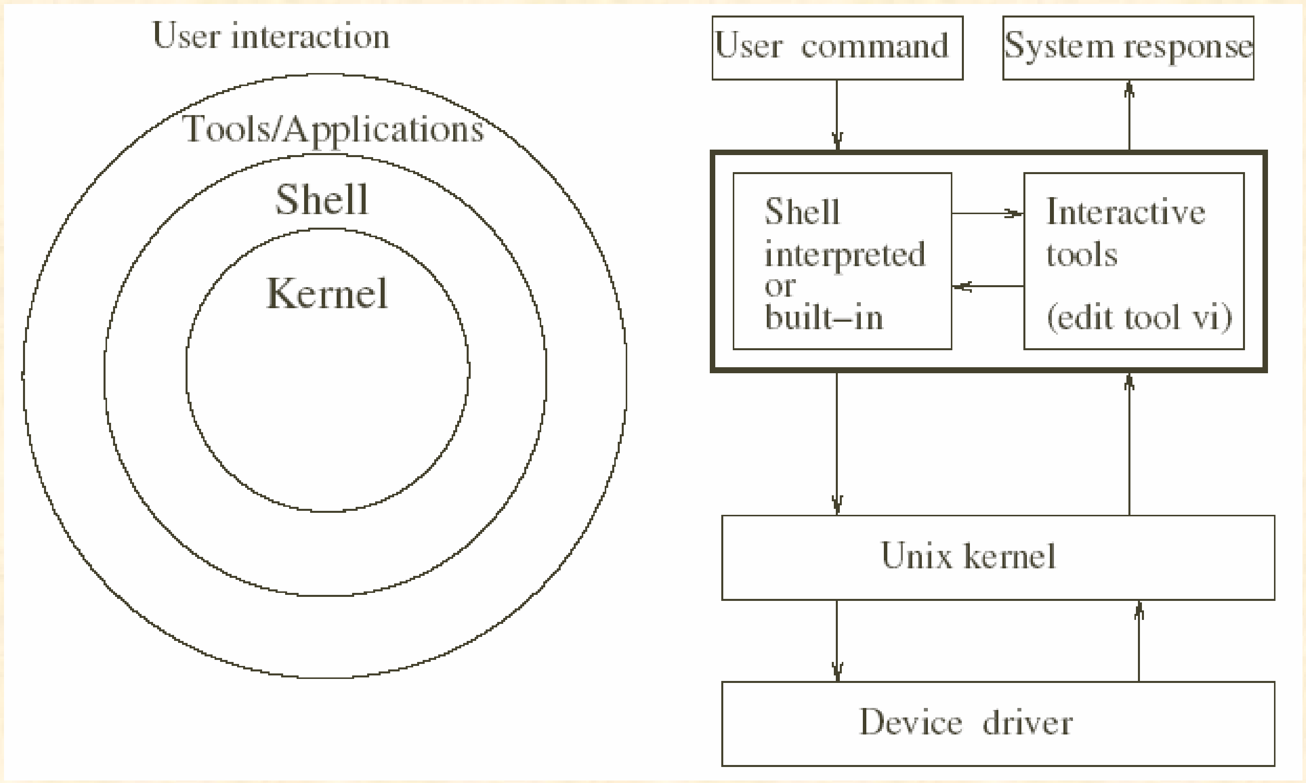
Shell Scripts in UNIX

Prof. P.C.P. Bhatt

Unix System Shell Organization

In the below figure we show how a user interacts with UNIX shell.



Facilities Offered by Unix Shells

- Shell offers a user an interface with the OS kernel.
- Shell distinguishes between the commands and a request to use a tool.
- In addition to interactive command interpreter, it also offers a very useful programming environment.

The Shell Families

- Bourne shell – oldest shell.
- BASH – Bourne Again Shell
- Korn shell
- C shell – c language programming environment
- tchs – more recent version of tchs

Four Step Operational Pattern of Shell

1. Read a command line
2. Parse and interpret it.
3. Invoke the execution of the command line.
4. Go to step 1

Shell scripts – programs written in shell environment.

Subshells

- Command to display the current shell

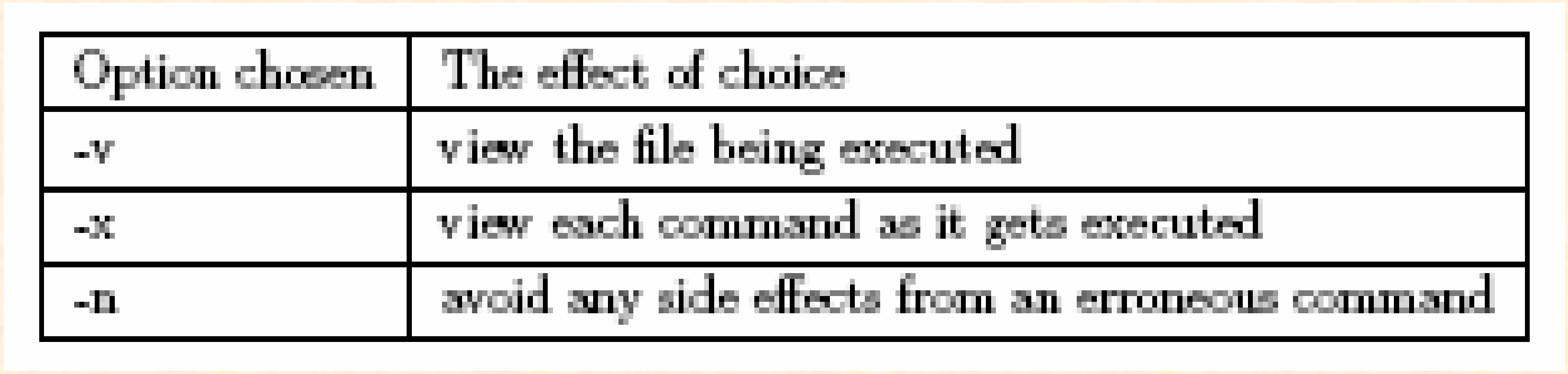
  echo $SHELL.

- *$SHELL* - environment variable storing name of the current shell.
- *set* sets values of environment values.
- *get* shows values of environment values.

