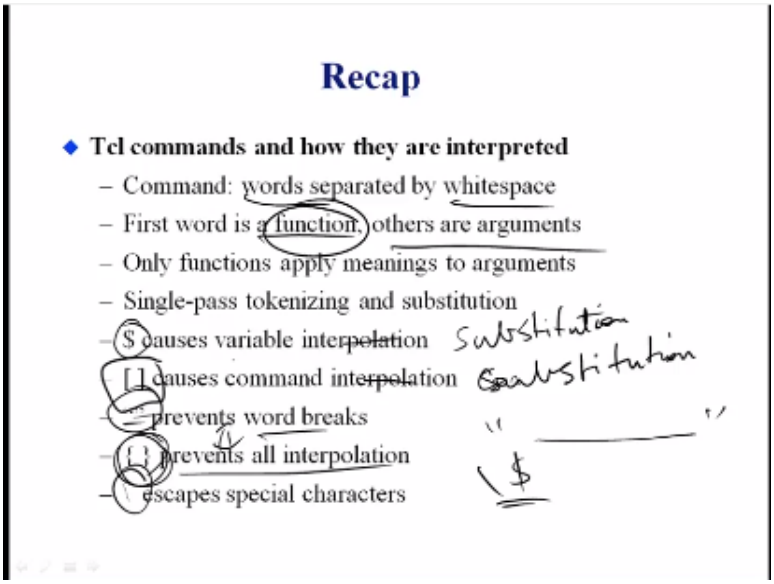
Programming using Tcl/ Tk
Seree Chinodom
Seree @ buu.ac.th
http:// lecture.compsci.buu.ac.th/TclTk

Hi everyone again welcome to this lecture today we will be continuing the discussion on Tcl/Tk that we started the last time so this is slitting 3rd lecture on the TK.
(Refer Slide Time: 00:29)



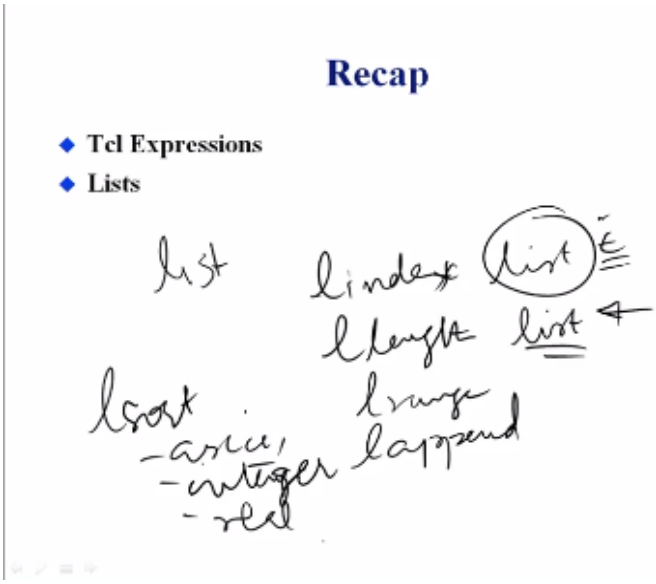
Let us see what we did last time so we started like I am in the1st lecture with the basic understanding of the TCL will works and how TCL will work invented and all the things then last week we started in earnest understanding TCL/TK programming language we started with the TCL commands and how they are interpreted as I said Tk command essentially like a mean for this for no command syntax or actually no grammar for TCL.
Essentially commands are nothing but words separated by white space and then the way that the interpreter works is form there is a built-in interpreter which divides the entire command into various words and then it sends those words to a parser the parser actually then finds out what is the real function and that can recursively call the interpreter for further elaboration but it just a very, very simple thing basically.

Again I want to emphasize this because in the very 1st lecture we saw that as we want, we want a very lightweight scripting language unlike C, C++ for much more heavy languages and essentially liked again once again I want to say TCL is used to for run applications.
And extending applications run multiple applications also let extending applications today to have been a very prevalent use of TCL now across various to that.

