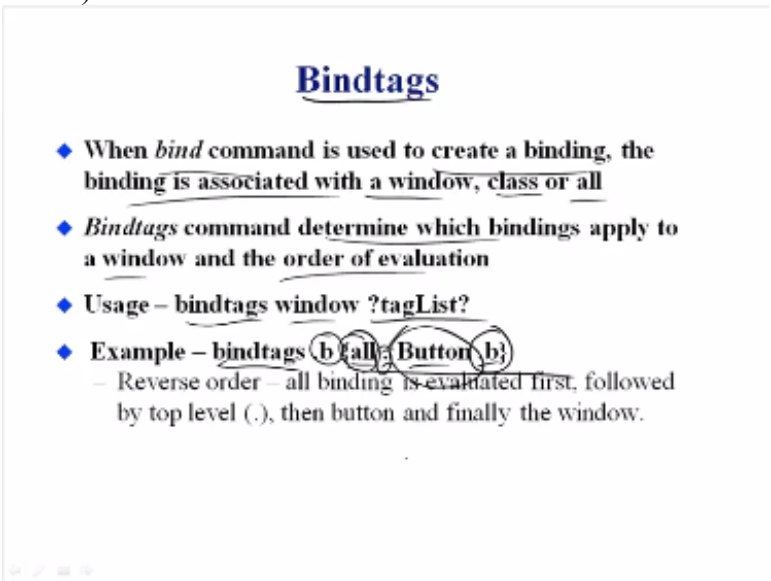
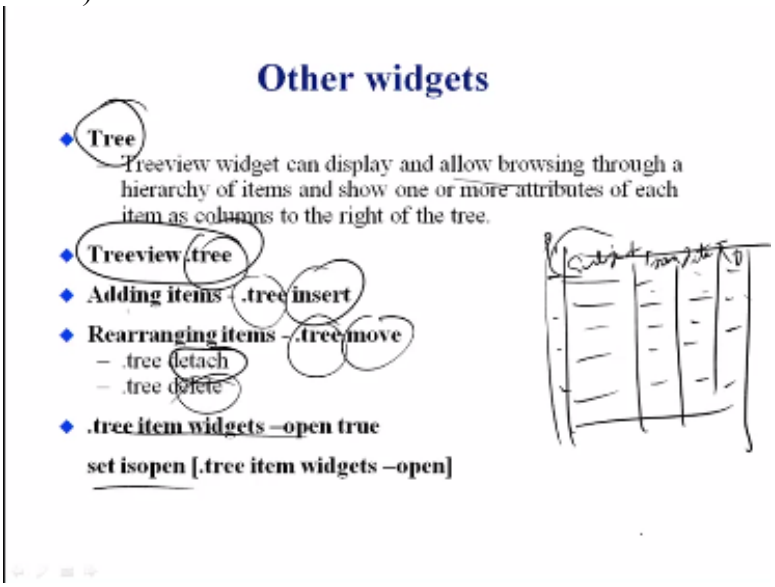
Programming using Tcl/ Tk
Week 5
Seree Chinodom
Seree @ buu.ac.th
http:// lecture.compsci.buu.ac.th/TclTk

Hi everyone again welcome to this another lecture of the LPS class today we are going to talk about we will continue to talk about TCL that we left last time.
(Refer Slide Time: 00:18)



