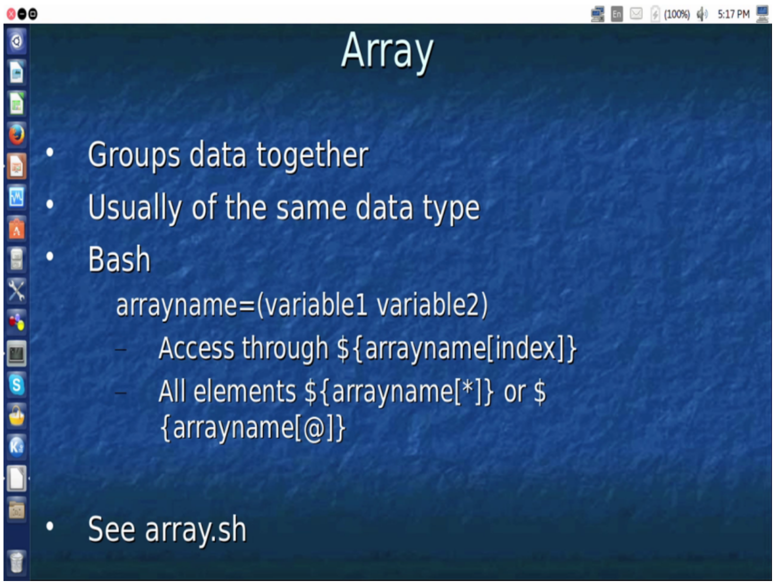
Information Security
Sri M J Shankar Raman,
Consultant of Computer Science and Engineering
Indian Institute of Technology Madras
Module 30
Array

Hi welcome back to the session on Shell programming in the last few sessions we had talked about variables we had talked about a brief introduction to shell we had also talked about declaring scalar variables. Now in this session we will talk about how to declare arrays in a shell.

(Refer Slide Time: 00:38)



Array

- Groups data together
- Usually of the same data type
- Bash
  arrayname=(variable1 variable2)
  - Access through ${arrayname[index]}
  - All elements ${arrayname[*]} or ${arrayname[@]}
- See array.sh

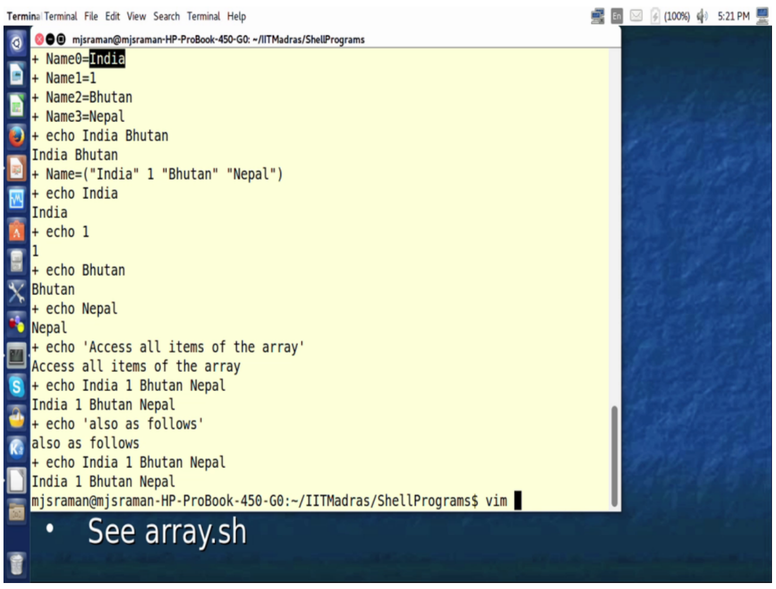
The context of arrays in shell is slightly different from the context that you use in other programming languages. In general an array groups a bunch of data that belong to the same type together. So if you take a programming language like C you will learn something as integer array.

Now in the case of Shell as I told you cannot have such kind of data types stored seperately therefore what usually happens in the shell is you call an array as grouping of data together. So in bash one again shell we can declare an array by using array name so if you look at this some giving some name equal to and then you open this curved brace and then you give variable name space variable 2 etc and you can access each element of the array through this notation.

