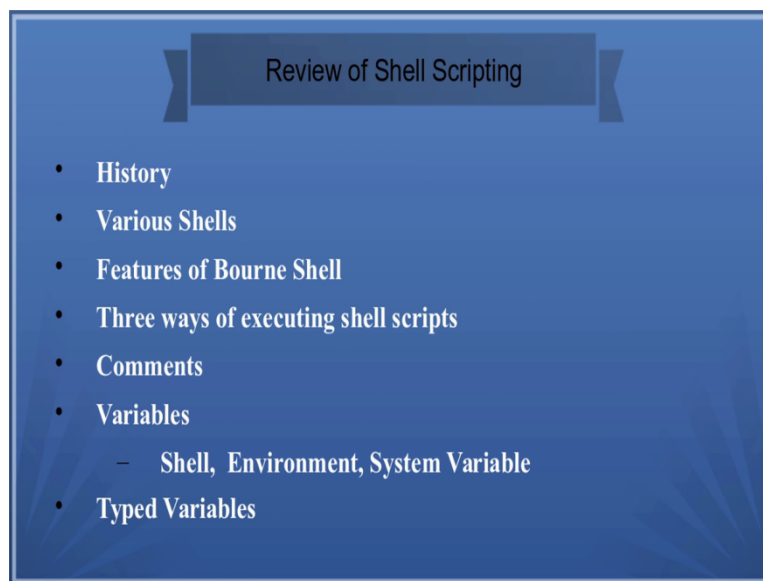


Information Security 3
Sri M J Shankar Raman,
Consultant Department of Computer Science and Engineering,
Indian Institute of Technology Madras
Module 44
Shell Review

Welcome to this session, final session on shell scripting. We will review what we had done in all these sessions. So this is essentially will be very useful for you when you are going to prepare for an exam and we are not going to look at any examples, but we'll just at a higher level we'll try to understand what are all the things that we studied in shell scripting.

There are lot more to be study ok, so our session you can look at the reference book that we supply along with the course or the references that we supply along with the course to improve your knowledge on shell scripting, but whatever we had done in the past 13 or 14 sessions we will try to review now, ok?

(Refer Slide Time: 01:00)



We started the course with history of What are all the different types of shells, history of shell, why we need a shell, a shell actually provides a nice user interface between the machine and the user and we also looked at various types of shells, we also mentioned during the course we are not going to stick with just Bourne shell, we'll also be taking normal shell, so it's Bourne again

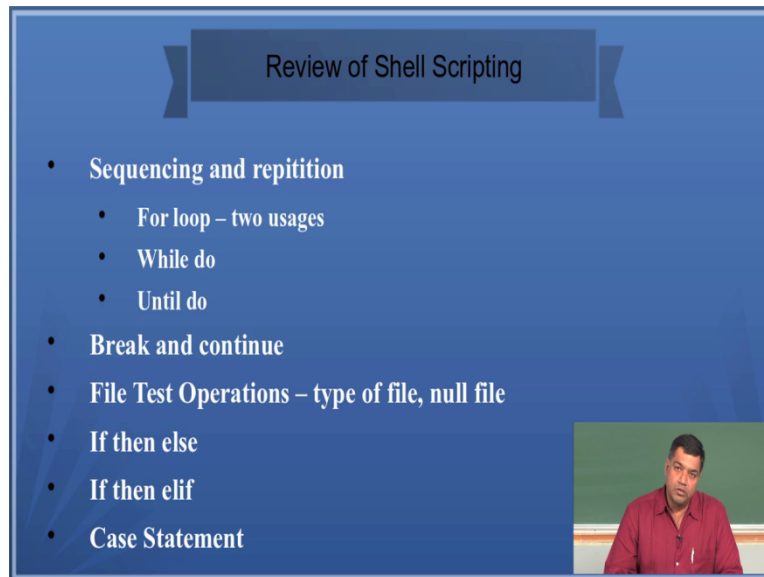
shell and then probably we said that we'll not be looking into corn shell and sea shell. The syntax for corn shell and sea shell is slightly different from what we had seen in the syntax today, but I hope in the last session we told you how to understand a shell script, without even knowing about how the shell scripts works, so it of course it needs lot of time to execute the shell and then debug it and understand, there are lot of other features also in shell ok, so this course was basically about the basics of shell scripting, we also found out that there were three ways of executing shell scripts ok, the one is using the ch mode, the other is making the invoking the shell itself inside the code or you can create an external shell and then invoke the program.