Environment Variables

$HOME	User Home directory
$IFS	Internal field separator
$LANG	Directory containing language information
$MAIL	Path containing user's mailbox
$PATH	Colon separated list of directories
$PS1	Prompt for interactive shells
$PS2	Prompt for multi-line command
$SHELL	Login shell environment
$TERM	Terminal type

Some Options



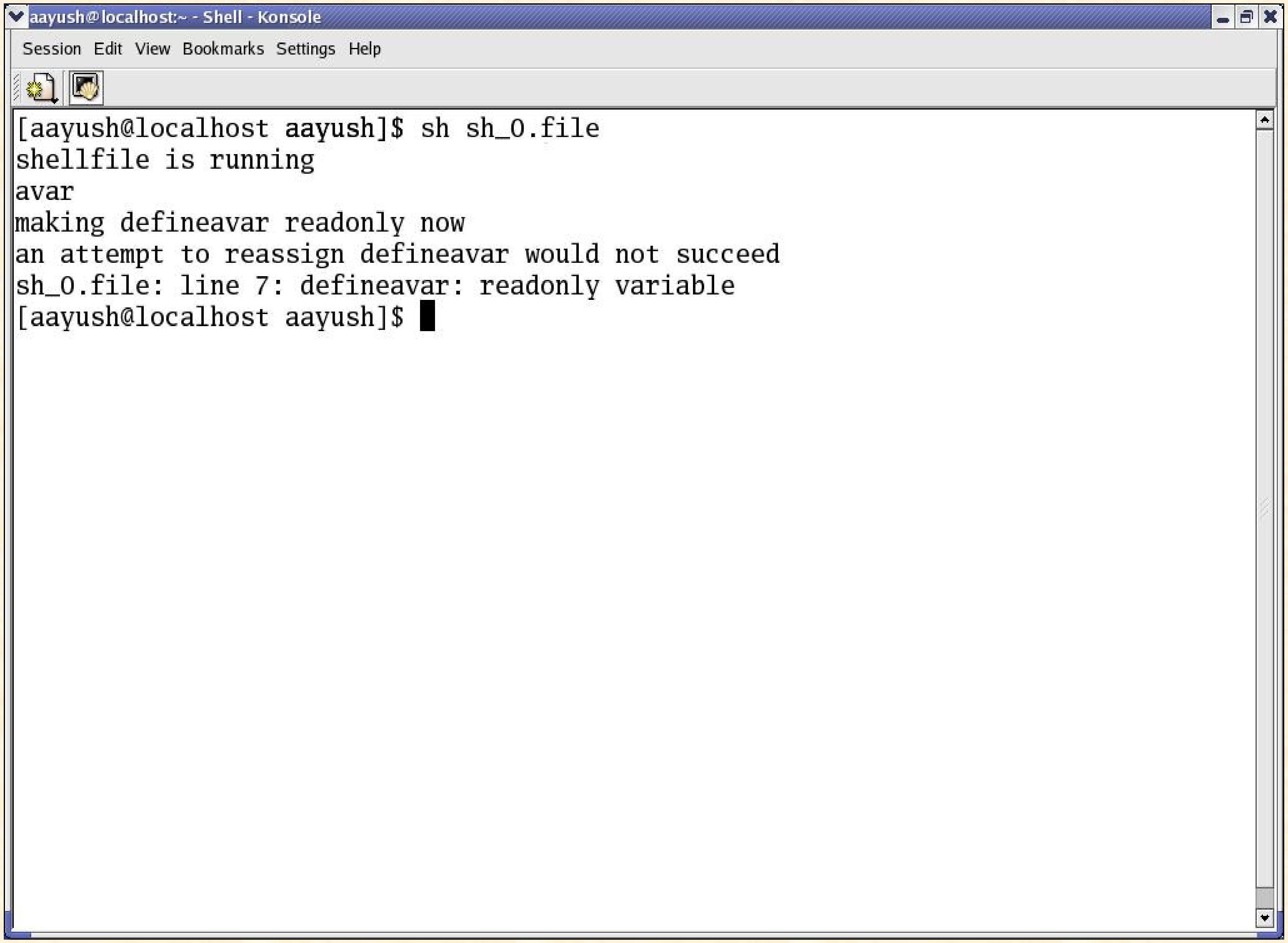
Option chosen	The effect of choice
-v	view the file being executed
-x	view each command as it gets executed
-n	avoid any side effects from an erroneous command

The options with their effects.

Example - 1

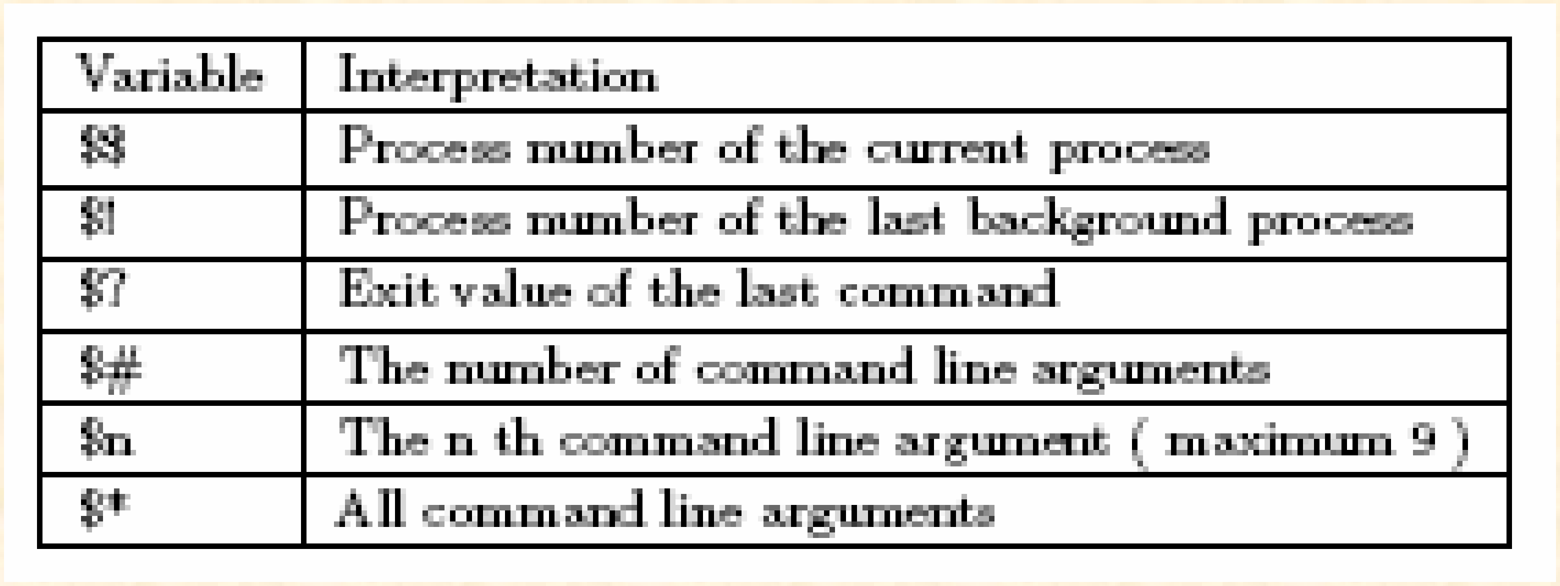
```
# file sh_0.file
echo shellfile is running
defineavar=avar
echo $defineavar
echo "making defineavar readonly now"
readonly defineavar
echo "an attempt to reassign defineavar would not succeed“
defineavar=newvar
```