This notation is slightly complex one all things that we should note is since array groups data items together each element in the array is retrieved using an index. So that is what this index is this is usually an integer and this is the name of the array so this matches whatever we had given here and then we know that dollar is used to take the values of the variables.

Therefore we put a dollar withing this braces. And you can one of the good aspects of born shell is that you can access all the elements of the array at the same time by using either a star symbol here as a index or the @ symbol as a index. So let us take a look at how we can use arrays in programming?

(Refer Slide Time: 02:41)


Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
+ Name0=India
+ Name1=1
+ Name2=Bhutan
+ Name3=Nepal
+ echo India Bhutan
India Bhutan
+ Name=("India" 1 "Bhutan" "Nepal")
+ echo India
India
+ echo 1
1
+ echo Bhutan
Bhutan
+ echo Nepal
Nepal
+ echo 'Access all items of the array'
Access all items of the array
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
+ echo 'also as follows'
also as follows
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim
```

- See array.sh

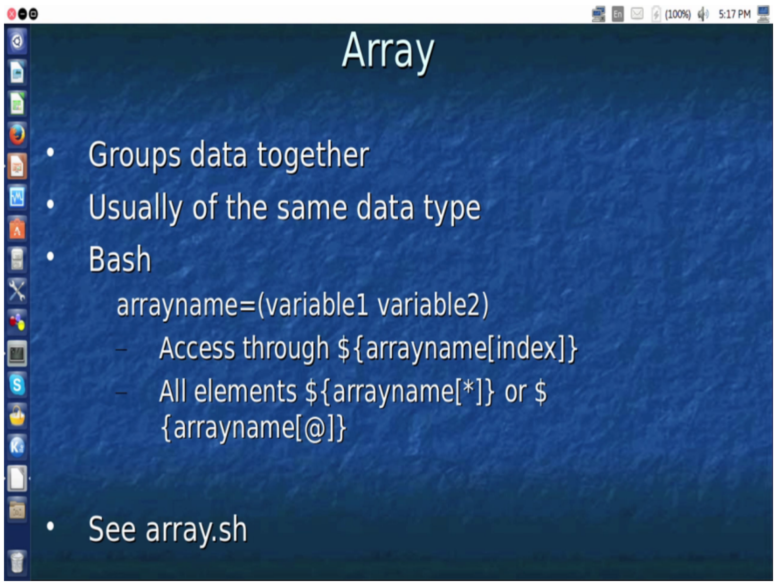
So we are having a program called arrays dot sh. So in this case if we are going to use only scalar quantities you see that I have to declare variable something like this Name 0 is equal to India, Name 1 is equal to 1 and this is what I was talking about what you see this is a string and this can be treated as a number, so in this case and you also have name 2 which is Bhutan then we have named 3 is equal to Nepal.

So if you print this if you run this program and print this and if you want to access the value of Name 0 then we can write As given in the echo statement dollar name 0 Remember this 0 is not the index it is just a naming convention this is a variable name. So in this case for example this line should print India and Bhutan.

So let us try to execute this program and see whether it prints India and Bhutan and as you see in this line so it prints India and Bhutan this is the line and we can also demonstrate using sh minus x command. It first prints India and Bhutan so you have to execute using the bash shell.

So if you look at this it just when you put Echo India and Bhutan it just prints India and Bhutan ok, so in this way these are called scalars because they hold only 1 quantity. Now coming back to the rest of the code we see that at line number 9 we are trying to initialize the array, the array name.

(Refer Slide Time: 04:30)



Array

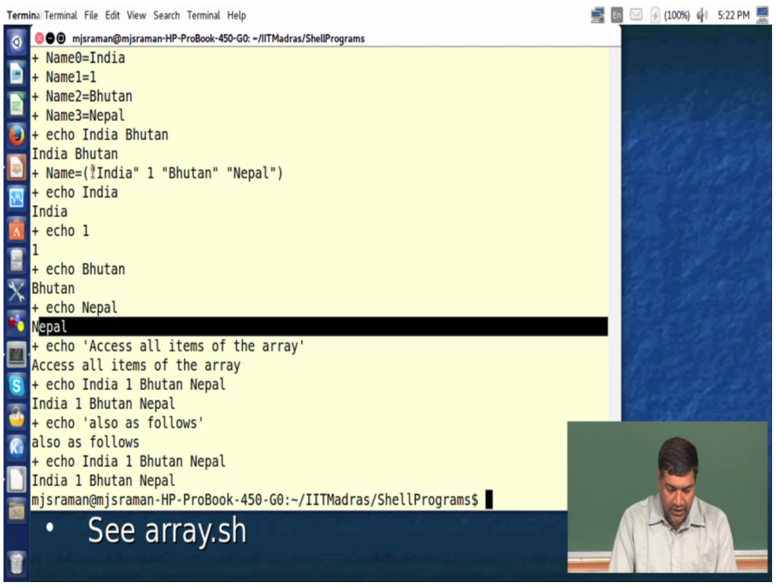
- Groups data together
- Usually of the same data type
- Bash

  arrayname=(variable1 variable2)
  - Access through ${arrayname[index]}
  - All elements ${arrayname[*]} or ${arrayname[@]}

- See array.sh

as we have seen in this slide this is the array name so if you see this and it is equal to and then we have the variable values of the variables.

(Refer Slide Time: 04:39)



