UNIT 1 USER TO USER COMMUNICATION

Structure

1.0 INTRODUCTION

You have by now a fair idea of many UNIX commands and utilities. Some of these are simple while others are quite complex and have only been touched upon in this block. While we have covered some of the text processing facilities of UNIX, there is another of its design goals which has not yet been touched. When UNIX was being developed, one of the avowed aims was to provide users with an environment which facilitated sharing of information. This implied good facilities for communication between users.

In this unit we will look at the features available in UNIX for easy communication. Each of the commands has its own advantages and is appropriate in different situations. You should learn to use these commands and actually put them into practice. Unlike the other commands which you can learn by solitary effort, the communication features are best mastered by working with a partner to whom you can send practice messages. Of course this is not an essential requirement, and you could, if necessary, learn even the communication commands all by yourself.

1.1 OBJECTIVES

This unit will take you further down the road to exploring UNIX. By now you have had a feel of what UNIX is like. In this unit you will learn about some more of the strong points of UNIX. By the end of this unit you should be able to :

- Communicate on-line with other users on your machine using write
- Communicate off-line with other users with the help of mail and news
- Use the Bourne and C-shells for writing simple shell programs

1.2 ON-LINE COMMUNICATION

We will first look at the write command, which allows you to send a message to another user logged in onto the same machine at the same time. This is thus a means of on-line communication because the recipient obtains the message immediately. There are three other means of communication in UNIX that we will consider in later sections, and two of them are off line in that you send the message and the intended recipient might not pay any attention to it if he so desires.

Because of this property, you should be careful in using write. It can be very annoying to receive a message which clutters up your screen while you are concentrating hard trying to do some piece of work. Sometimes you have some precious output on your screen which you have obtained after some effort and a write message appears and causes that output to scroll off. Or you are working in an editor and the message garbles your screen (you can of course redraw it but you do not care to be disturbed). There are many such situations in which you would rather not receive any message on your screen. Again, some people are more touchy than others on this issue. So you should be careful and use write only when you have something urgent to say, or when you are sure that the other party will not mind. In fact one of the most irritating times to receive a write message is when you are already replying to somebody else's write message. Arrange for such a situation with some friends and see how it feels.

However you can try doing some homework before issuing a write command, if you must. You can try doing a ps on that person so that you get an idea of what he might be doing. If he is in the shell, for example, then it might not be so inconvenient for him. On the other hand if he is in vi or inside some application package, it might be best not to bother him. But then again, the urgency of your situation needs to be taken into account. Another thing you could do is to write a brief message asking if it is all right to write more. If the recipient does not want to be bothered at that time, he can tell you so. All said and done, you should be aware of the fact that write can be a rude command to use.

After this long sermon, we can come to the command itself, which should be quite simple for you after all of the two units of experience in UNIX that you have had. Let us find out some of the users on the system

```
% who
ramk      tty01   03:48
khanz     tty04   09:38
netman    tty06   08:43
shyama    tty07   08:19
```

Suppose you, khanz, want to write to shyama and ask her about some program she had promised to give you. So you say

```
% write shyama
Can I have that matrix inversion program please?
I need it badly to test out my work.
^D
%
```

We get the prompt back after sending the message. Notice a few things about write. One is that after issuing the command there is no indication at all that you should proceed with your message, because there is no prompt which appears nor any automatic acknowledgment which comes from the other end. The command expects input from the standard input and faithfully transmits it to the destination.

Therefore after issuing the command you can type in your message, which can be as long as you like. When you are finished, you should type the end of file character to signal of input. This character is usually ^D. On doing this you will get back the prompt. There is no indication whether the other party has received the message or not. Also the message should not be longer than a screenful or the other person might have difficulty reading it.

Now what about the other end of the channel? Let us zip across to shyama's terminal and see what she has got, omitting the secret work she was busy with when you butted in

```
Message from khanz tty04 (Fri Jun 13 13:13:13) ...
Can I have that matrix inversion program please?
I need it badly to test out my work.
EOF
```

So she did get your message, but what about a reply? Well you did not see anything on your screen because she has not had a chance to answer yet. Let us see her reply now

```
% write khanz
I was downloading it but the line got cut halfway.
Will give it when I have it.
Have patience or get it yourself.
^D
```

and that is exactly what you receive on your terminal except for the letters EOF in place of the ^D.