Example - 1


aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_0.file
shellfile is running
avar
making defineavar readonly now
an attempt to reassign defineavar would not succeed
sh_0.file: line 7: defineavar: readonly variable
[aayush@localhost aayush]$
```

Special Variables



Variable	Interpretation
$$	Process number of the current process
$!	Process number of the last background process
$?	Exit value of the last command
$#	The number of command line arguments
$n	The n th command line argument (maximum 9)
$*	All command line arguments

A partial list of special variables.

Example - 2

# file sh_1.file

# For this program give an input like "This is a test case" i.e. 5 parameters

echo we first get the file name

echo $0

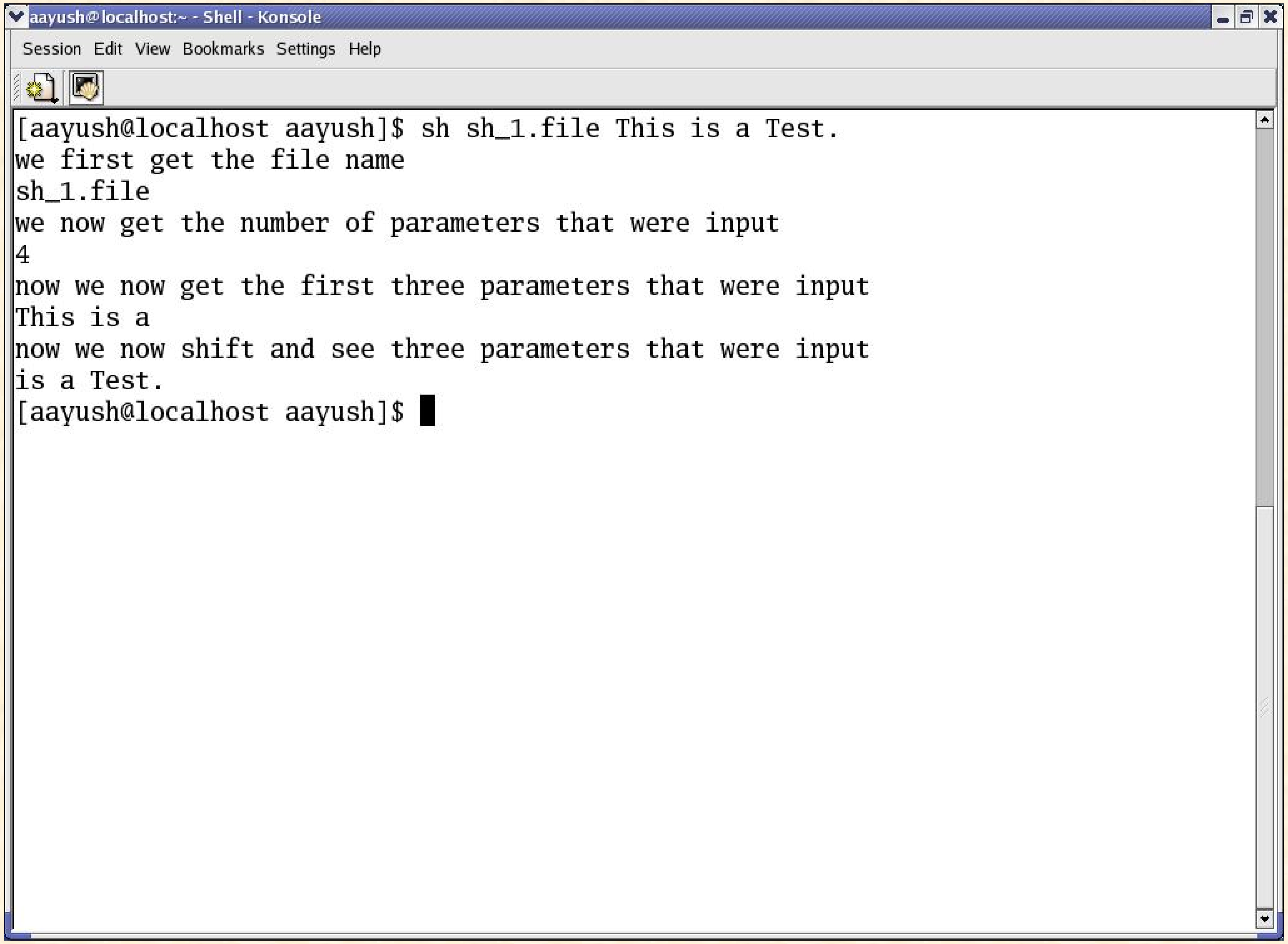
echo we now get the number of parameters that were input

echo $#

echo now we now get the first three parameters that were input echo $1 $2 $3 shift

echo now we now shift and see three parameters that were input echo $1 $2 $3

Example - 2

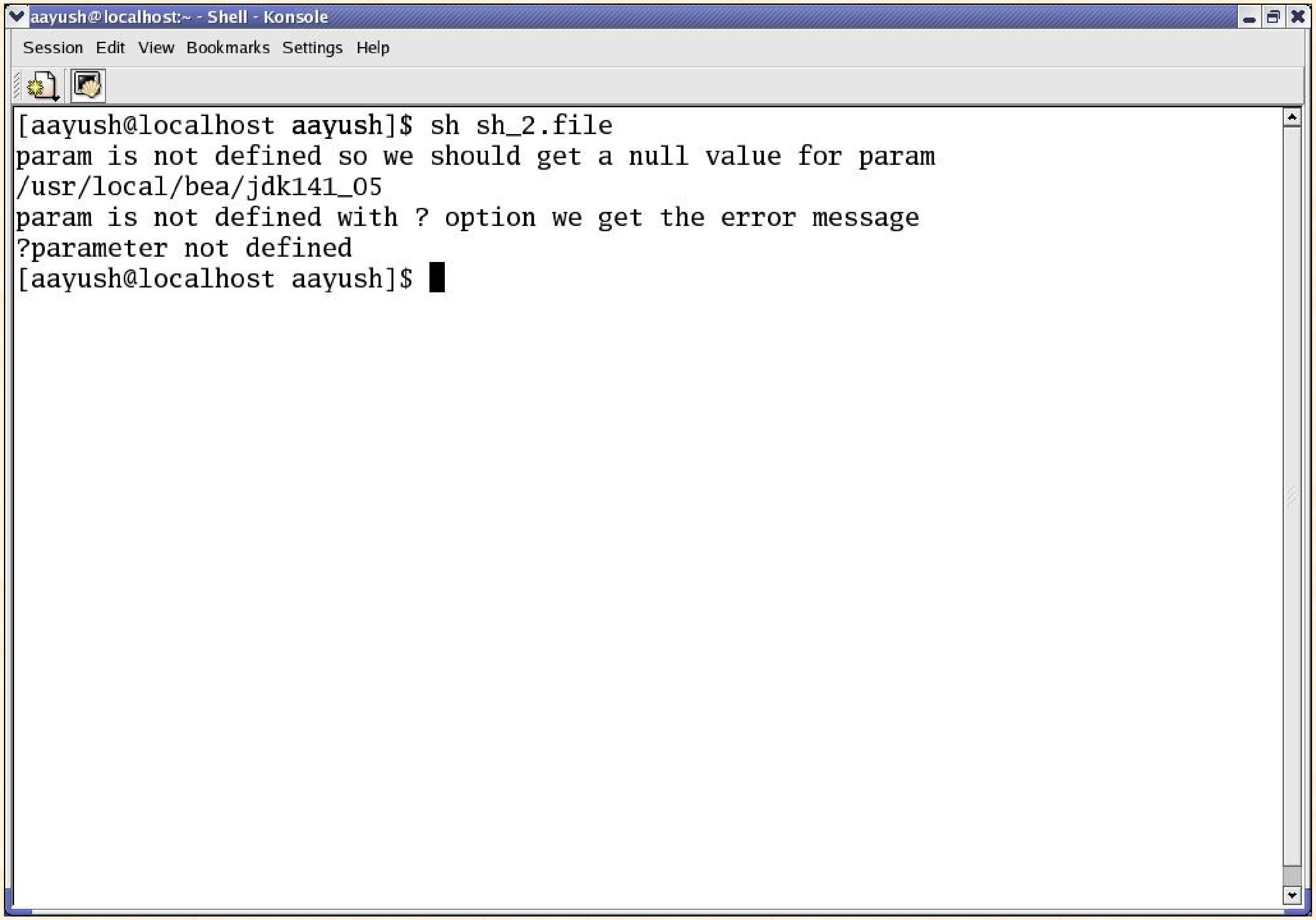

aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_1.file This is a Test.
we first get the file name
sh_1.file
we now get the number of parameters that were input
4
now we now get the first three parameters that were input
This is a
now we now shift and see three parameters that were input
is a Test.
[aayush@localhost aayush]$ 
```