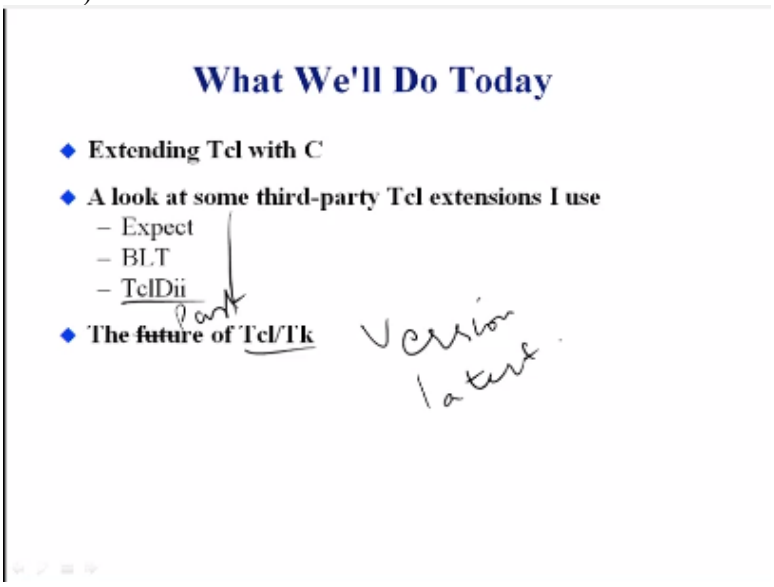
Last class or the last lecture we saw from the canvas as a widget and we got lot of commands as to how to play with the canvas and how to actually get people to use the canvas we also found some of these new commands like the first of all like the bind command that we started talking about it like couple of weeks ago that bind commands and then we also associated with bind tags because there are more than one binding or an event.
So essentially like I mean when there is an event whether it is a button press or a move of a mouse or whatever it is particular graphical user interface we typically use the bind command to create a binding for an action so when you actually select or put together a command using the button press they can write all the characters and then unit press run that command gets run in the background.
That is because of this binding that causes it to run there the run command is bound to that event what that event is actually bound to perform so that is something that we have seen, now we also talked about this bind tags essentially the bind tags essentially like I mean determine it is a command that determines which recommend that determine which bindings apply to a window and the order of evaluation.

So a bind command itself is created creating the binding and then the binding is associated usually with a window a class or all so you can specify one of them basically so we saw this in the previous lectures the bind tags is command actually now determine which bindings applied to the window and order of the evaluation what does that mean so the usage is essentially is bind tags window ordered by the tag list.
So an example will be bind tags. D is our window and then the order that we specify is all. Button and then. B so when an event happens basically like this it is applied to be all which is like everywhere outside and + the button and then to the top-level then the button itself whatever is that is going on that will be evaluated and then finally like .B so this will allow you to actually like this multiple events can be trapped and then.
We I mean I am sorry multiple bindings can be trapped and then they can be run and then they do not need to be running at the same time or all simultaneously which they follow this order by which it will be run so this is something that I wanted to tell you like I mean last time we saw this bind tags we did not explain at that point whole lot now it is there.
(Refer Slide Time:03:51)

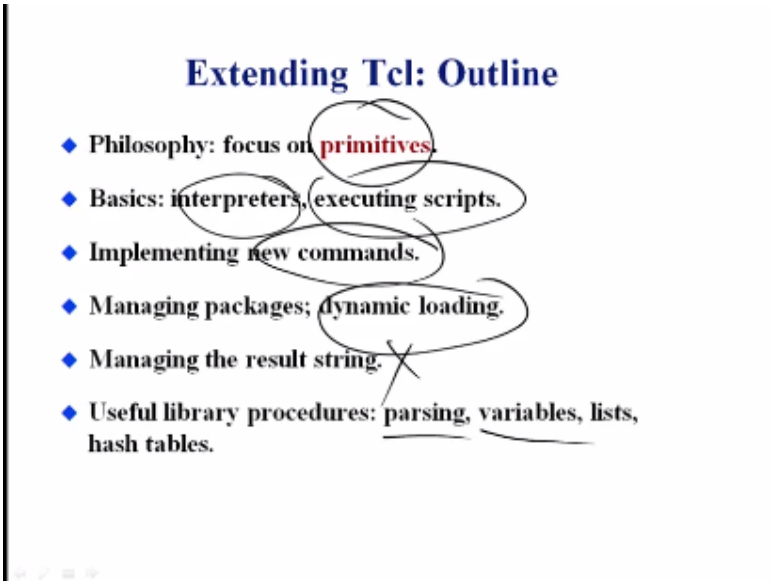


And then the other widgets that we have not really touched upon but I really want you to understand and more try to read upon this is the tree widget this is also like very common when you look at your Outlook mail or any kind of form lists basically like I mean the views that are displayed are based on the tree so the preview widget can display and allow browsing through a hierarchy of items.
And show one or more attributes on each item as columns to the right of the tree so you have like big thing basically have each item which is displayed and on the right hand side there are