So you have now seen a straightforward case of exchanging urgent messages with write.

However in practice such messages would have been exchanged with the mail command, which allows you to send electronic mail messages to other users. Write is more commonly used to carry on a conversation with some other party on the terminal, rather than just send a message. Let us see how this can be done.

```
% who
ramk      tty01   03:48
khanz     tty04   09:38
pramod    tty05   14:07
netman    tty06   08:43
% write pramod
```

Instead of immediately writing out your message, you can wait to see if pramod is in a mood to respond. After a minute or so you have not had a reply and so you just come out with a ^D. He is too busy to talk to you, or maybe he just does not want to be disturbed and has arranged matters that way. This is getting to be irritating now. Let us try somebody else

```
% write ramk

Message from ramk tty01 (Fri Jun 13 14:10:08) ...
```

At last somebody is ready to talk to you! This means that ramk has something like this on his screen.

```
Message from khanz tty04 (Fri Jun 13 14:09:28) ...

% write khanz
```

Now you are both all set to carry on a conversation, because both have issued a write command for writing to each other and can consequently write to each other's terminal. So go ahead and pour out your woes to your only friend in the installation.

```
I wanted a program so badly but shWhich program? yama has not yet got it. I don'I am sure she must have tried.

t know what to do ....
```

What is happening here? This can be disconcerting for beginners, but there is nothing mysterious about it. The communication is asynchronous and full duplex. Both sides can transmit and receive at the same time, and unless you wait until the other side has finished, there can always arise opportunities for confusion. What you need is a protocol to be adhered to so that the screen does not get cluttered up and cause confusion. The thing to understand here is that there is no way of knowing when the other party has finished unless the protocol is set up and observed. This is because every character you type goes to the other side, and there is nothing which restricts a message to one line.

The UNIX documentation suggests an o (for over) after every bunch of lines and an oo (over and out) to indicate that you have finished the conversation and want to hang up. Of course you could give a ^D and the other party would know from the EOF that you had finished, but terminating a conversation like this without warning is as rude as hanging up a telephone without warning. You can work out some other protocol for your installation to indicate end of a message and the intention of hanging up. You could even specify these when you first issue your write command.

Let us see the result of a protocol where we are going to use ga at the end of each message and terminate the conversation with a bye. Now the previous conversation will be more intelligible to both parties.

```
I wanted a program so badly but Shyama has not yet got it.

ga

Which program?

ga

I don't know what to do.
```

```
ga

I am sure she must have tried.

bye

EOF

bye

^D
```

You will not fail to notice that things are much more orderly and understandable now.

This was when everything was favourable. Sometimes you will have the following spot of bad luck.

```
% write ramk

ramk is not logged on.
```

The response is self explanatory. If you try to write to a user who is not logged in you will get this message from UNIX. Since write is an on line command, it is not enough for the user to be a valid user on that installation. The user must be actually logged in to the system at the time you try to write to him. You can always make sure of that by doing a who first. If the person you want to write to is not logged in, you can save yourself the trouble of trying to write to him, knowing you will fail.

Then there are the times when you check out the users of the system

```
    % who
    ramk        tty00    00:46
    shyama      tty02    08:49
    khanz       tty04    08:53
    shyama      tty07    09:53
```

and try to write to shyama.

```
% write shyama

shyama logged more than once

writing to tty02
```

Now if that user is logged in onto more than one place, you can do a few things. You could ignore the fact and you will then be always writing to the lowest numbered terminal onto which that person is logged in. That is usually fine, but if you want to specifically write to the person at some other terminal, you can do so by giving the terminal number of your recipient as well with the command itself. Thus

```
% write shyama tty02
```

or

```
% write shyama tty07
```

depending on the terminal required. Thus it will be clear that you write to a terminal, not to a user. It follows that you can write to yourself if you wish.