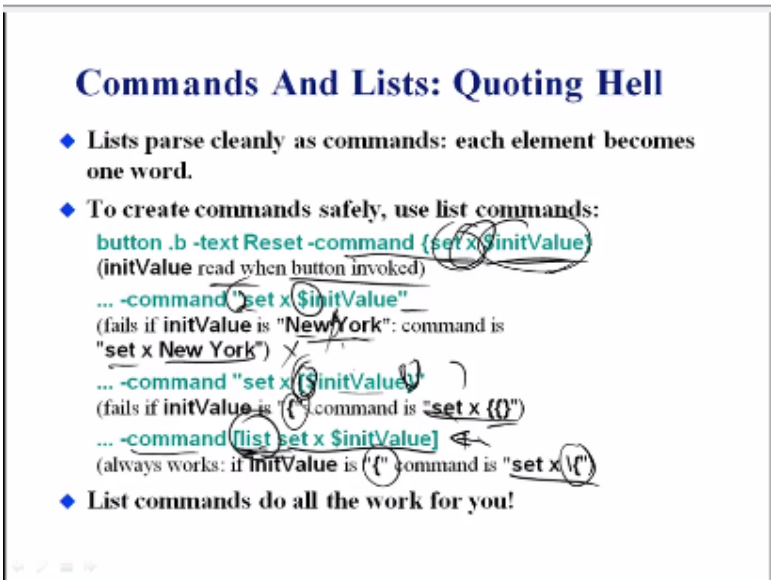
May be that you will use later on in your career there is not a single tool today that does not use TCL also wanted to emphasize that it TCL is a platform-independent scripting language so unlike any kind of shell or anything you peaked and then they are dependent on the particular platform that we use and what would be like small quirks associated with those platforms there is a shell associated with TCL that is the wish that we saw very 1st lecture.
I did not I forgot to mention about that as the last month so just remember that the wishes was developed specifically for TCL and again the TCL was developed at UC Berkeley by Oscar Haute who also wrote that book and if you look at the other book that I mentioned which is the Brent Welch Brent is a student also, also wrote and essentially is probably one of the first persons who actually wrote programs in TCL.
So though those 2 books are very good need to then paid you to look at those two books and so this so coming back to TCL command for the TCL command is a when it is words separated by white space the 1st word is a function and all the others are arguments only functions apply meanings to the argument as I said the TCL itself does not have any kind of grammar so the function really applies the meaning to the arguments.
It is always it is single passed tokenizing and substitution and organizing and then there are 2 main concepts that being reused from the last lecture one is what is called the variable interpolation or variable substitution is a substitution, variable substitution is essentially using the $ sign so whenever it sees the $ it substitutes with variable with the actual value and then you also have the second one which is a 2nd important processes in command substitution.
So saying in substitution, so the command substitution is use using these 2 factors whenever we see that anything enclosed in [] essentially like I mean that it is the 1st one again if interpreted as commanded that executed first and then the result is actually fed into omega of the expression then there are two ways of preventing the word breaks one is with the “” so whatever is enclosed within “” it is treated as one word.

And then we also have this {} essentially like that also prevent so interpolation so this also prevents the word breaks so the difference between these two is that the courts will still cause the interpolation of the variable or command essentially so the command substitution in the word substitution will possible within the “ “ whereas if you put it in the {} and it prevents all the interpolation of as well.
Then the last one is we saw escape character essentially the characters are for special characters so if you escape them is even as a $ then that will not cause any interpretation will be printed as

is again I want to emphasize that TCL is a scripting language usually takes strings as the input and strings as output so it does not attach any kind of type to the datas the way that type is provided is through those functions with this we will talk about more and more.
(Refer Slide Time: 06:41)



So after this we also looked at the TCL expressions as to what are the rules for the expressions one rule that comes to my mind is this floating point versus fixed point so whenever we write the fixed point expressions the TCL also let consider the result as a fixed point whereas the moment we start a floating point does not matter after that how many TCL points used the TCL expression will return a floating point number.
So this ,this is something that you need to use it with caution so I will , I will advise you and then I leave that and then we also started looking at TCL lists essentially, so lists are essentially like a mean the collection of objects essentially and then we assign ,you can actually group those, the elements in a list and separated by a white space essentially like the list is simply a string with list elements separated by whitespace.
We can use {} or codes to group the words with white space into a single list element which is very similar to what we saw here essentially so we can use the {} to group elements and treat them as a single element inside of the scenes and then there are many ,many commands associated with this a list command essentially list will create a list and then there are several of the other commands.
That we will see in this section into a section that will actually work with the list and manipulate and cause of the lists.
(Refer Slide Time: 09:05)



So some of the commands we also saw in the last ones are L index the L index essentially like a mean with an argument list and a number I will return the I element of from the list and the l length list , return the number of elements inside that list we also have L range I think you can guess what it is and L append there are L insert which inserts exactly where you want to insert l append will actually insert an end.

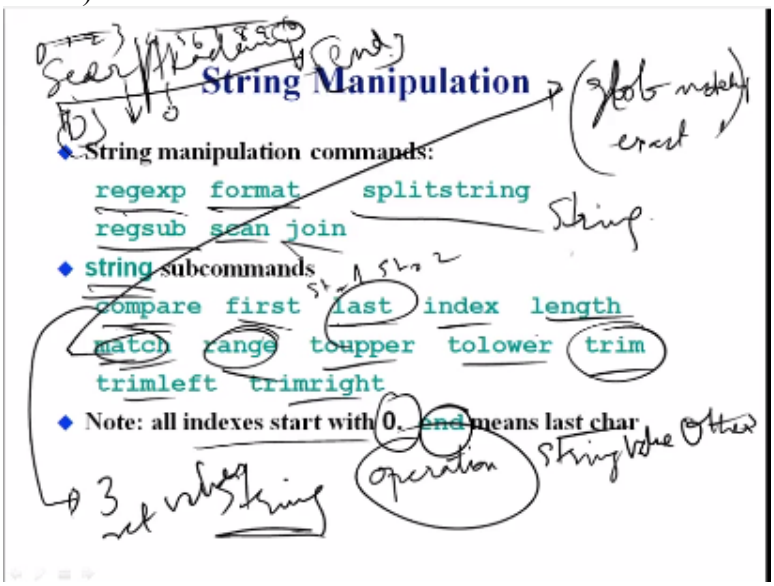
And then L sort is an another way to actually sort a list and this can take multiple options essentially we can give like very switches for hash key or integer, or real and then you can specify whether you want an increasing sort and decreasing sort things like that and then there are also like to contact need a list essentially like we have concatenate multiple lists we have CONCAT command we also have a joint command for joining a new string into a list.