```
+ Name0=India
+ Name1=1
+ Name2=Bhutan
+ Name3=Nepal
+ echo India Bhutan
India Bhutan
+ Name=("India" 1 "Bhutan" "Nepal")
+ echo India
India
+ echo 1
1
+ echo Bhutan
Bhutan
+ echo Nepal
Nepal
+ echo 'Access all items of the array'
Access all items of the array
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
+ echo 'also as follows'
also as follows
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

- See array.sh

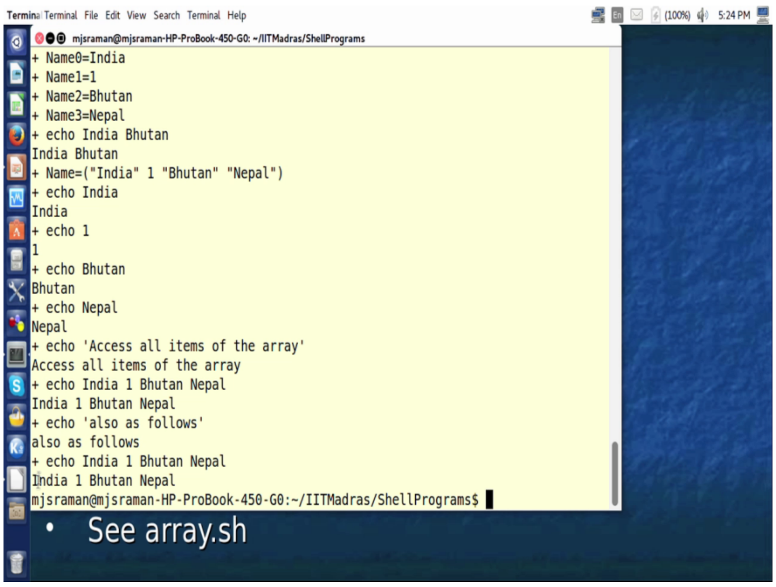
So here it is the array name equal to and then you can see the values of variables for India 1 Bhutan I mean whatever I had done here I have grouped them together that is the only difference between the variables been assigned the values here and the variables been assigned the values here. However the way we access the values is different from the ways so here.

If you look at this we just access those variables values using echo dollar name 0 and then name 2 etc but in this case we use something known as an index. So the index starts with 0 then this becomes 1 this becomes 2 and this becomes 3. Therefore this will print India 1 Bhutan and Nepal

So let us see a demo of this If we look at this so we're assigning the array to be India 1 Bhutan and Nepal and then it echoes India first this is all the name of zero then it Echoes India then dollar name of $1 name of two $1 name of three table 2 print all the values using this is dollar array name of 0

Then it echoes India then dollar name of 1 which corresponds to this, dollar name of 2 which corresponds to Bhutan and dollar name of 3 which corresponds to Nepal here. the index of the latest now locate example in C axis and start if I can use this implies that and if you remember that put 10 - x. in the morning. So in this way we are able to print all the four values using the index of the array.