Are you at the mercy of the others to be able to work peacefully? Not entirely, because there is a command in UNIX to prevent other users from being able to write on your terminal. In UNIX every device is a file and it has permissions associated with it like any other file. We will see more about this in the next unit on system administration. Here it is sufficient to understand that normally when you login, your terminal device file has permissions such that all users can write to your terminal. If you turn off write permission for other users, then nobody will be able to write to your terminal using the write command, and consequently will not be able to disturb you while you are working. You can do this using the chmod command, but you would then need to know more about device files, like what the file name is and where it is located.

A simple way to turn off write permission for other users is to say

```
% mesg n
```

Now anybody trying to write to you will get a message denying permission. You can issue this command when you are doing something important and when you do not want to be disturbed. But it is not polite to have this command in effect all the time. Just as it is rude to unnecessarily bother a person, it is also rude to shut your doors to others trying to reach you, perhaps with something important. So when you are not working on anything requiring great concentration, you can restore write permission to others by saying

```
% mesg y
```

If you just want to check the status of write permission on your terminal, you can say

```
% mesg

is y
```

and the current status is displayed.

You must remember that the denial of write permission lasts only for your current login session. The moment you logout and login from the same or some other terminal, you will have mesg set to y. This is because UNIX would not like some other user to be starting off a login session with a state dependent on the previous history of the terminal. All users must start off in the same mode. Therefore the permission modes on the terminal device file are reset everytime a login occurs. Also if you are logged in from more than one terminal, the mesg status on one terminal can be different from another. Changing that status on any terminal does not affect the status on any another terminal from which you are logged in. It should thus be clear that the mesg status pertains to a terminal rather than a user.

Another thing which we have not mentioned so far but you would have deduced from our discussion was that the super user is immune to your attempts at denying write permission to others. The super user can always read or write any file and your terminal device is no exception. There is no way you can prevent root from writing to your terminal, irrespective of mesg status.

We mentioned that write takes its input from the standard input. So you can always put a message in a file like writemsg and then send it with the write command like this

```
% write ramk  writemsg
```

After the usual message, ramk will see the contents of the file writemsg on his screen. You should take care to see that the file is not so long that it cannot fit on one screen, otherwise the poor recipient will not be able to read it unless he has the reflexes of a fighter pilot with ^S.

Another feature in write that is sometimes useful is the ability to run a UNIX system command while engaged in conversation. This is something that is not uncommon in interactive commands in UNIX and there will be many utilities that allow you to escape to the shell and come back to the utility in whatever state you were afterwards. And the moment you are allowed to run one command you can go to the shell itself by saying csh. In write this facility can be used by preceding the command with a !. So if you are talking to somebody and want to tell him the settings you have set in your\exrc, you could ask him to copy your .exrc file or if it is small, you could see the settings and tell him what they are. Let us say ramk wants to know whether number is on in your .exrc file.

```
% write ramk

Hello, ramk

ga

Hello, khanz

ga

Is number set on in your vi environment?

ga

Let me see. This will take a minute.

!cat ~/.exrc
```

```
set number
set tabstop=4
set terse
!
Yes it is set.
ga
Thank you
bye
EOF
^D
```

You have seen that the output produced by the command does not get sent to the recipient automatically. A ! indicates the end of the command given, and you can then continue with your conversation. If you wanted to send the output to your friend, you would have to type it out again after you got the !.

We will now take a quick look at another command which is on-line because the message is received immediately. This is the wall command which sends a message to all users who are logged in at the time. To use it say

```
% wall
Don't any of you guys want lunch?
^D
```

This message might not be liked by some serious types, especially if they are working on something requiring hard concentration. The command is usually used by the super user to send system messages to users. For example if a printer is going to be under maintenance for the next couple of hours he can tell all those who are logged in

```
# wall
The line printer is being serviced and will be available only after 2 o'clock.
^D
```

The # is the normal super user prompt. Since this message has come from the super user, all users will get irrespective of their mesg status. If an ordinary user issues a wall command, he cannot be certain if all those logged in will receive his message at all. That is why this command is suitable for the super user only.