Example - 3

```
# file sh_2.file
# This is to find out if a certain parameter has been defined.
echo param is not defined so we should get a null value for param
echo ${param}
echo param is not defined with "?" option we get the error message
echo ${param?error}
```

Example - 3

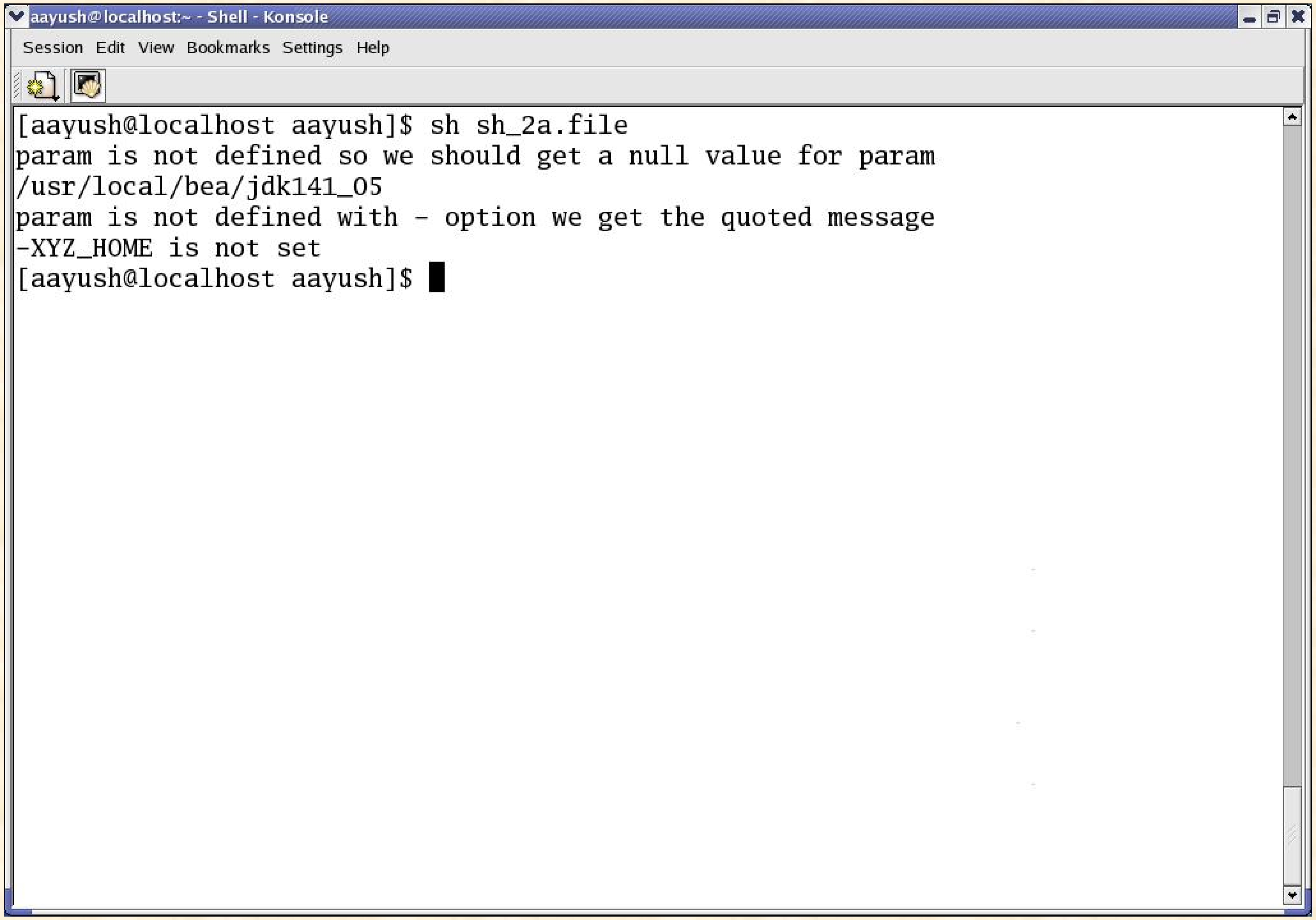

aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_2.file
param is not defined so we should get a null value for param
/usr/local/bea/jdk141_05
param is not defined with ? option we get the error message
?parameter not defined
[aayush@localhost aayush]$ 
```

Example - 4

```
# file sh_2a.file

# This is to find out if a certain parameter has been defined.
echo param is not defined so we should get a null value for
param

echo ${param}

# echo param is not defined with "?" option we get the error
message