And then split string the other one which is essentially split command and this basically to split on a particular string the list into two to six the separate lists so, I think I mean these are the things that we talked about we will we will also like go through some, some examples later on but I also wanted to actually talk about some of these things on is the command send list essentially so lists parse cleanly as commands each element becomes one word.

But in order to create commands safely we use the list command itself that is the one that we saw here just list the reason for that is here there is a one command and the command itself is set X in its value $ init value.

So this is essentially like a mean what is this mean to init value is red when the button is involved now this command is fine we can also do this init value in codes so that it actually does command substitution but here the issue is if this init value is the 2 different words separated by

a blank space in between then it fails because now this variable that it expects is actually in this essentially our set X.
New York are is not possible this command fails because this is the X that it is setting this value to and in init value is New York which it cannot set it so essentially like the command will say here whereas if you say okay now set X and in ports we gave the init value or so in {} can give the init value now what happens is if you have unit value as one of the {} then it fails because again like I mean you know this is this is not a valid command.
But there is no escape character nothing for the curly braces so how do we specify the same thing there we want to commander set X to the init value which is space so that it covers any condition possible so what we do is essentially like now we say command here we specify the list and then we say the XS init value instead of the ports or inside that the {} so here it will work because if the init value is {}.
Then it basically automatically escapes that {} and essentially gives you as that X has escaped {} which will work so essentially using the list command is a very safe way to actually specify lists first specify this so we will see this more and more again the one thing to notice TCL it is turn to lot of errors and I mean you can make a lot of mistakes and we can make lot of errors very easily and then quickly.
You can actually get bogged down into your own errors and trying to debug errors so some of the safe practices when we say show to you I think you should actually heat and pretty much try to adopt these kind of paper practices which will enable you to succeed in coding in TCL.
(Refer Slide Time: 15:30)

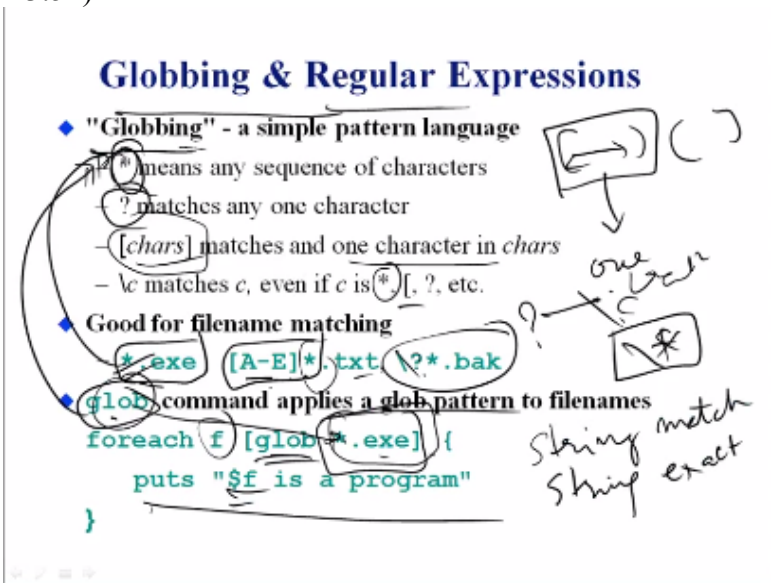


So now we come to the strings essentially so we know that , the string commands essentially our name there are many ,many comments or string manipulation , strings again ,again the strings are

group of characters that we talked about previous lectures and then string manipulation is one of the main things basically and string is the basic data item in TCL and as I also mentioned that the commands themselves or essentially like I mean the whole commands are parsed.
As strings which is essentially character separated by work space and essentially like I mean using that we can do like lot ,lot of things and then the key thing to also understand is for using strings we can do pattern matching with talk about and pattern matching is again one of the key elements in ,in the TCL and it is slightly different than what we saw in a Perl for the matter.
And the 2 main concepts are one is exact matching and globe matching this is GLOB globing so glob matching and exact match , will see like man what is this glob style and what that means , so the string manipulation command so regexp so this is for the regular expression is also a format this is the split string reg sub scan and join these are all the string manipulation commands there is the basic command.