1.3 OFF-LINE COMMUNICATION

Let us now look at two commands which allow UNIX users to communicate in off-line mode. This means that the users will not be able to talk or converse, but a message sent by one will be sent to the other, and the recipient can then decide whether he wants to look at it and maybe even act on it if needed.

You all must have heard about electronic mail or e-mail, as it is usually called. In fact, many computer professionals now refer to e-mail as mail and to conventional mail as paper mail. Today if you are onto some international network like the Internet, you can send mail to far off places like say, the United States, and if your partner wants to respond, you could have the reply the next day. However this is not the place to discuss this kind of communication. We will confine ourselves to sending electronic mail to other users on the same machine.

There are advantages and disadvantages to using mail, as opposed to using write. The problem with mail is that you cannot carry on a conversation with your counterpart at the other end. So if there is some small, urgent message to be sent and which the other party must see at once, you need to use write.

But this situation is not common as compared to the times when you just want to send a message to the other party. You either do not need a reply or you can wait for one.

Sometimes your message is a long one, much longer than can conveniently be digested during a conversation with write. These are the times when mail is very useful. Then again with write the other user has to be logged in at that moment if you want to communicate, while with mail you can send a message to any user who is registered on that system, irrespective of whether he is logged in at that time or not.

A message sent to a user by using mail gets stored in a mailbox allocated to that user, and stored somewhere in the file system. The user gets this message the next time he logs in

```
You have mail.
```

As long as there is some mail in your mailbox you will get this message everytime you login. You should therefore look at your mail and dispose it off while it is recent. It is not obligatory to look at your mail and UNIX does not compel you to do so. If you neglect to read your mail it might go stale. That is a good reason to inspect your mail regularly. You can delete all or part of your mail without reading it, if you wish.

With this background, let us see how to send mail to a user on the system. Let us say that you, khanz, want to send a message to ramk. The most straightforward way is to say

```
% mail ramk

How about a trip to the exhibition tomorrow at 14:00?

^D
```

As you can see, mail takes its input from the standard input. So after typing in the command, you see nothing else, not even a prompt. You can then type in your message and terminate it with ^D, as usual. The message then gets sent to ramk, because that is what you had asked for. Many users can be sent a message at the same time

```
% mail ramk shyama pramod

How about a trip to the exhibition tomorrow at 14:00?

^D
```

This method is not very convenient, as you can imagine. It is practical only for short messages, because you cannot edit any errors you happen to make while typing. If you have a long message, you can type it into a file by using any text editor like vi. If you save it in mailmsg, you can now send it to the users you want

```
% mail ramk shyama pramod  mailmsg
```

If you try to send mail to a user who is not mentioned in the list of users of the machine (in /etc/passwd), the message gets stored in the current directory of the sender in a file called, appropriately enough, dead.letter.

And if you have been wondering, yes, you can send mail to yourself. In fact it is usually a good idea to send one copy of messages to yourself so that you automatically record your outgoing messages. So you would say

```
% mail ramk shyama pramod khanz  mailmsg
```

and you would get a copy for yourself.

We will now see how to read mail. All you need to do is to say on the command line

```
% mail

No mail.
```

This means simply that you do not have any mail. But if you do have mail that you have not read, mail prints out the messages for you to see. Normally messages are printed out in reverse order of receipt, which means that the message received most recently is printed first, and the earliest message is printed last. After printing each message, mail waits for you with a ?. This means it is waiting for your decision on what to do with the message. We will just see what options you can exercise here.

To see the available choices, you can respond with a ? or a *. Both these are help commands which give you a list of available options with a brief explanation. We will look at each of them quickly. But first let us see mail print out a message.

```
% mail

From shyama Sat Jun 14 01:56:08 1994

I have that program for you. You can pick it up from

~shyama/utils/pub/c++/invert.C

Do let me know when you have it and if it works. If it does not work, tell me the problem
and I will try to see if it has been noticed and if a fix is available.

Sorry, but the lines were really bad and that is why it took so long. I finally made it early
morning.

?
```

You can ask for the message to be printed with a p. You can also ask to print the previous message with a -, or the next message with a + or a
. If you are done with a message, you can just ask for it to be deleted with a d command. This will delete the message from your mailbox.