(Refer Slide Time: 06:08)



```
+ Name0=India
+ Name1=1
+ Name2=Bhutan
+ Name3=Nepal
+ echo India Bhutan
India Bhutan
+ Name=("India" 1 "Bhutan" "Nepal")
+ echo India
India
+ echo 1
1
+ echo Bhutan
Bhutan
+ echo Nepal
Nepal
+ echo 'Access all items of the array'
Access all items of the array
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
+ echo 'also as follows'
also as follows
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

• See array.sh

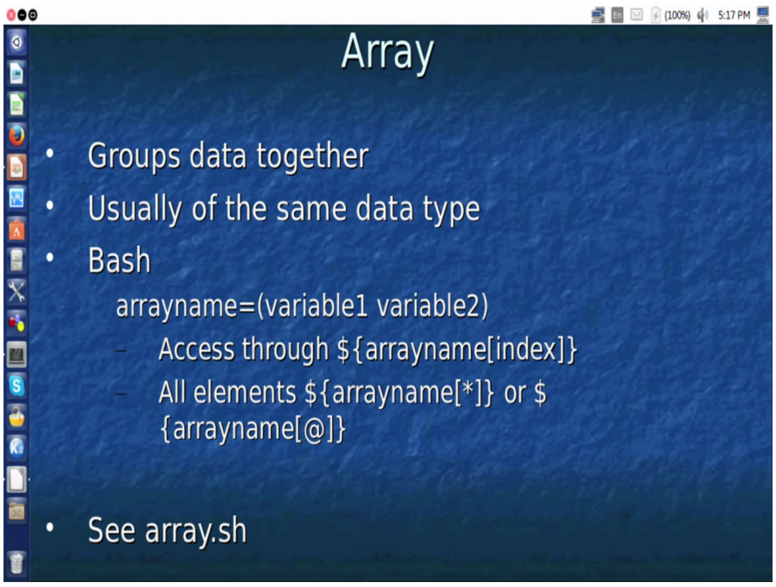
Let us now look into this example once more and then see the usage of the other two access methods. So if you look at this I had introduced another access methods which uses dollar array name and then star so if you look at this if i want to access all the elements of the array.

I can use echo dollar name and within this I can put a star which implies that this will access all the elements of the array either you can follow this method or you can follow this method that is minor difference between these two which we will discuss in the subsequent class.

So if we run this program again and then we can see how these two these four line work so what we do is we just run and if you remember we had but bash minus x which means we are running it in a debugging mode so that we can see what is happening so if we look accessing all our orients so it prints access all items of the array then it prints India 1 Bhutan and Nepal and also they can print as follows India 1 Bhutan and Nepal.

So in this way we will be able to make use of the array variables okay in our programs. Probably what is the use of these arrays we will see for example if we want to store some table of values etc these arrays can be used.

(Refer Slide Time: 07:40)



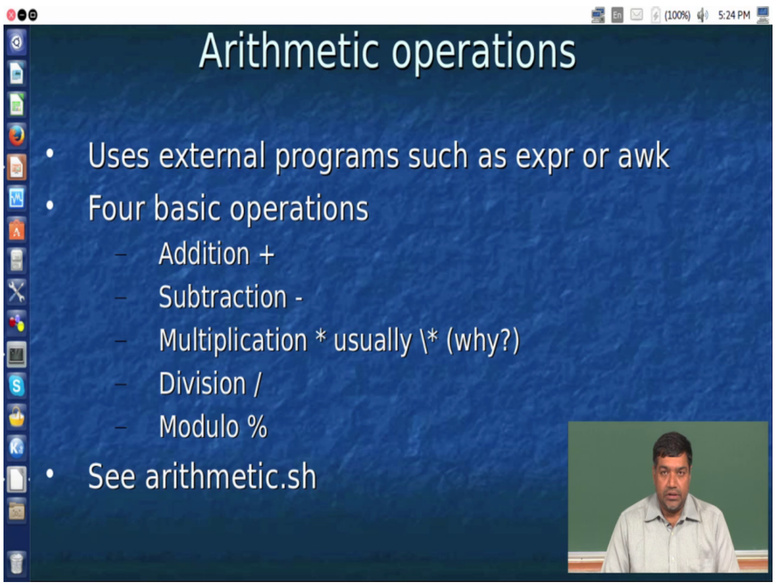
Array

- Groups data together
- Usually of the same data type
- Bash

  arrayname=(variable1 variable2)
  - Access through ${arrayname[index]}