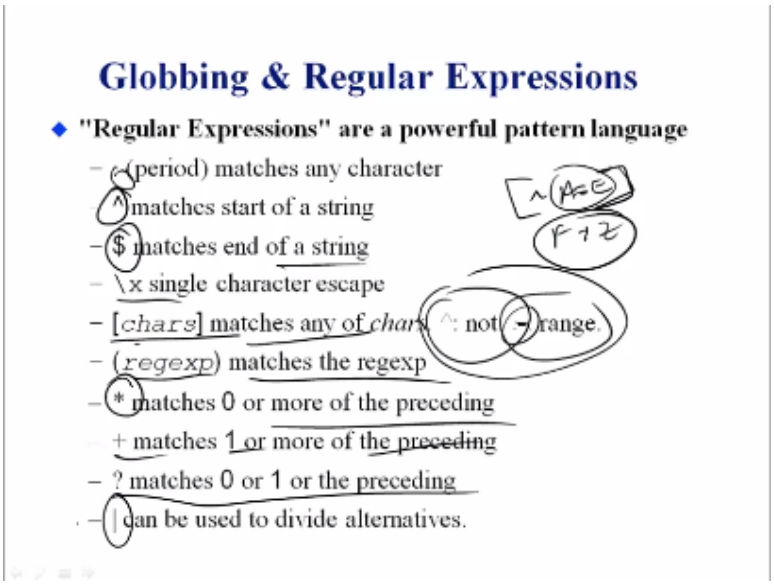
Which is also a string and the way to use is string followed by operation and then, string value and then any other arguments , so string command similar to the list commander talked about is the main command essentially like that is listed here this is the string command and then the string has the sub command which is this operation that can be compared first last index length match range to upper to lower trim ,trim left and trim right.
So we will see like some of the examples of how to use this commands one quick note is all the indexes within the string start with 0 so if I say hear Academy so this is string values 0, 0 and I is also you can say end okay so it starts with 0 and then end means the last character.
In this example and notice that actually one word so if I split here this is a string and this is another string and for this string this is 0 to n or this string this is 0 and this is the end.
So it actually goes like 0123 if you make it continuous then it is actually 45678910 so the value index10 if also same of index n okay , so now let us look at these comments the string compare takes two more arguments which is string 1 and string 2 and that returns 0 if they are =-1 if string 1 starts before string to R it returns 1 so just this compare as 3 return values , and in the first also takes to 2 arguments string 1 and the string 2 and then that return the index in string two of the first occurrence of string 1.
If the string 1 does not exist as a path in string2 then it returns negative 1 to indicate that this string 1 is more , and then the same is last essential which is the last occurrence of particular string 1 some of the other ones that are interesting are range ,range use basically it also takes 2 additional actually like 3 additional arguments one of the string and then 2 numbers and then it returns the characters between those 2 numbers and arrange.

And then the string match actually that takes a pattern and also after that it is a string so takes 2 more arguments a pattern and the string and then it returns 1 if the string matches the pattern otherwise it return 0 and here the string match command this one uses the globe style matching that means that you can use special characters and while cards we will actually see that further example then we are we can also like trim the string essentially.
Which is the it trims from both ends of the string and then the character usually ligament also has an optional characters that you can take and that is the characters it will print and optionally I mean for as a default the character is always white space Which is it trims the white space is from both ,both ends if there are many and then the trim left and trim right does the same thing on the either side the 2 upper and 2 lower is essentially to produce the string or the lower case.
(Refer Slide Time: 23:52)



So now let us look at the key thing that I mentioned in the previous one which is the Globbing so the globbing and regular expression essentially the globbing is a simple pattern language so I want to emphasize these four things which you want to use in your Globbing style which essentially like I mean with which you can really use those two those 4 steps first 1 is the * that means any sequence of characters.
So if you want to match say fear a * that matches here academy any kind of spelling or just the array that is fine so anything that the * will specify that, then we have the? Person mark matches one character at a time.
So how many? You give it not just that much campus for example if you say yes? our then it would not match here Academy before so you have 2 characters in between and it only matches one of them so that match SIR Academy SAR academy SUR academy whichever way that you

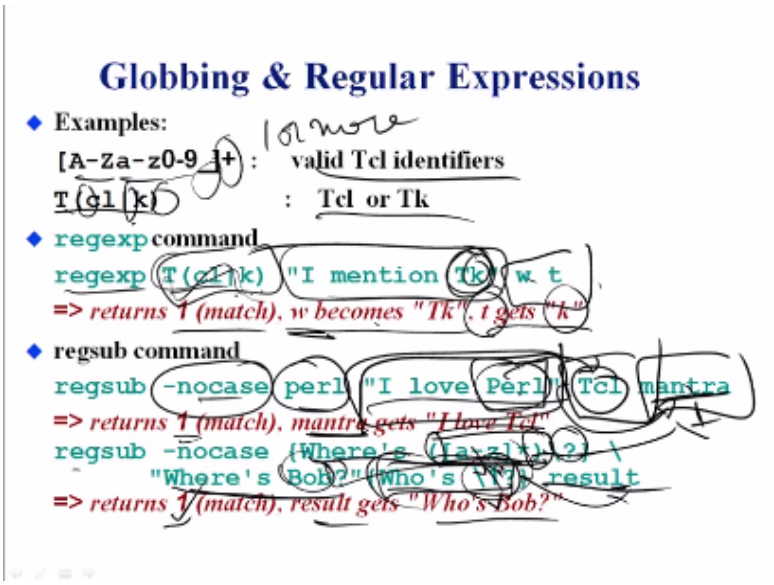
can spell it will match, now we can also have [ ] and some characters in the middle this can be characters or a range of characters.
And here it matches those specific characters and it is matches one ,one character in that one or in that range again you can have like the multiple of be essentially to match more than one character but every [ ] will match one character of that set the set that is the design here , and then we can also like \ with the C is Match to see even if it is all these special characters so if you want to exactly match and * you basically escape it as so that matches exactly that counter.
So I wanted to make a distinction between the globe style pattern matching and the globe command that we saw in the earlier box before that here are some of them so start out exe that is specifically this Globe style which is this one and then we can also have a through e *.
That is first character it matches ABCDE after that it can have any number of characters and with the extension text.
Here we can escape? Which is basically though it needs to have a? Then any number. back is what is matched here so it will be a good exercise if you can actually identify these kind of things so later on in lexical program now the globe command is also like another common command please do not confuse it with the globing style matching the globe command applies a globe pattern to filename essentially the globe actually returns the Unix file names.
In that particular directory so here for each F globe * exe but $ F is a program basically it goes to your current directory and look for any of your piece and then it prints out this message for those programs so the globe command is return the unique file essentially whereas globing cell matching is used to match the particular characters now if you specify like I mean so what is the alternative so this is this is by using string match now if you say string exact then this needs to have one match be * or in all these as per had Electra.
(Refer Slide Time: 28:53)