You must be wondering where your mail is stored. It is in a file in the directory /usr/mail, and the name of the file is the same as your login name. This is usually called your mailbox. Saying d in response to the ? of mail will not remove your mailbox, it only removes the message from the mailbox.

Over time you will most probably accumulate a lot of mail in your mailbox. Apart from using space, this will make it difficult for you to see all your messages. You will not need to do that anyway, because you will have seen your older messages already. So you will have to remember which messages you have seen already, and as soon as you come to them, you will need to quit mail. It makes much more sense to delete mail you have read and do not want in your mailbox.

There could be some messages which are important or carry useful information. You would not want to lose such messages, nor would you want them in your mailbox for all eternity. Such messages are best saved in a file. To do so, say

```
? s shyama
```

This saves the message in a file called shyama in your current directory, provided you have write permission there. It is therefore better to be in your home directory or in your own directory tree when invoking the mail command. Of course if mail refuses to create the file, you can repeat the command with a full pathname to a directory where you do have write permission. The s command saves the entire message including the header in the file mentioned.

If the filename that you give already exists, the message is appended to the file. So the contents of the file will not be overwritten. In some situations you could find this feature useful. If you are saving correspondence on some subject, you could place all messages pertaining to it in the same file, where it will be easy for you to examine them later when you want. But saving unrelated messages in the same file might cause some confusion. You are the best judge of what messages to save and where to save them.

Do not save every message. Many messages are trivial and are relevant for a short time only. For example, an invitation to lunch received over mail need not be saved for a long time. You can let it be in your mailbox and delete it when the date of the invitation has passed.

There is another save option called w. This works just like s, but the message header is not saved. You can use this option when it is the contents of the message which are important and the header information is not. For example, if somebody mails you a document on how to use a particular piece of software, you might like to save the document but not the header. If you neglect to specify a file in which to save the message, it is stored in a file called mbox in your home directory.

Both s and w cause the message to be deleted from the mailbox. This is safe because the message is available in the file you saved it in. The next message is then displayed, and you can take whatever action you like to on that one.

You must remember that messages you ask to be deleted are not removed immediately. So if you have "deleted" a message and then come back to the previous message with a -, you will still see that message. The message goes away only when you quit from mail and invoke it again. Then you will no longer find messages that you had deleted the last time.

Here you will need to know how to exit from mail. There are three ways you can do so. Both ^D and q will get you back to the UNIX prompt after permanently removing messages you had marked for deletion in that session. The next time those messages will not appear. The x option will get you out of mail without affecting anything. So even if you had marked a message for deletion it will not be removed from your mailbox. This option is useful when you have marked messages for deletion by mistake and the whole mailbox is a mess. To get back the previous state you can exit with x.

A useful option is m for forwarding mail to other users. If you have received mail from somebody and want other users to see it as well, you can use this option. Thus if you want to send the message from shyama to gopalan and alice, who work in your project and might find the program useful, you can let them know about it

```
? m alice gopalan
```

They will both see a message like this

```
From shyama Sat Jun 14 01:56:08 1994

From khanz Sat Jun 14 10:43:38 1994 forwarded

I have that program for you. You can pick it up from

~shyama/utils/pub/c++/invert.C

Do let me know when you have it and if it works. If it does not work, tell me the problem
and I will try to see if it has been noticed and if a fix is available.

Sorry, but the lines were really bad and that is why it took so long. I finally made it early
morning.
```

Thus they can see the name of the original sender as well as that of the person who forwards the message to them. A recipient who wishes to forward a forwarded message can do so. A forwarded message gets deleted from your mailbox automatically.

We had mentioned in the previous section on the write command that most interactive commands have a way of escaping to the shell to run a command. The mail command also allows you to execute any shell command with a ! followed by the command. Thus while examining messages if you find that you could not save a message because you did not have write permission, you could check to see where you are

```
? s new_soft.doc
mail: cannot append to new_soft.doc
!pwd
/users/shyama/utils/pub/c++
!
?
```

As in the case of the write command, the end of the command is indicated with a !, because otherwise you would not know when the command hand ended. That could happen if the command you executed did not produce output on the screen, for example.