# echo ${param?error} echo param is not defined with "-"
option we get the quoted message

echo ${param-'user generated quoted message'}
```

Example - 4

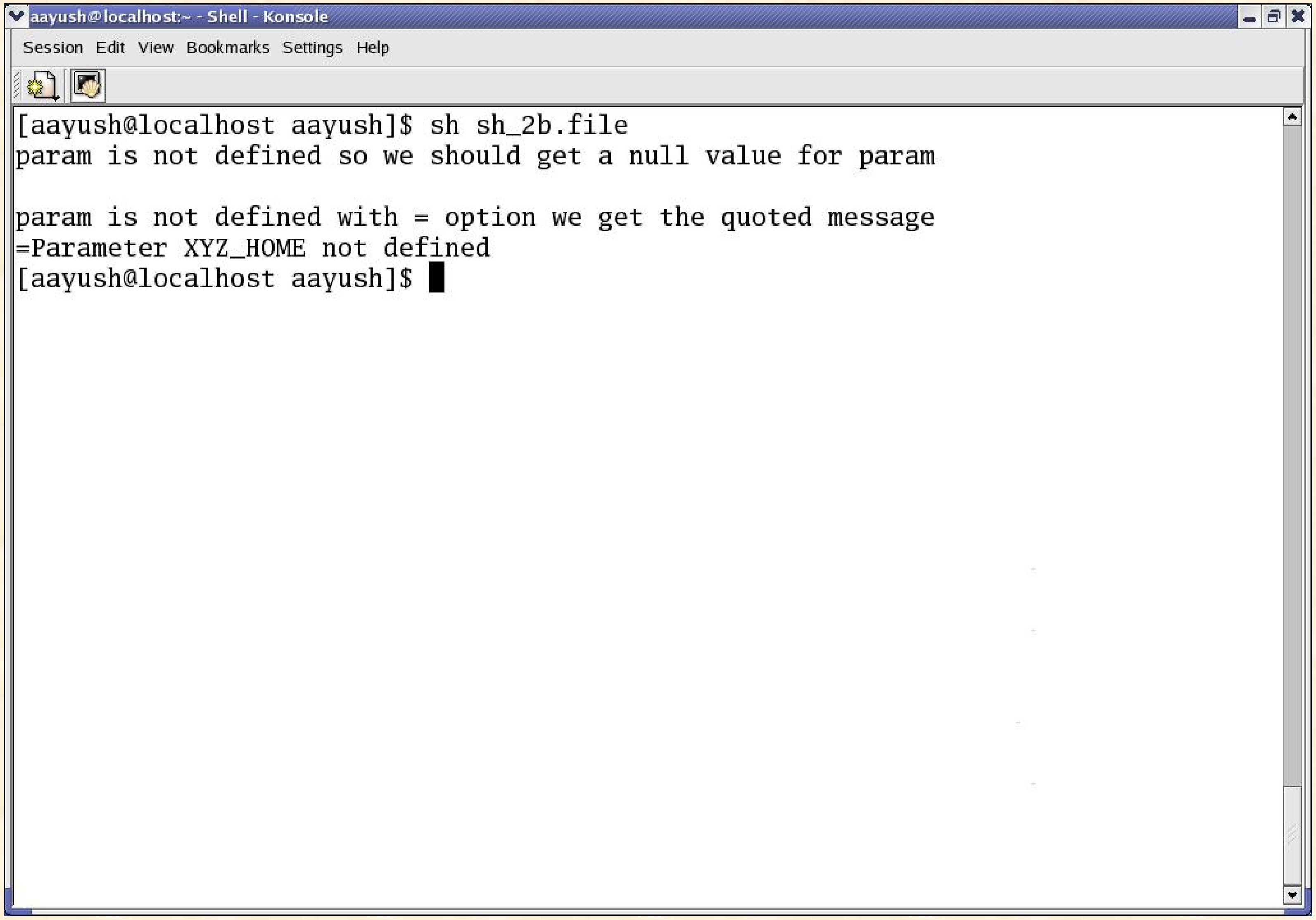

aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_2a.file
param is not defined so we should get a null value for param
/usr/local/bea/jdk141_05
param is not defined with - option we get the quoted message
-XYZ_HOME is not set
[aayush@localhost aayush]$
```

Example - 5

```
# file sh_2b.file

# This is to find out if a certain parameter has been defined.
echo param is not defined so we should get a null value for
param

echo ${param}

# echo param is not defined with "?" option we get the error
message

# echo ${param?error}

echo param is not defined with "=" option we get the quoted
message

echo ${param='user generated quoted message'}
```