So now a couple of more things about regular expressions one is this period that also matches any character the chars matches the start of the string the $ matches at the end of the string and then the \ X is a single character escape and then , the [] essentially not in any set of characters tile the , if not and essentially you can use - to indicate the range so these are some of the things that you can use within [] so if you do [] till the a E then , would not match the set 8 through e it is range for the 1st character.

You can have anything like from A through Z or any numbers or anything of that okay , in the parentheses and the regular expression matches the regular expression , and here the * matches 0 or more preceding the plus matches 1 or more receiving and then the ? Match is 0 or 1 of the preceding and then we can use this / to divide alternatives.

(Refer Slide Time: 30:58)



So here are some examples so for the regular expressions, so here A to Z lowercase a to Z 0 to 9 and then - and then when it is + it is one or more this is a valid TCL identifiers so any kind of variable names are using this then here we say like TCL of TK with parentheses and then using the bar TPL of a so that is TCL our , so here another in the in this red X command it is a red x tcl/tk I mention PK w T so this will return one.

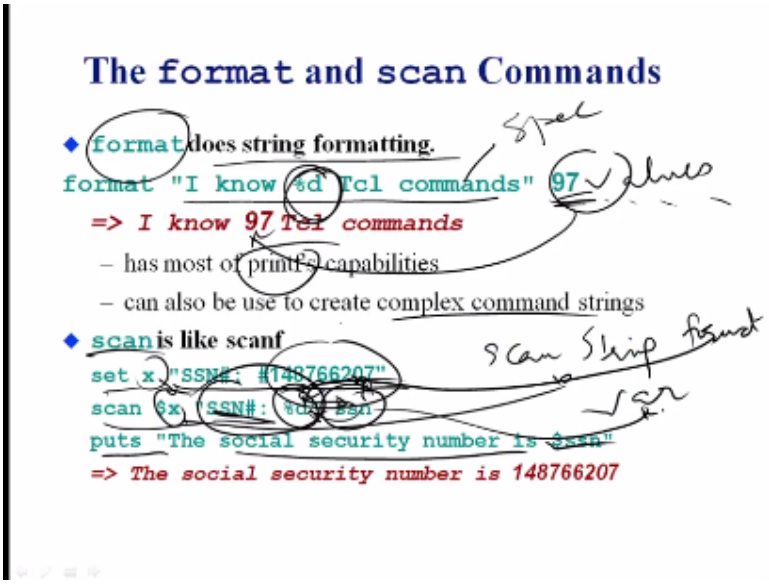
Which is a match and then the W becomes TK and then the T gets K essentially , so if you recall the reg x command actually it is it gives basically the reg exp leg X followed by flags which is optional and the pattern here and then the string , and then the last 2 are the sub 1 and sub 2 which is essentially here what are the sub strings from this match that gets added to it , okay so now let us look at the Reg sub-command.

So that is so essentially like I mean so here it matches the TK here so the W becomes the TK which is what is matched here and then the T gets the substring K which is where the matches happened , now let us look at the Reg sub-command accept command again has switches here in this case it is the no case is the switch and then the pattern which is essentially like here it is Perl then followed by string and then we also have a sub spec essentially.

For this and a variable name so here the no case Perl means match any case of Perl so it returns one a match because it matches the Perl and then the mantra is the variable name that gets this I love TCL as the string because once it matches this Perl it replaces that Perl with TCL and then assigns the whole string to is the new variable mantra, okay so now here is one example one quick question essentially like a more case essentially.

There is into the * escape? and then you have various bob and then we say basically like the sub spec is whose and then escape one and then put it in result so how do we interpret this is