columns there it is displaying the its attributes, so it is a mail program here you can subject from, date two things like that the various fields that you think of or a particular mail I mean you can also examine important priority whatever.
So for this mail you can associate on set this also and then this particular widget is created by the tree view command so here we create tree view. Tree is the command and if you want to add anything to the tree we just say like the tree insert and then rearranging items is. Tree move and then we can detach an item from the tree using both totally attach or even completely delete that item using. Tree deletes then we can open widgets basically.
So that is like set is open command first we need to specify whether it is an open is true and then set is open to open the widget and then by opening the widget we can now add stuff into it the list of query serve things like so this is another widget that we never talked about in the previous lecture but I just want you to know we useful one.
(Refer Slide Time: 06:12)

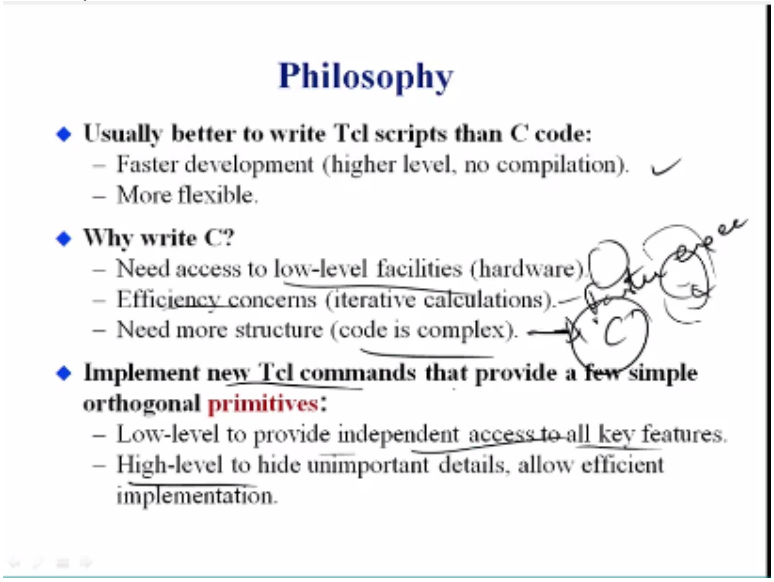


So in today is lecture we will be talking about extending TCL with C and then some of the third-party TCL extensions so expect BLT and TCL that and then finally the future of TCL/TK this is more like the past the past kind of determines the future past of TCL/TK what are the various versions and what they did what kind of thing and then what is the latest version , so that is what we will cover today and , let us look at the topics.
(Refer Slide Time: 07:27)



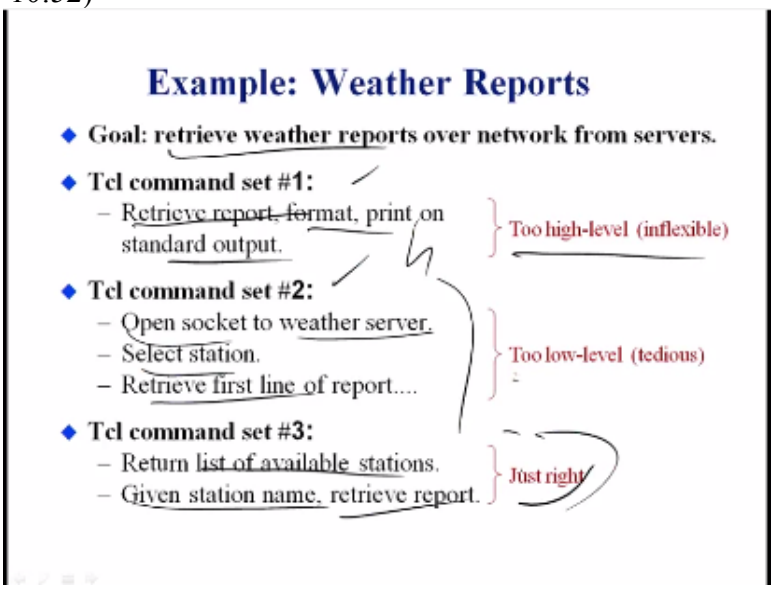
So extending TCL so the main philosophy that we will be talking about this to focus on the primitives and then we will have like some basics on the interpreters and how to execute scripts how to implement new commands dynamic loading managing the result string I am NOT going to talk about it today the main thing is essentially like I mean this is regarding machine level understanding as to what are the different memory pieces then there the result goes into and how do you manage the whole results And then we will also like look at some more useful it is like parsing variables lists and hash tables.