Then we also found out that your code should have lot of commands ok, which helps you to understand usually if you remember we write code for others, we don't write the code for ourselves because we know what we've done but we want others to understand what we have done, therefore that is one of the reasons why you actually command the shell script or the code, then we looked at variables, ok?

We looked at initially we started the course we said that only one type of variable can be declared called the string variable but then later we found out that we could also have typed variables which ensured that your program did a strict type checking so that then amount of errors that you'll get can be reduce for example if you declare a variable as INT it has to be a int throughout the end of the program, you can't convert it as a string and then make it as a int, so such kind of restrictions can be actually enforced by using the command called declare,

So we also saw that the variables were of three types, one is the system variable ok the environmental variable and the shell variables ok, so we had variables like path, the internal field separator, the log name and things like that. Now one of the things I mean, whether I should worry whether the system variable or argument variable or shell variable you'll get use to it as you start to do more and more programming and start looking at these variables.

(Refer Slide Time: 03:21)



The slide is titled "Review of Shell Scripting" and lists the following topics:

- Sequencing and repetition
 - For loop – two usages
 - While do
 - Until do
- Break and continue
- File Test Operations – type of file, null file
- If then else
- If then elif
- Case Statement

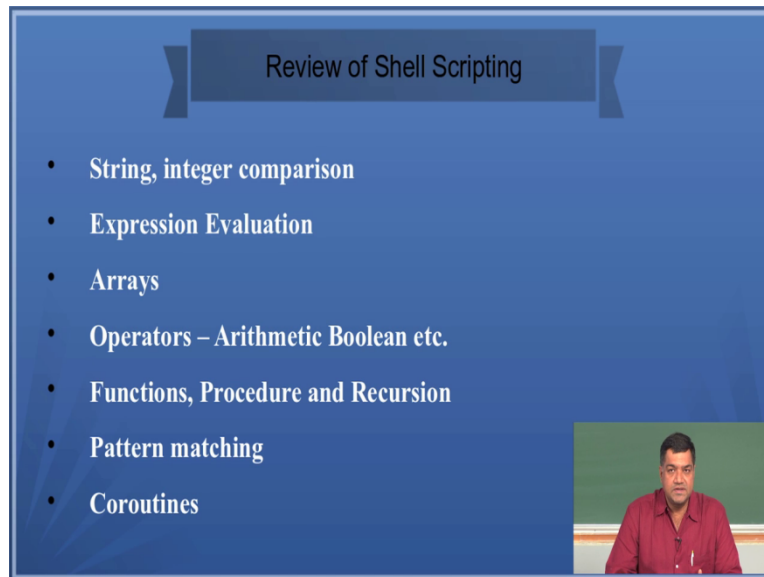
A small video inset in the bottom right corner shows a man in a red shirt speaking.

Then we looked at flow control ok, we looked at sequencing operations and how to repeat a particular commands more than one, we actually looked at the for two syntaxes of for loop, the one that worked on strings and the other that worked on integers, so and then we also looked at the while do and the until do loops ok, which one to choose, you could choose one that is that follows the logic of the code if you are commanding the code.

And then if you want to match your commands with the logic and choose one of this, we also saw that the loops can be broken ok, or you can continue executing the loops, then we saw a bunch of file test operators, ok you could very very I mean this is one of the powerful concepts in shell scripts ok, where you can test if the if condition test for various types of files whether you are using a special file, whether you are using a system file, whether you are using a system file such as pipes, whether you are using, whether your file size is zero ok, all those aspects can be tested and if you if you want to write a C program and system, C program is all extremely difficult.

So shell scripting actually helps you to bug with files in a much easier fashion, then we saw flow control where we used if then else statement and we also used if then elif statements etc. And if you had multiple choices or multiple paths based on a decision, we found that that that we can use the case statement which is similar to the case statement and C programming language.