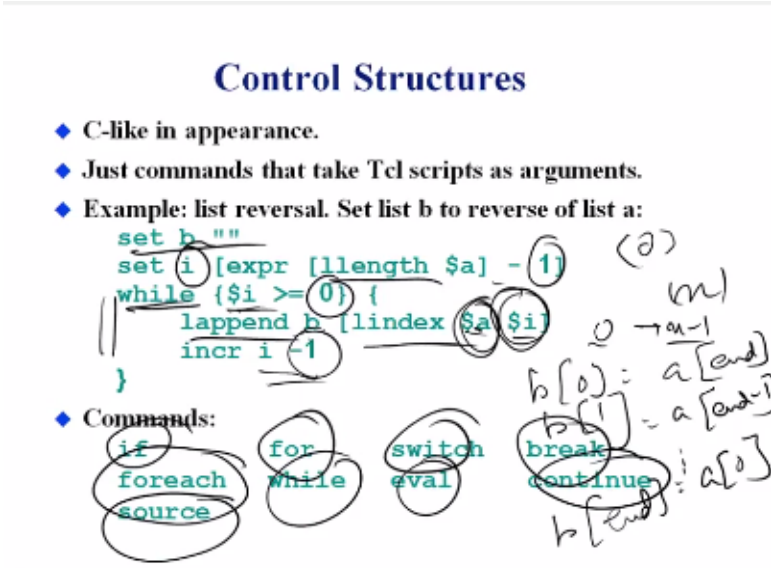
essentially like I mean so this is going to match varies some name any name followed by ? so here the string is there is Bob so matches them so it returns 1 perfect match now look at the Parenthesis here so whatever is within the parentheses will get replaced because this is termed as escape 1.
So now basically the sub specs call for who is and then we places with this here and then that string is assigned to result so the result gets whose bob , the Bob is the one that got matched.
And then this is the parentheses that is called access key 1 then that goes here , so I may give you like some more exercises to work on these concepts so I think these are very important ones or demo understanding of glob star matching and regular expressions.
(Refer Slide Time: 38:19)



Now the format command essentially this is something that also let me talked about in the this one this does string formatting so in how do we format a string essentially giving you the, the particular value and then essentially like I mean use we use it in substitution to that, that particular and so again here we say basically it is format and then this is what is known as the spec and then these are all the way these values.
So this is when 1 and you can have multiple values so every occurrence of this various characters get replaced with those values so here we say like % D that means that it is a signed integer and so 97 incident it is replaced here you can have unsigned integers we have , O for unsigned octaves x for unsigned hexadecimal C for mapping and integer to the ASCII character it represents s is for a string F is for floating point and then E is using scientific notation.
And then the G is in either , the positive H which is floating point or the you know scientific notation which is shorter use that ,so essentially like I have been pretty much involved with all

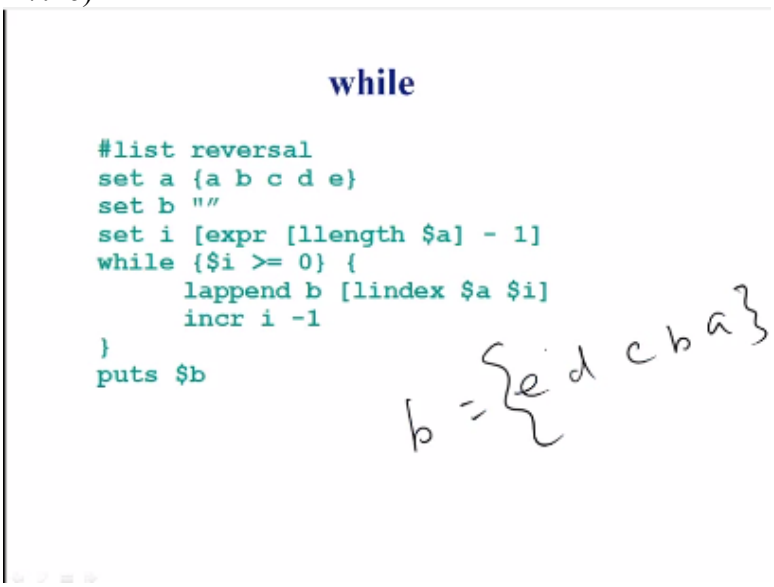
matters as most of the print f capabilities we can also use it to create complex commands strings now the next one is scan ,scan is like scan F and now C programming , so the scan f actually parses this string according to a format and assigns the values to the variables.
So here let us assume that the string X as this information which is SSN and then we say basically like some number so when we say scan we scan this particular variable $ X and then we say a format which is SSN # is %D and then that is assigned to assessing now if you say like the social security number is $ SSN this SSN basically it gets replaced with this number here from here and then we can actually display that.

So again for scan comment this syntax is scan string this famous the $x here and then the format and then followed by variable and here the format is %D on this whole thing is a sin %D and then the variable is efficient so that is how this assignment works.
(Refer Slide Time: 42:43)



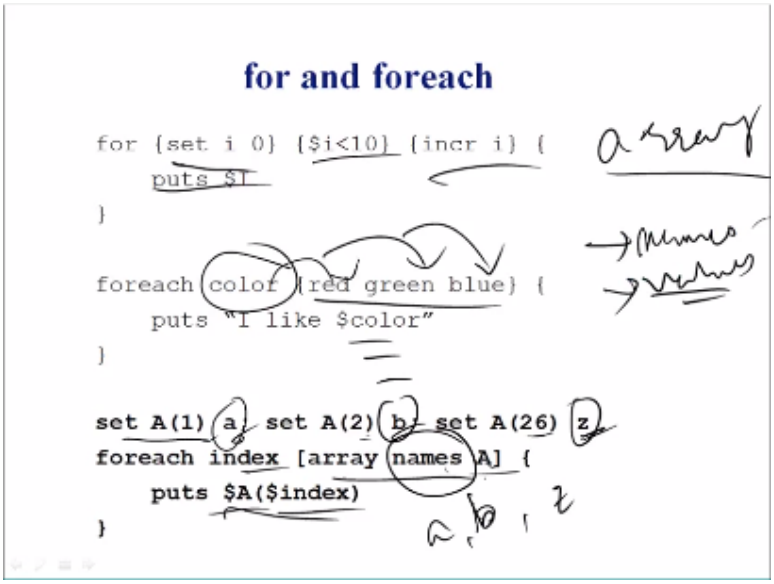
So now we come to another interesting topic which is the control structures, control structures are needed for controlling the various operations this basically gives the program the various structure or essentially the essence of a problem with all four structure we covered this for Perl we talked about it is data structures and then we went into more control structures even this is very similar.
The control structures are like see in TCL so if you know see it is ready to write these control structures in particular you will understand it much better, they are just commands the takes TCL scripts as arguments again you should think again when we come to the TCL there is no grammar