Example - 5

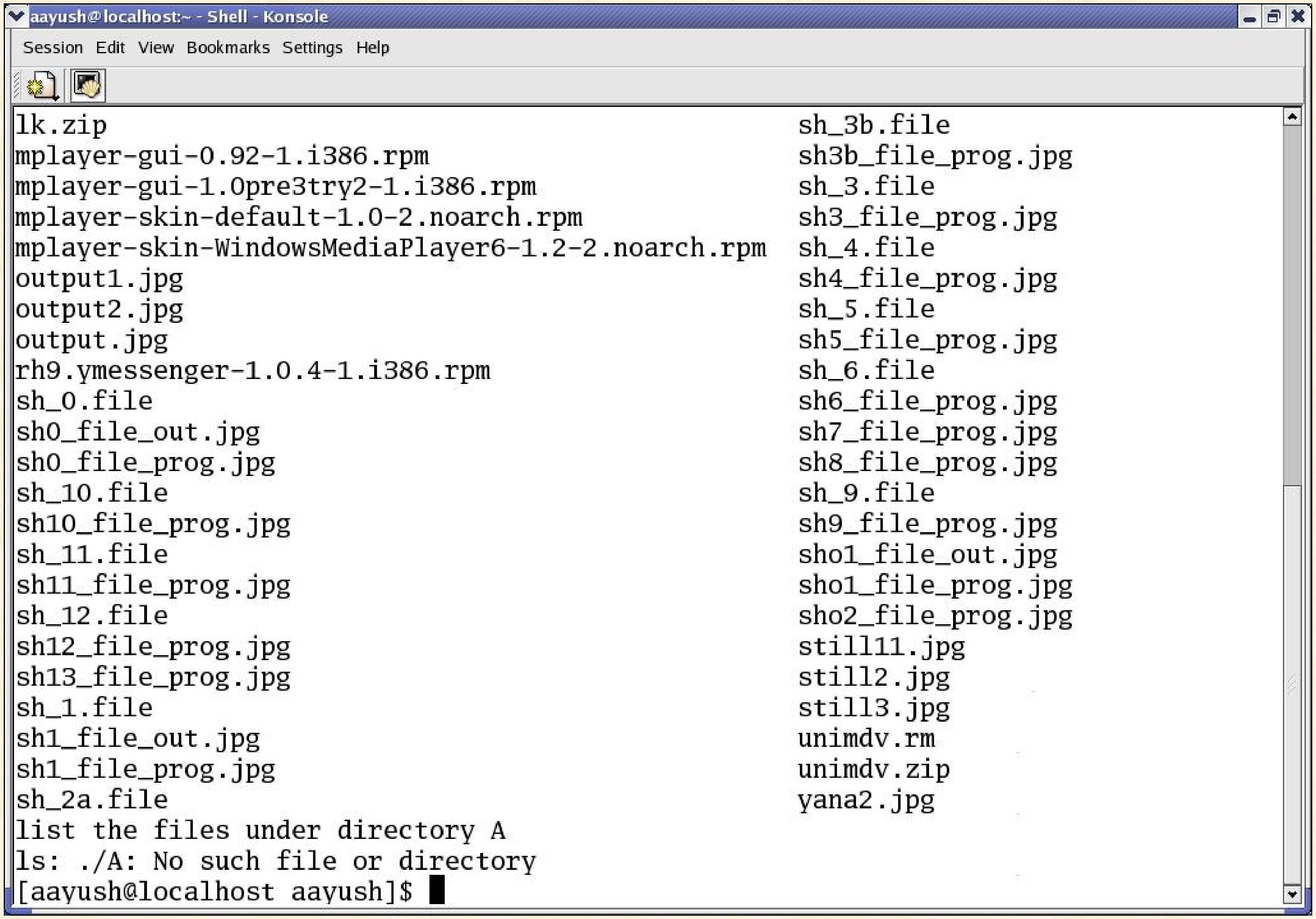

aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_2b.file
param is not defined so we should get a null value for param

param is not defined with = option we get the quoted message
=Parameter XYZ_HOME not defined
[aayush@localhost aayush]$ 
```

Example - 6

```
# file sh_3.file
echo the next line shows command substitution within back quotes
echo I am `whoami`
echo today is `date`
echo there are `who | wc -l` users at the moment
echo var a is now assigned the result of
echo backquoted whoami a=`whoami`
echo we shall output its value next
echo $a
echo also let us reassign a with the value for environment
var HOME
a=`echo $HOME`
echo $a
echo a double dollar is a special variable that stores process id of the shell
echo $$
echo the shell vars can be used to generate arguments for Unix commands
echo like files in the current directory are cur_dir=. ls $cur_dir
echo list the files under directory A ls $cur_dir/A
```

Example - 6 Contd..


aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
lk.zip                                             sh_3b.file
mplayer-gui-0.92-1.i386.rpm                        sh3b_file_prog.jpg
mplayer-gui-1.0pre3try2-1.i386.rpm                 sh_3.file
mplayer-skin-default-1.0-2.noarch.rpm              sh3_file_prog.jpg
mplayer-skin-WindowsMediaPlayer6-1.2-2.noarch.rpm  sh_4.file
output1.jpg                                        sh4_file_prog.jpg
output2.jpg                                        sh_5.file
output.jpg                                         sh5_file_prog.jpg
rh9.ymessenger-1.0.4-1.i386.rpm                    sh_6.file
sh_0.file                                          sh6_file_prog.jpg
sh0_file_out.jpg                                   sh7_file_prog.jpg
sh0_file_prog.jpg                                  sh8_file_prog.jpg
sh_10.file                                         sh_9.file
sh10_file_prog.jpg                                 sh9_file_prog.jpg
sh_11.file                                         sho1_file_out.jpg
sh11_file_prog.jpg                                 sho1_file_prog.jpg
sh_12.file                                         sho2_file_prog.jpg
sh12_file_prog.jpg                                 still11.jpg
sh13_file_prog.jpg                                 still2.jpg
sh_1.file                                          still3.jpg
sh1_file_out.jpg                                   unimdv.rm
sh1_file_prog.jpg                                  unimdv.zip
sh_2a.file                                         yana2.jpg
list the files under directory A
ls: ./A: No such file or directory
[aayush@localhost aayush]$ █
```

Example - 7

```
# file sh_3a.file
# In this file we learn to use quotes. There are three types of quotes
# First use of a single quote within which no substitution takes place
a=5
echo 'Within single quotes value is not substituted i.e $a has a value of $a'
# now we look at the double quote
echo "Within double quotes value is substituted so dollar a has a value of    $a"
echo Finally we look at the case of back quotes where everything is evaluated
echo `$a`
echo `a`
echo Now we show how a single character may be quoted using reverse slash
echo back quoted a is \`a and dollar a is \$a
echo quotes are useful in assigning variables values that have spaces
b='my name'
echo value of b is = $b
```