(Refer Slide Time: 04:57)



The slide has a blue background. At the top center, there is a dark blue banner with the text "Review of Shell Scripting" in white. Below the banner is a bulleted list of topics. In the bottom right corner, there is a small rectangular video inset showing a man in a red shirt speaking.

- String, integer comparison
- Expression Evaluation
- Arrays
- Operators – Arithmetic Boolean etc.
- Functions, Procedure and Recursion
- Pattern matching
- Coroutines

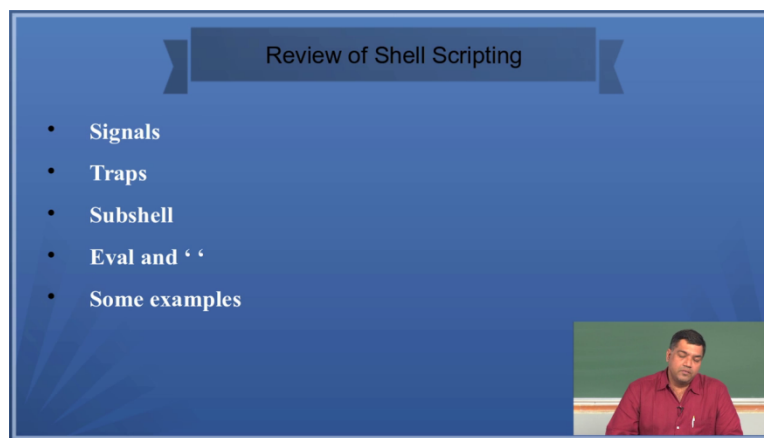
We also saw that the way you've to compare the strings is different from the way you've to compare the integers ok, we also found out that we can do expression evaluation ok you can look at kind of you know use the expression expr function or you can bunch of other utilities such as bc and other calculating, more powerful calculators you can incorporated within your shell scripts.

We also took a look at arrays ok so how to declare arrays and shell scripts, then manipulation of arrays ok, we also looked at operators ok, we looked at arithmetic operator, builtin operators and shell also provides you ways to write functions and procedures, we can also do recursive functions in shell and if you if you look at the functions, functions were very powerful because you can actually do a modular kind of development ok.

There are two or three ways in which you can do modular kind of development in shell script, one you can break a program into functions, the other way is you can write co routines or and you can execute not only co routines you can also execute program within sub shells, so so all these features put together provide you a wide range of tools to solve the particular problem, you can you can write parallel algorithms using shell scripts in that way your shell scripts are very very powerful.

The most and the power get enhanced further if you start using pattern matching, shell scripts provides bunch of pattern matching syntax ok, if you remember in the last examples that we saw in the last session, one of the examples was actually extremely cumbersome to understand but then with just two lines of code we are able to solve the huge problem, where there were about 10 lines of code that was written, solved these problems in just two or three lines of code, so that is writing a shell script is also extremely important when you are learning about shell scripting, so we also saw about co routines and then we also saw that you could do inter process communications with signals, ok?

(Refer Slide Time: 07:00)



And then we saw how to ignore signals we saw how to execute important parts of the code with and then trap, we also how important parts of the code can be executed by masking the signals and then again how you can unmask the signals, unmasking is a process by which the signal I mean your programming again starts responding the signals ok, otherwise your program and with signals you also understood that your program would either dump a core or it exits,

So those kind of abnormal properties of programs can be solved if you can use trap function or call then we also understood about sub shell, so how variables are passed between sub shell we looked at an example of Eval, where you have to execute the program twice ok, so you can execute the same program twice it was that that provided more functions and power to shell scripts and finally we also saw some examples whereby you could and try to understand if even if you are not familiar with syntax of the shell script, we were explaining how you could try to execute the code and then try to understand the logic behind the shell script.

Now is that all to shell script no there are lot of things that are there, we've not taken in this course because this is just an introductory course. You please take a look at the references ok where we have suggested some books, some of these books must be complicated ok, but then try out the best way is by practicing, you learn any language by using it, ok so please practice lot of shell scripts I think our assignments will help you and we wish you all the best.

Thank you very much.