so everything is despicable then the command will take other TCL scripts as arguments for it for example there is a list reversal.
Where we set the list B to reverse of list A so how do we do this so here we declare a variable called B and then we say we set the I to the L length of A-1 can anyone tell me like I mean what, what is this do, so l length actually gives the length of the list essentially And then we need to subtract one from it first, first element is 0 so if you have n numbers the actual list indexes of 0 through n - 1 so there are n elements exactly each one is in X 0 3 - 1 so we subtract that one from the thing , and then now we just go through this while loop which is one of the control structures and then we say $ I >>= 0 as long as the is I is <0 do this which is L append b2l index a $ I.
So again L index a $ I returns the character or the, the list element which is of I from the list $ a so now B becomes or b0 in terms A and , and then the increment I +1 oh sorry so increment I with -1 which is actually dependent so here now the next element will be 1 because we do L append it becomes a n- 1 and so on so now you can see that actually like I mean it would be consider this and finally this is be and will be a0 and that is when the program terminates.
So it is a good program to do the reversal of the list but you can use this so you can see the next video is a couple subject will be the while so for doing the for controlling the commands controlling the structures or the control structure commands are essentially if for which break for each while eval continue and source the source command is very commonly used I cannot believe this just source a set of procedures or a new commands.
(Refer Slide Time: 47:26)

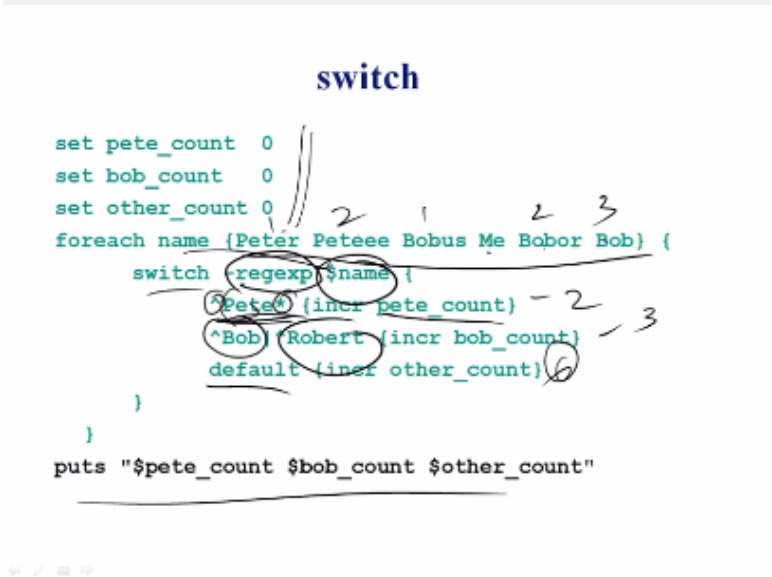


So let us look at if-else so here we say the variable X we set it to two and say that if it is <3 then put X is < 3 if X is otherwise in this is X < 3 X is 3 or more so when you execute this program

for x value of 2 now it will say this and it will exit from executables at all because it is already exist this entry , and then this is the while example before this reversal essentially here the b is e d c b a after you put in this level this is the same thing that we saw.
(Refer Slide Time: 48:26)



Now we have far and for each so far usually we say set I 0 then $ I <10 agreement I and then put $ whatever essentially we can take $I itself and then for each actually uses a list as it is main arguments and it is the users another variable in the middle. And this variable essentially or just taken through each of the values inside the list so for each color red green blue which I like color so it basically puts I like bread I like green I like blue same three things.
Now there is another one which is the set a1 as a2 SB 8 3 a 26 s Z assume that everything in between is also set somehow now for each index array names a puts a and then the index, so there are two things basically like I mean now in an array so these are what is called array commands essentially as I also like I mean we will go through this that kind of commands but in array as an data structure the command is array.
And then you have names and values so the names actually return the inbox only indexes and then the values return what it is set to for example these so in this case essentially like a index is a indexed array names a and then put a $ index now we can see that this program will output a B and Z.
(Refer Slide Time: 50:51)