Unfortunately in this case you cannot rectify the situation completely by reaching your own directory tree, because you cannot use the shell escape facility to change your directory. You can only give the save command with a different filename, like /users/khanz/doc/soft/new_soft.doc.

Why can't you change directory using the shell escape facility? It might seem a great mystery, but you will get a complete answer in this unit itself, when we discuss shell programming. In brief, the reason is that every shell has an environment associated with it.

The current directory is part of the environment. When you invoke a shell command with the shell escape facility, the command gets executed in the environment of that shell, but does not affect the environment of the shell from which the command was called. In other words, trying to change the directory will work, but only in the shell in which that command was called. In the current shell in which the mail command is being executed, there will be no effect. However there is no problem with commands which do not seek to alter the shell environment, like pwd or ls. Since these commands only tell you about the environment (pwd) or about files in a directory (which is not part of the environment), there is no difficulty if you use them.

While we have seen a fair amount of options available with the mail command, we have not looked at the command line flags it provides you with. There is not very much to it here. You can use the -p flag to print every message without prompting you for action with the ?, or -q to exit without altering the mailbox whenever you interrupt the printing of a message, or -f to look at a different mail file. Thus if you have been saving a sequence of messages, perhaps on one topic, into a single file with s, then

```
% mail -f soft_file.doc
```

will execute the mail command from this file just as it usually does from the mailbox. The format should be the same as that of the mailbox (this is so if you use s to save) otherwise mail will not be able to distinguish between the different messages and things will not work out well.

The last flag can be useful. We have seen that mail prints the messages in reverse order. Sometimes this might not be what you need. If you want to look at messages in the order they were received, say

```
% mail -r
```

Now you will get the oldest message first and the most recent message last. Again this is valuable when studying a series of messages on the same topic, or where a later message can have references to earlier ones.

Finally, options can be combined with or without separate - signs, as in all other standard UNIX commands. So you can say

```
% mail -r -f soft_file.doc
```

or

```
% mail -rf soft_file.doc
```

to look at mail from soft_file.doc instead of the default mail file, and in the order of receipt of the messages.

Now while this covers the basic facilities that mail provides, it should be realised that the features of mail are rudimentary. There is a better user interface, mailx, that is available in UNIX. It gives more facilities and you should study it from the documentation. Now that computer networks are common and you will probably be sending mail to people in distant places, you might even like to use some other mail program. If you are on the Internet, for example, you can locate some good public domain mail program and use it on your installation.

You should use mail to communicate with people in your installation and elsewhere. It has many advantages over a phone call, paper mail or casual conversation. Since it amounts to writing down what you want to say, it ensures that you will be systematic, unlike a verbal exchange. Paper mail can be unreliable. It can get misplaced or might reach after a long time, and the recipient has to be present at his usual place of work to be able to see it. A phone call might not find the other party on the line or available at that time.

Electronic mail has none of these difficulties. The message reaches quickly and will not get misplaced. The other party does not have to be there at that time. He will get the message whenever he decides to look at his mailbox. The message need not be brief unlike the situation with a telephone answering machine. So you can see that the person need not be at his usual place of work at all. He can look up his mail from anywhere in the world if he can connect to his machine. That is why electronic mail is so popular now and it will soon become commonplace.

A disadvantage of electronic mail as described here is the lack of privacy. On the system, the super user can always look at anybody else's mail and you might not feel comfortable with this. You could encrypt your mail after you save it, but the super user can look at it before you do so. One possible solution is to use a public key cryptography mechanism and interface it to a mail program. Such schemes are already available in the public domain.

Let us now look at another communication command available in UNIX. This is again, like wall, a one to many or broadcast kind of command, though not quite, because the recipients have the choice of deciding whether to look at the message or not. The command is called news and is typically used to convey information about the local system or installation. However it can also be used for personal matters, and often is.