  - All elements ${arrayname[*]} or ${arrayname[@]}
- See array.sh

We will now move on to the next part of our presentation which now deals with arithmetic operations.

(Refer Slide Time: 07:51)



Arithmetic operations

- Uses external programs such as expr or awk
- Four basic operations
  - Addition +
  - Subtraction -
  - Multiplication * usually * (why?)
  - Division /
  - Modulo %
- See arithmetic.sh

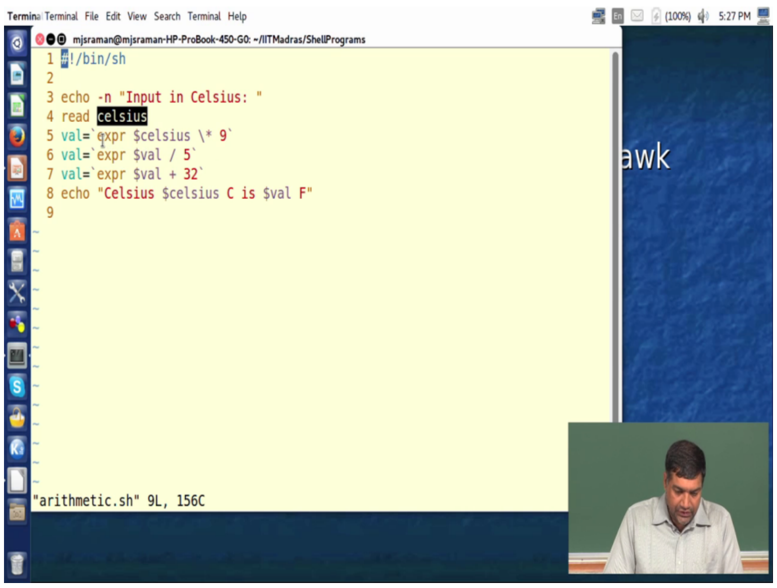
So arithmetic operations see what are the things that we should know with shell is there is no direct support for arithmetic operations with shell. So what it does is it uses external Programs such as expr or a much more powerful version of a calculator program called bc or it makes use of awk So what we do essentially is inside the shell we call one of these programs and such programs will help us perform arithmetic operations.

Shell actually supports 5 basic arithmetic operations, so it is addition subtraction division and modular operations which is essentially similar to any other programming language like C whereas in multiplication we used star even though multiplication represented by star we are supposed to usually use backslash star.

I leave it to you people to answer why we are going to use backslash star and not star one of the reasons, there are many others one of the answers is that if you look at this presentation you see the star was used to access all the elements therefore syntactically it carries two meanings, one meaning is it can access all the elements of an array and the other meaning if I put star that means that you have to access all the elements rather than multiplying a number.

Therefore we put a backslash this is known as a escape character we put the backslash and then we put the star to do the calculations so let us take a look at the integer arithmetic first and for integer arithmetic we will use this expr an external program.

(Refer Slide Time: 09:48)