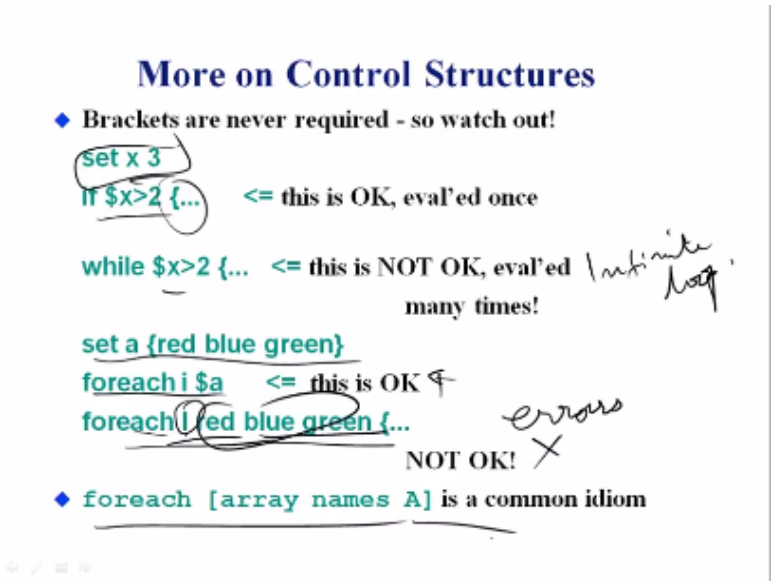


So and then finally the command called switch essentially this is very similar to the case statement that you may be familiar with so here we see another example , so we put this counts as 0 and then for each name we have several names here a list of names and then we say switch and based on the Reg X name so it is basically this name and then if it matches with Pete at the first character essentially this is the first one and with the number.

And then this is the main thing and every increment been count , and then if it meets Bob or Robert then we will increment the Robert and then default is increment other count so now can you tell me like I mean what this program will generate as the counts.

So for the Pete count it counts 1 2 so this will be 2 for the Bob Robert essentially it matches here 12 and 3 and then finally like me the default is incoming other count the other count is actually 1 2 3 4 5 6 it goes to the catch-all anyway so now you have 6 2 and, okay.

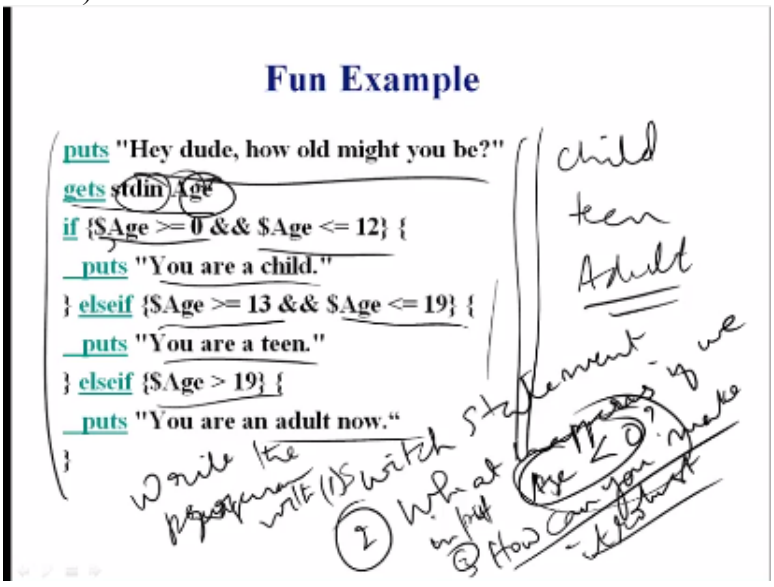
(Refer Slide Time: 53:03)



So let us look at some more rules on the control structures so here we have X, is 3 and then we say if X is > 2 and this is okay because it is evaluating what we wants but if you say while x> 2 and then you are setting X 3 this is just an infinite loop, now here we have a set a as reg blue green, now say like we say for each I $ a this is an OK sentence but if you say like I mean same thing with the red blue green then this is Not OK .

This is essentially it basically just a result because it just gets confused as we know that or each has it is 2 arguments 1 and 2 so once it like more number of word and then , for each array names a this is a common idiom in TCl.

(Refer Slide Time: 54:52)



So now let us look at one small example which is just a fun example for all that I mean so this is the control structure that we all familiar with and then very easy to understand too so we look at

this one and then we look at some complicated examples in the next lecture so here first of all what is this what is this program do this program identifies person as either a child a teen or adult or does it do so it takes for a question how old are you? And then it gets that the value.
So it takes it from the standard in and assigns to this variable called age then it just examines the variable so if age is >=0 and if it is <= 12 then it prints out as you a child if age is>=13 and age< 19 then it resolved that age in your team and then finally if age is > 19 then in case that you are an adult so a simple problem so how can you write with say switch statement you for once which statement, and number 1 switch statement and number two is some what happens , if a input aged<0 and then how can you , maybe Prove this so let us discuss this in the next lecture so I want you to go and work on this one essentially.
First of all modify this program into using switches this switch decision which is essentially which we want above it here , okay and then the second thing is what happens in this program if we put age as <0 what is the program on put try to find out and then the last question is how can you make it robust meaning how can you power this condition if this errors out or does not error out you cannot cover this condition and that also you can use the switch command essentially so let us pick it up from this point in the next lecture thank you all you go you have wonderful day thank you bye.