(Refer Slide Time: 08:17)



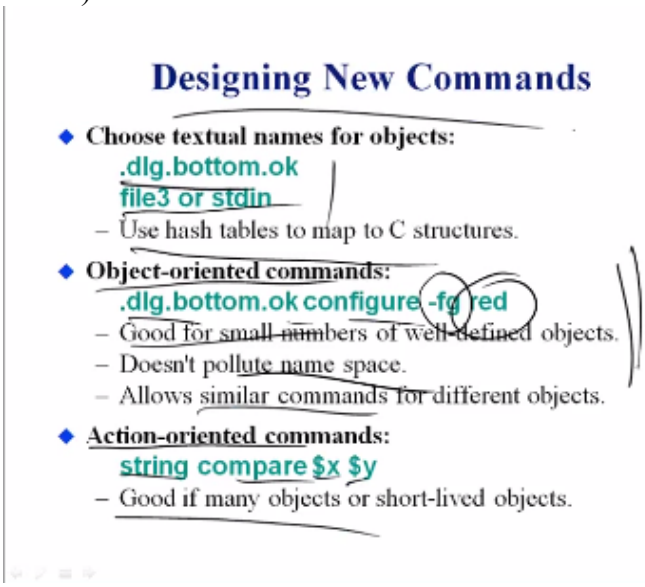
So we saw this in the very early lectures of TCL one is usually we can write usually better to write a TCL script rather than it is free code it is quick development and more flexible and it is high level also no compilation it is also like a higher levels as to write the codes and why do we

write it in C this is something that we also they talked about earlier you need to access the low-Level facilities that is like hardware resources can be managed like in fact the malack or ml of actions and memory allocation in C that specifically goes to the heat.
And actually allocates memory we can request all those kind of different hardware features directly using C then there is also efficiency concerns basically like when we have like iterative calculations Steve and since it is a compiled language in optimize and hence the it provides much faster execution ,and then finally if we need the more structure and then if the code is complex say for example we are generating a program.
For DNA sequencing momentum or DNA sequencing that entire algorithm can be coded as a C program than TCL script again this is the hypothetical and probably ideal case you ,you may argue even TCL port which is okay but when this port gets complex we need to have like definite infrastructure to be in place and for that C is literal ,the if you have to write a new TCL command like I mean they are implementing.
New TCL commands that provide few simple orthogonal comments so ,it is low level to provide independent active all the key features but at the same time it can be high level to hide unimportant details and allow efficient implementation so that is the reason why we can implement new TCL commands.
(Refer Slide Time: 10:52)

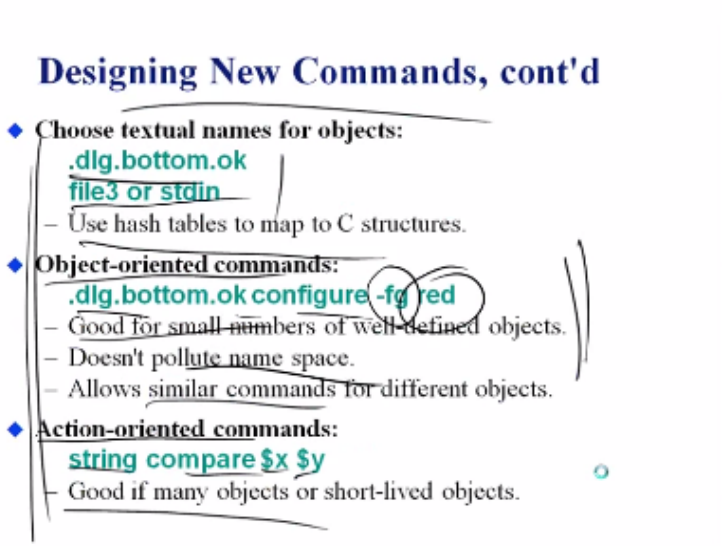


So here quickly like a weather reports so the goal is to retrieve weather reports over network from different servers so if you use the TCL command set one where we say like retrieve report on that print on standard output that is too high level so it is inflexible now we let us look at a