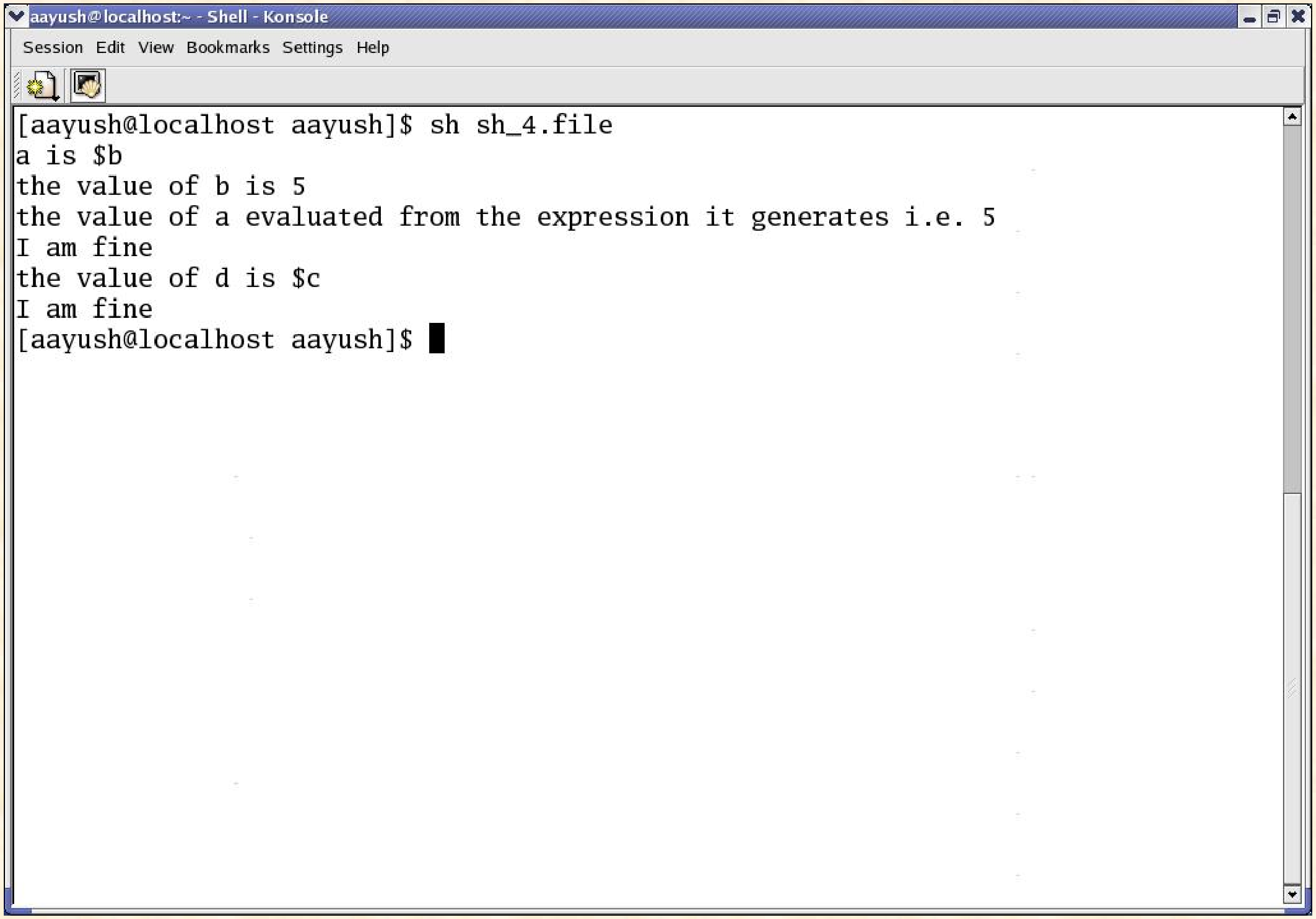
Example - 8

```
# file sh_3b.file

# In this file we shall study the set command. Set lets you

# view shell variable values

echo ---------out put of set -------------- set

echo use printenv to out put variables in the environment
echo ---------out put of printenv -------------- printenv
```

Example - 9

```
# file sh_4.file
# this file shows the use of eval function in the shell
b=5
a=\$b
echo a is $a
echo the value of b is $b
eval echo the value of a evaluated from the expression it generates i.e. $a
c=echo
eval $c I am fine
d=\$c
echo the value of d is $d
eval eval $d I am fine
```

Example – 9 Contd…


aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_4.file
a is $b
the value of b is 5
the value of a evaluated from the expression it generates i.e. 5
I am fine
the value of d is $c
I am fine
[aayush@localhost aayush]$
```

Example - 10

```
file sh_5.file
# This file shows how we may group a process into a detached process
# by enclosing it in parentheses.
# Also it shows use of sleep command
echo basically we shall sleep for 5 seconds after launching
echo a detached process and then give the date
(sleep 5; date)
```

Example - 11

```
# file sh_6.file
# Typically << accepts the file till the word that follows
# in the file. In this case the input is taken till
# the word end appears in the file.
#
# This file has the command as well as data in it.
# Run it : as an example : sh_6.file 17 to see him 2217 as output.
# $1 gets the file argument.
grep $1<<end        /* grep is the pattern matching command in Unix */
me 2216
him 2217
others 2218
end
```

Example - 12

The basic pattern of the if command is just like in the programming languages. It is:

```
if condition
then
command_pattern_for_true
else
command_pattern_for_false
Fi
```

```
# file sh_7.file
if ls my_file.ps
then lpr -Pbarolo-dup my_file.ps /* prints on printer barolo on both sides */
else echo "no such file in this directory"
fi
```

Example - 13

```
# file sh_7a.file
# This file demonstrates use of case
# In particular note the default option and usage of selection
# Note the pattern matching using the regular expression choices.

case $1 in
[0-9]) echo "OK valid input : a digit ";;
[a-z]|[A-Z]) echo "OK valid input : a letter ";;
*) echo "please note only a single digit or a letter is valid as input";;
esac
```

Example - 14

```
# file sh_8.file
# In this file we illustrate use of for command
# It may be a good idea to remove some file called
# dummy in the current directory as a first step.
#
echo removing dummy
rm dummy
for i in `ls`; do echo $i >> dummy; done
grep test dummy
```

Example - 15

Now we shall demonstrate the use of ***expr*** command. This command offers an opportunity to use integer arithmetic as shown below :

