system a file */etc/profile* file gets executed. The */etc/profile* is similar to AUTOEXEC.BAT file of DOS operating system. It gets executed every time the user logs in. Moreover, every user has his own user profile file, *.profile,* resides in user's home directory. The sequence of execution of these files is */etc/motd, /etc/profile* followed by *.profile.*

10.5 mail

It is off-line communication command. It sends and receives mail. Unlike, **write** and **talk**, it allow user to send a mail to users even if they are not logged in to the Unix system. The general syntax of **mail** command is:

Syntax:

```
mail [-s subject] [-c cc-addr] [-b bcc-addr]    to-addr1 to-addr2...
mail -f [name]
mail [-u user]
```

When users receive a mail message, they can do several things with it:

- ✓ View a mail on the terminal
- ✓ Save it in a mailbox
- ✓ Save it in a file
- ✓ Delete a mail
- ✓ Reply to it
- ✓ Forward it to other users

Following options are used with **mail** command:

(i) -s (subject): It specifies subject of the mail on command line. The subject is enclosed in quote if it contains spaces.

(ii) –c(carbon copy) : It sends carbon copies to list of users.

(iii) -b (blind carbon copy) : It sends blind carbon copies to list. List should be a comma-separated list of names.

(iv) -f: It reads the contents of your *mbox* ($HOME/mbox) or the specified file for processing; when you quit, mail writes undeleted messages back to this file.

```
mail -f /var/spool/mail/bca1    #reads the contents of bca1 mailbox
```

(v) -u : It is equivalent to **–f** option.

```
mail -u bca1    #reads the contents of bca1 mailbox
```

✓ A user want to send e-mail to user *bca1* then he typed on his terminal as follow:

```
$mail bca1             # to-addr
subject: Just hello    #ask for subject
Hello                  #type messages
How r u?
Have a nice day.
^d (ctrl+d)            #end of text
$
```

✓ A user can give *subject* and *to-addr* at command-line as follow:

```
$mail -s "Just hello" bca1
```

✓ A user can give *subject, cc-addr, bc-addr* and *to-addr* as follow:

```
$mail -s "Just hello" -b bca2, bca3  -c bca4, bca5     bca1
```

✓ When any new mail comes to the user, he is notified by the following message on his terminal.

```
$
You have new mail in /var/spool/mail/bca1
$
```

✓ Without any argument, **mail** command display list of header of each messages and then give '?' prompt. This is an interactive mode of **mail** command.

```
$ mail
mailx version nail 11.25 7/29/05.  Type ? for help.
"/var/spool/mail/bharat": 5 messages 1 new 1 unread
O 1 examination     Mon Sep 10 13:58   20/659   tmt
O 2 examination     Mon Sep 10 14:32   19/628   forward
O 3 bharat          Mon Sep 10 14:37   24/874   Re: forward
U 4 tybcasem5       Tue Aug  6 16:44   20/637   just hello
N 5 tybcasem5       Thu Sep  5 17:06   19/644   hello
?        #interactive prompt of mail command
```

The result shows that there are five mails among them 3-mails are old, 1-mail is unread and 1 is new. Letter O, U and N indicates Old, Unread and New mail message respectively. Second column indicate mail number, 3rd column indicates group name of a sender, 4th column shows mail date and time and last column shows subject of a mail.

✓ A user can view mail, delete it, reply it, forward it and so forth using the internal command shown in table (b.10). Internal commands are used in interactive mode of **mail** command at '?' prompt.

Table-(b.10): Internal commands of mail command

Command	Action
+	It displays next message.
-	It displays previous message.
N	It displays Nth message.
h	It displays only headers of all messages (default).
d N	It deletes message N, the current message deleted if N is not specified.
u N	It undeletes message N, the current message undeleted if N is not specified.

s flname	It saves current message with headers and postmarks in *flname* ($HOME/mbox if *flname* is not specified).
w flname	It saves current message without headers and postmarks in *flname* ($HOME/mbox if *flname* is not specified)
m user	It forwards mail to user
r N	It replies to sender of message N (the current message if N is not specified); delete message after reply.
q	It quits mail program.
! cmd	It runs UNIX command *cmd.*
?	It displays on-line help.

✓ To save header and postmarks of 3rd mail in file *mymail* then the command is:

```
?s3 mymail
?
```

✓ To delete mails 1 to 3 then the command is:

```
?d1-3
?
```

✓ A user can give reply to second message of mail like this:

```
?r2
I am fine.
<ctrl+d>
?
```

✓ A user can forward mail to user *bca10* like this:

```
?m bca10
subject: Just FD        #ask for subject
hello
<ctrl+d>
?
```

✓ **mail** saves messages in a mailbox, which normally is placed in the directory */var/spool/mail* and has same name as the login name i.e. *bca1*'s mail saved in */var/spool/mail/bca1*.

✓ When a mail message has been viewed by the user, it is saved in *$HOME/mbox* automatically on quitting from **mail** program.

NOTE: A user can recovered deleted mail in the current interactive mode of **mail** command. Once a user quit from mail program and re-enter again in interactive mode then he cannot recovered deleted mails of previous interactive mode.

news and talk command

(a) news:

This command is invoked by user to read any message that is sent by the system administrator. A system administrator types the information which he wants everyone on the network to know in different files in */usr/news* directory. Whenever a user logged in and if any fresh news has come in after the time of last logged out then a following message is displayed on user's terminal:

news: mess1 mess2 mess3

Where *mess1*, *mess2* and *mess3* are the names of the files in which the news items are available. When a user invokes the **news**, it shows the contents of all unread file(s) in sequence, the most recent item shown first.

Any news item already seen once can't be displayed again, so **news** either outputs this message or simply returns the prompt if all news items have been seen before.

```
$news
No news                    #SCO Unix simply returns the prompt.
```

The options used with **news** command are as follow:

(i)-n (name): It displays only the name of files whose contents are not been read before.

```
$news -n
news: mess1  mess2  mess3     #these files are not seen
$
```

(ii)-s : It lists the number of news items that have still not been read.

```
$news -s
3 news items              #3-messages are not read
$
```

(iii)-a (all) : It displays the contents of all news items regardless of whether they have been read or not.

✓ A user can also use **news** with a filename to display the specific message. You can do that even if the item has been read before:

```
$news  mess1
  ---display news---
$
```

(b) talk:

It is used to talk to another user who is still connected to the system. It is superior to **write** command that it splits the screen horizontally into two windows; one window shows you the received data while the other window is used for transmission. The general syntax of this command is:

Syntax:

```
talk person [ttyname]
```

talk is a visual communication program, which copies lines from one user terminal to that of another user.

- ✓ If you wish to talk to someone on your own machine, then *person* is just the person's username. If you wish to talk to a user on another host, then *person* is of the form *user@host*.
- ✓ If you wish to talk to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name, where *ttyname* is of the form `ttyXX' or `pts/X'.

When first called, **talk** contacts the *talk daemon* on the other user's machine, which sends the message to the user.

Once communication is established, the two users may type simultaneously; their output will appear in separate windows.

To exit from **talk** program, just type the interrupt character (normally ^C); **talk** then moves the cursor to the bottom of the screen and restores the terminal to its previous state.