second set of commands we open a socket to the weather server select a station the day of the first line of the report this is too low level.
So you can see that actually like we still use the TCL command set here and here , now consider the third command set which is basically returns the list of all the available stations and given a station name literally put that that is your command structure this is just right so we will look at how to do it.
(Refer Slide Time: 11:59)



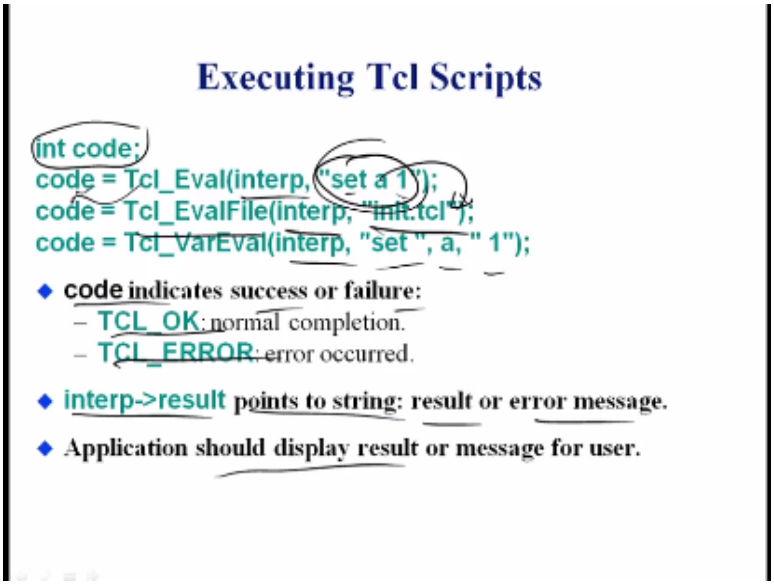
With these kinds of so, commands itself so in this section I means for designing the new command we want to make sure that we need tools to the textual names for the objects for example. Dialogue bottom thought ok file three your STDIN so really like objects can be aimed in the textual context if this is the case then we have to use like hash tables to map to the see structures other one is essentially like.
We can occupied you of the determine the commends for example we can say like dialogue bottom up a figure for ground red, this kind of style is good for small set of small number for well-defined objects they do not pollute the namespace basically each one will be independent and also like allows similar commands for the projects so that is there are definite advantage we can go in with object-oriented commands.
And then the third category is actually the action oriented command which is string compare $ X $Y this one is good if many objects or short your objects so again when you design a new command make sure that you considerations go to these.
(Refer Slide Time: 13:33)



Now how do you format the, the new commands so usually the preferred they is make them easy to parse with TCL script so somebody else write your TCL script they can easily pass your command results so for example here temp 53 high 68 low 37 precip it .2 Sky part so this kind of thing basically now you can move that book like this is the index and this is the value so the current temperature is 53 highest is 68 low is 37 and amount of precipitation is .02 and then sky is partly cloudy whatever it is.

And make them symbolic wherever possible example it Is a we do not want to specify just numbers alone so these kind of in between like those things will make it easy now we can also use package prefixes in the command names and global variables for example instead of these stations we can see with the other stations or if the state weather station is xxx we can say like weather stations acceptance -xxx they are saying like this exact same weather report and then maybe play think like that basically like we can use the package prefixes so this way like I mean it allows you to allows for the packages to coexist without any mean rushes.