```
#!/bin/sh

echo -n "Input in Celsius: "
read celsius
val=`expr $celsius \* 9`
val=`expr $val / 5`
val=`expr $val + 32`
echo "Celsius $celsius C is $val F"

"arithmetic.sh" 9L, 156C
```
Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms
(100%) 5:27 PM
awk

So what we do now is we will take this program called arithmetic dot sh and then see this program go through this program and then we will execute this program and see how this program works. So the first line of this program we had already seen if we see echo minus n it means it will print this and then the cursor will not move down to the next line it will actually expect the input to be given here which essentially means you are providing a nice user interface and then we have already seen that the read statement can be used to take the value input values from the user.

And then now we will come to the calculation part if you look at this we are now using a variable called val and what we are doing is see remember this is a bad code this is very very important when you are typing any command or any expression in shell you need to be very careful about the spaces for example I cannot type it like this if I do not give space between these two then shell will actually present an error.

So we have to be extremely careful when you type it so its better you have your laptop in front of you and do not cut you do not cut and paste these kind of strings you actually type it and see because that can lead to lot of syntax related errors. And you need to get practice to ensure that you donot get such kind of syntax related errors.

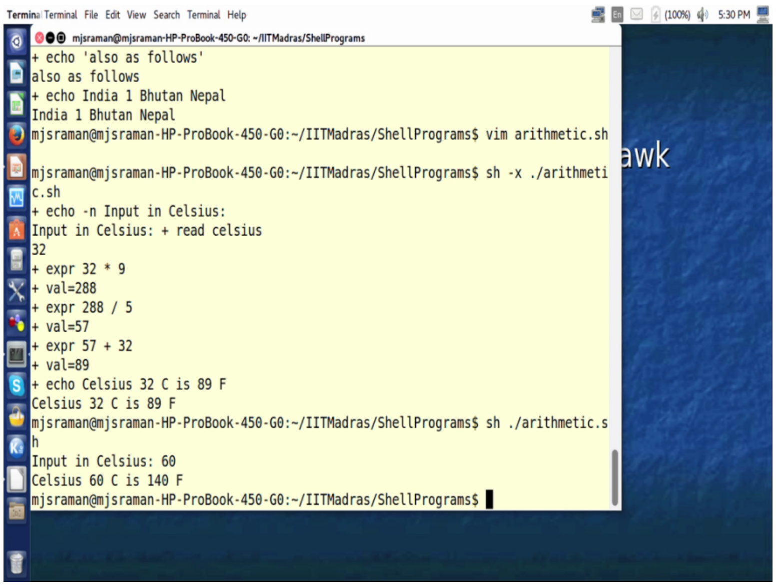
If you look at this you see this Celsius is now retrieved so whatever values the user is given is now retrieved and then this quantity is multiplied by 9 a constant, of course all these things could have been done in a single line but we just want to show you how mathematical calculations can be done with shell.

So if I look at this I have a back code here in line number 5 then I have another back code here and if you look at this what this back code says is that after you calculate this expression of Celsius into 9 then assign the result whatever the result you get out of this expression to val.

So the val will now get assigned what I suppose let us assume that you give a Celsius of 0 then 0 into 9 is 0 and the result of 0 is assigned to val. Now the next line says that val should be divided by 5 now remember I have used all integer values, you may have a question on what I should do if I have a floating point value we will answer that question but initially we will see the integer arithmetic and expression does not help you do this floating point arithmetic so let us go back and see.

So if you look at this if i put a value of 0 and then divide it by 5 it will again give a value of 0. And since we have put a back code here if you look at this we have put a back code therefore the value 0 again is transferred to val. And again in the next line I am using that val of 0 plus adding 32 to it and then reassigning it to val and finally in line number 8 I am printing what is the value of Celsius in Fahrenheit. so let us run this program and see how what is the result it produces we will run in debug mode and then finally and then finally we will run it again in the normal mode.

(Refer Slide Time: 13:13)

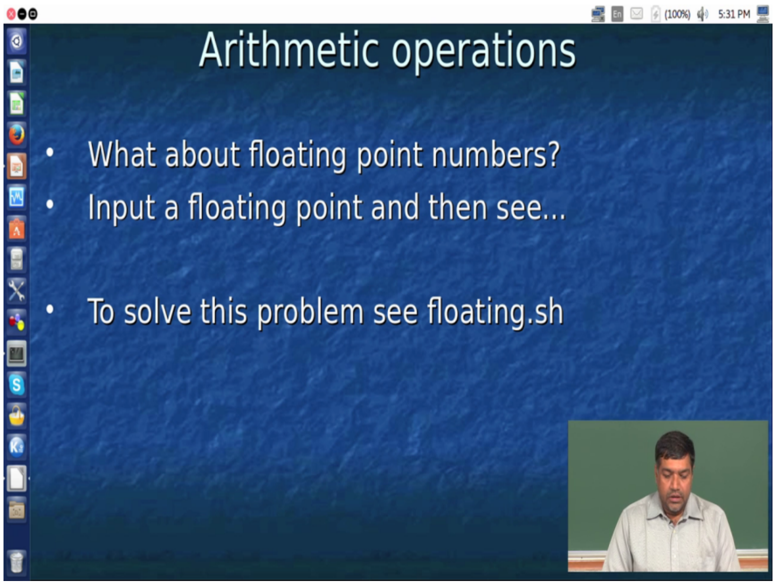