The news is not forced upon you, and you have the option of not looking at it all. If you want to read the news, say

```
% news -n

news:new_machine  seminar  classes
```

This option tells you about the names of the news items available for reading. You can then choose to look at any one or more items in detail by saying

```
% news classes

classes (khanz) Fri Jun 13 15:23:48 1994
```

Ram Kumar, our local guru, will take classes on the X-Window System from Monday June 16. They will last four weeks, Mon to Fri, from 17:15 to 19:15. All those who want an introduction to the subject should come regularly.

If you omit the name of a news item, then all items are displayed with the most recent item coming first. You cannot delete items that you do not want to see again, unlike the mail command, because here the news is not for you but for the whole user community. However, you are not bored to death with the same news again and again because every time you invoke the news command, only those items are shown which you have not seen before.

This is controlled by examining the time of a hidden file in your home directory called .news_time. Every time you invoke the command, the time of this file is updated. The news command will display only those items which are more recent than the time of this file. Thus the command lives upto its name.

But should you ever want to look at all the news, you can do so by saying

```
% news -a
```

This shows you all the news irrespective of .news_time, and is useful when you feel you have missed out on some news or when you want to read a news item again.

Now how does one put in news that one wants to make known? Each news item is the name of a file in /usr/news. So in our example, the news item classes will be /usr/news/classes, and the contents of that file will be the news that you see when you invoke the news command. The command compares the time of the files with that of .news_time to determine whether or not to display it. So if you have write permission in /usr/news, you can create a file with the appropriate name and put in the news with any text editor like vi. If you do not have write permission, you will need to make a file in your own directory tree and ask the system administrator to insert the news item for you. If you have the permission to add news, you should remove the news when it becomes too old to be useful or interesting or when enough time has passed. If the system administrator is the only one who can insert news, then he will be responsible for removing stale news items so that they no longer appear, even with the -a option.

We have now looked at the user level communication facilities in UNIX. You will need to experiment to become familiar with these. You should make use of them to make life at the installation easier. But the availability of none of these methods means that you start to disregard the normal communication channels we have always used, meaning do not stop talking to your colleagues because you have mail, news or write.

Check Your Progress

1. We have talked about learning the communication commands by yourself. How do you think this could be done?

....

....

2. What are the advantages and disadvantages of electronic communication compared to

   (a) Postal communication

   (b) A telephone conversation

....

....

....

3. What are the advantages and disadvantages of off-line and on- line communication?

....

....

4. If you wanted to send some message requiring some preparation, how would you use write?

....

....

5. Can you think of a couple of other ways of sending output to another terminal other than the ones covered in this unit?

....

....

6. What happens if a user logs out after you have started writing to him?

....

....

7. Arrange with two friends to have both of them write to you at the same time. Can you reply to both of them at the same time? Can you even figure out who is saying what?

....

....

8. Suppose you wanted to send a long message to all your colleagues giving the details of a picnic they can go on. What command would you use and why?

....

....

....

9. Apart from the fact that the news can be read by everybody, how does it differ from mail?

....

....

1.4 SUMMARY

In this unit, you have learnt to communicate with other users, both interactively and by electronic mail and news. You should use these facilities extensively because they are so easy to use and so convenient.

1.5 MODEL ANSWERS

Check Your Progress

1. You can learn the communication commands by yourself. Just acquire two (or more) accounts, even if temporarily. Then all you need is two (or more terminals), preferably located side by side so that you can easily work on both or all.

2. No model answer.

3. No model answer.

4. Prepare the message and store it in a file, say msgfile. For this you can use vi or any other editor. Then say-

   ```
   % write khanz  msgfile
   ```

   because write reads the standard input. You will have to see that khanz is logged in. If he is logged in from more than one terminal you will need to specify the terminal as well.

5. Try

   ```
   % cat  /dev/tty 04
   ```

   Where /dev/tty 04 is the device file corresponding to the target terminal.

   Also try -

   ```
   % cp msgfile /dev/tty 04
   ```

6. No model answer.

7. No model answer.

8. News is more like a newspaper because you do not know who really put it in. You cannot respond to what you read in the news by using that command. You do not know when the item was put in.

9. No model answer.