```
b=3
echo value of b is = $b
echo we shall use as the value of b to get the values for a
echo on adding two we get
a=`expr $b + 2`
echo $a
```

Example - 16

```
# file sh_9a.file
# this file illustrates the use of expr and test commands
b=3
echo on adding two we get
a=`expr $b + 2`
echo $a
echo on multiplying two we get
a=`expr $b \* 2` /* Note the back slash preceding star */
# We shall see the reason for using back slash before star in the next example
echo $a
test $a -gt 100
$?
test $a -lt 100
$?
test $a -eq 6
$?
test $a = 6
$?
test $a -le 6
$?
test $a -ge 6
$?
test $a = 5
$?
if (test $a = 5)
then echo "found equal to 5"
else echo "found not equal to 5"
fi
test $a = 6
if (test $a = 6)
then echo "the previous test was successful"
fi
```

Example - 17

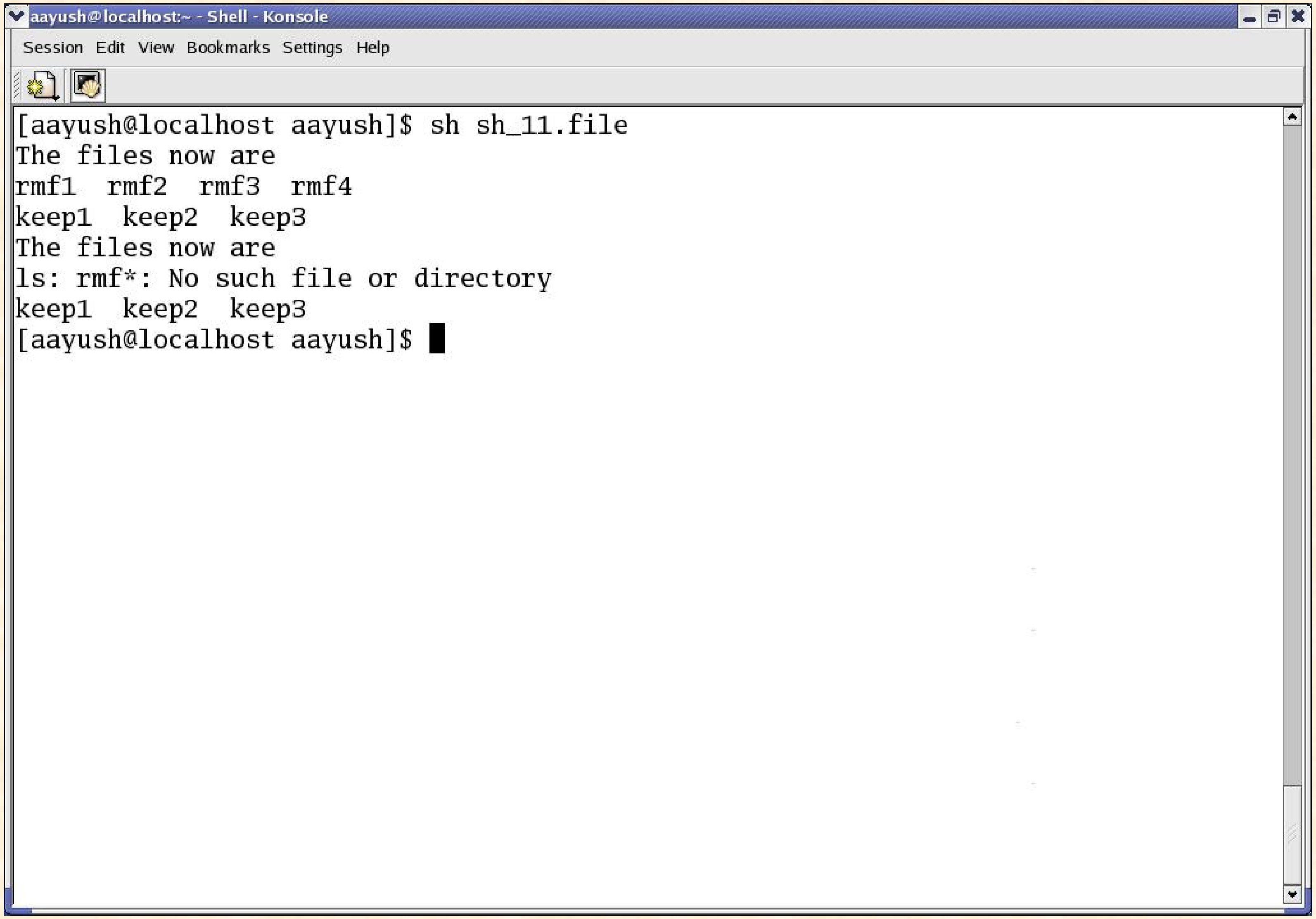
Now we shall use some regular expressions commonly used with file names.

```
# file sh_10.file
# in this program we identify directories in the current directory
echo "listing all the directories first"
for i in *
do
if test -d $i
then echo "$i is a directory"
fi
done
echo "Now listing the files"
for i in *
do
if test -f $i
then
echo "$i is a file"
fi
done
echo "finally the shell files are"
ls | grep sh_
```

Example - 18

```
# file sh_11.file

# In this file we learn about the trap command. We will first

# create many files with different names. Later we will remove # some of
these by explicitly trapping

touch rmf1

touch keep1

touch rmf2

touch rmf3

touch keep2

touch rmf4

touch keep3

echo "The files now are" ls rmf* ls keep* trap `rm rm*;

exit` 1 2 3 9 15

echo "The files now are" ls rmf* ls keep*
```

Example - 18 Contd..


aayush@localhost:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
[aayush@localhost aayush]$ sh sh_11.file
The files now are
rmf1  rmf2  rmf3  rmf4
keep1  keep2  keep3
The files now are
ls: rmf*: No such file or directory
keep1  keep2  keep3
[aayush@localhost aayush]$
```

Example - 19

Now we assume the presence of files of telephone numbers. Also, we demonstrate how Unix utilities can be used within the shell scripts.

```
# file sh_12.file
# In this file we invoke a sort command and see its effect on a file
# Also note how we have used input and output on the same line of cmd.
sort < telNos > stelNos
# We can also use a translate cmd to get translation from lower to upper case
tr a-z A-Z < telNos > ctelNos
```