```
+ echo 'also as follows'
also as follows
+ echo India 1 Bhutan Nepal
India 1 Bhutan Nepal
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim arithmetic.sh

mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh -x ./arithmeti
c.sh
+ echo -n Input in Celsius:
Input in Celsius: + read celsius
32
+ expr 32 * 9
+ val=288
+ expr 288 / 5
+ val=57
+ expr 57 + 32
+ val=89
+ echo Celsius 32 C is 89 F
Celsius 32 C is 89 F
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh ./arithmetic.s
h
Input in Celsius: 60
Celsius 60 C is 140 F
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So I am taking arithmetic dot sh so it tries to read a value let us say i give a value of 32 now if you look at this the whatever value 32 it gave is multiplied by 9 and the result of 288 is assigned to val. Again 288 is divided by 5 and then assigned to val 288 divided by the remainder 3 is ignored.

And then again this 57 plus 32 is added and the value 89 is printed. So, and finally Celsius 32 degree is 89 Fahrenheit if I do not run using the sh minus x option you will get something like this. So I get say 60 degrees and I get 140 degrees Fahrenheit. So in this way we can do the integer arithmetic operations in a shell.

(Refer Slide Time: 14:06)

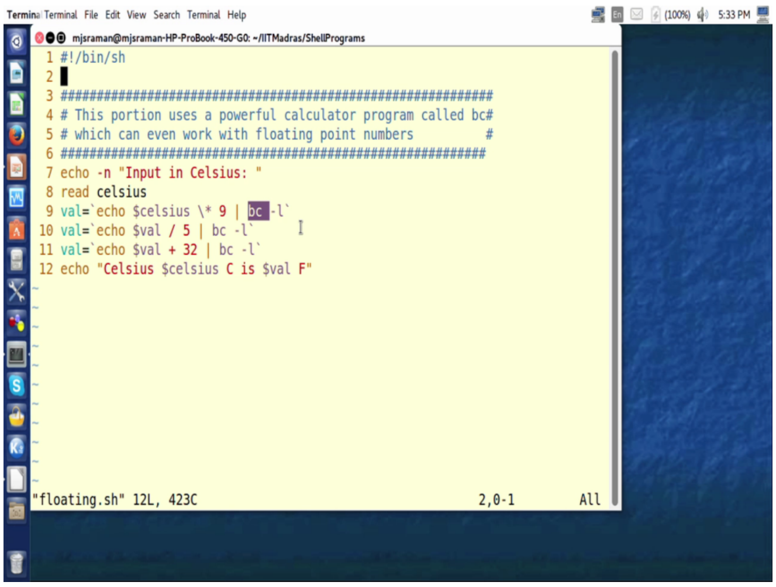


Arithmetic operations

- What about floating point numbers?
- Input a floating point and then see…
- To solve this problem see floating.sh

Now this raises the next question of how do I do the floating point operations. So for doing floating point operations what we can do is we can change our tool, so this is one good part of shell since shell does not provide support for this plus minus directly what we can do is we can make use of other tools that are available in Linux or UNIX and use those tools to do this mathematical operation so we will see one such example of how to do floating point operations with a tool called BC. :

(Refer Slide Time: 14:42)


Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

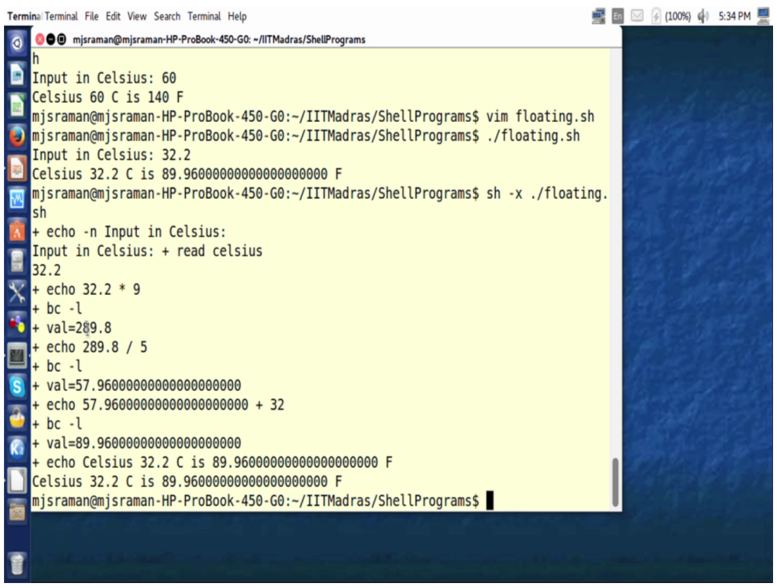
```
#!/bin/sh

############################################################
# This portion uses a powerful calculator program called bc#
# which can even work with floating point numbers          #
###########################################################
echo -n "Input in Celsius: "
read celsius
val=`echo $celsius \* 9 | bc -l`
val=`echo $val / 5 | bc -l`
val=`echo $val + 32 | bc -l`
echo "Celsius $celsius C is $val F"
~
"floating.sh" 12L, 423C                                2,0-1         All
```

So let us look at a code for this. So as I told you in the first class its better we actually comment the code and if you look at this portion we actually comment the code which says that this portion is powerful calculator program called BC which can even work with floating point numbers. So previously if you remember the one of the things that you should be careful is the syntax will change if you are going to use the program like BC or you are going to use a program like expr.

So the first two lines of the program are the same, there are no problems and the change starts occurring in the third line. So this is the way you have to give an input to the BC program. So what I do is I just echo whatever mathematical operation i want to do is echo to BC minus l so you can take a look at what is BC or Bc minus l using the help information now what happens is once you do this even though you are given the values as numerals or an integers here you can see that BC will actually converted into floating point and then the final result will be presented in a floating point value.

So this is the same program as before what we have done as before except that we are using a new tool called BC which is more powerful and we can do floating point operations based on this.

(Refer Slide Time: 16:05)


Terminal Terminal File Edit View Search Terminal Help
5:34 PM
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
h
Input in Celsius: 60
Celsius 60 C is 140 F
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim floating.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ./floating.sh
Input in Celsius: 32.2
Celsius 32.2 C is 89.96000000000000000000 F
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh -x ./floating.
sh
+ echo -n Input in Celsius:
Input in Celsius: + read celsius
32.2
+ echo 32.2 * 9
+ bc -l
+ val=289.8
+ echo 289.8 / 5
+ bc -l
+ val=57.96000000000000000000
+ echo 57.96000000000000000000 + 32
+ bc -l
+ val=89.96000000000000000000
+ echo Celsius 32.2 C is 89.96000000000000000000 F
Celsius 32.2 C is 89.96000000000000000000 F
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ █
```

So let us run this program and see how it executes, so what I can do is first i will run this program without any debugging mode and then we will see so I have taken an input value so we will give the input value to let us say 32.2 degree Celsius and then it prints if you print the values it prints as 89.9600 degree Fahrenheit.

I would leave it to you as an exercise you should look at how you can reduce all these 0 or round it off to nearest 2 decimal places three decimal places etc. I leave it as an exercise to you but in this way if you see let us do the same program by in the debugging mode so that we can understand what is happening.

So I say give the same input 32.2 and if you look at this once I give it 32.2 it just gets multiplied by 9 so you get 289.8 and then this gets divided by 5 which leads a value of 57.96 and 57.96 gets added to 32 and finally I get a value of 89.96 and this value gets printed.

So in this way we will be able to print the floating point numbers also using shell. So in short let us recollect what we have done the first thing that we found out was we can declare variables using arrays and the second thing that we discussed was we can actually do integer based arithmetic operations using powerful calculator programs that are already available on the shell.

The first program that we used was expr where we tried to do integer arithmetic operations the second program that we tried to use was BC infact if you look at BC it has lots and lots of features. So this becomes actually very very powerful calculator program. And what we have done is that using BC we have covered the floating point Arithmetic operations.

Now can we merge arrays into all the variables that are given or scalar variables which take only one value. So the question is can I use arrays and do the same operation yes you can use arrays if you are going to calculate for example I want to calculate all the values from 0 to 100 and convert them into Fahrenheit just it is possible you can declare them in an array and then you can run this program on the array and print out the result and store the result.

In the next session we will talk about the relational operators.