(Refer Slide Time: 15:23)

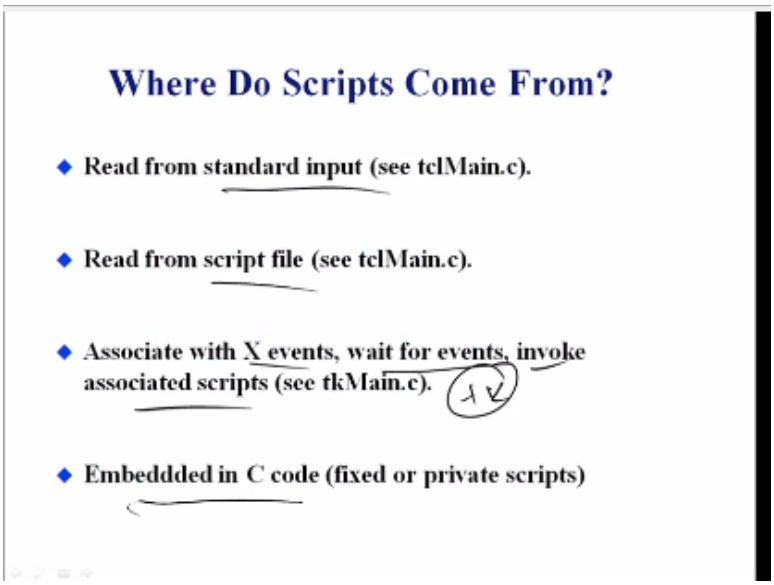


Now let us talk about the interpreters so that is the basic one and now the interpreters, so TCL inter ,interpreter, interpreter this is a structure that captures that encapsulate is they executing a execution state what are the execution States it is as variables the commands implemented in C the TCL procedure the execution stack so it encapsulate this entire the different states you can have many interpreters in a single application.

But usually just only one is active so the way to create interpreter is TCL interrupt and use a * or * inter this basically creates the structure and then we can just point that structure to the create command create an inter comment so once we have the structure then you can say basically inter people to TCL create inter and this is just an object or in command which now creates your inter and then if you want to delete use the procedure.

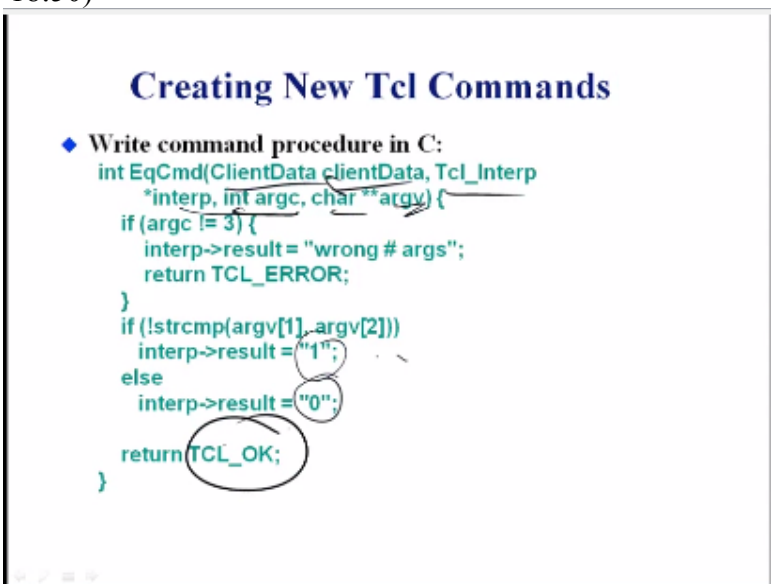
TCL get into I in P T so this will delete the interpreter and if you want to execute TCL scripts we basically like define integer code and then code= TCL valve inter set a = 1 or set a 1 now it evaluates this 1 and then sets the value of that exit code the code typically indicates the success of failure is a TCL okay normal completion TCL error, error occurred another way to do it is this set a1 you can put it inside the init.TCL and then you can say TCL Eval file enter any init. TCL and there is yet another third way to do the TCL valuation which is inter set a1 are three different strings , so now the inter preserved points to the string the result or an error message and then the application will display the result or message for the museum okay.

(Refer Slide Time: 18:24)



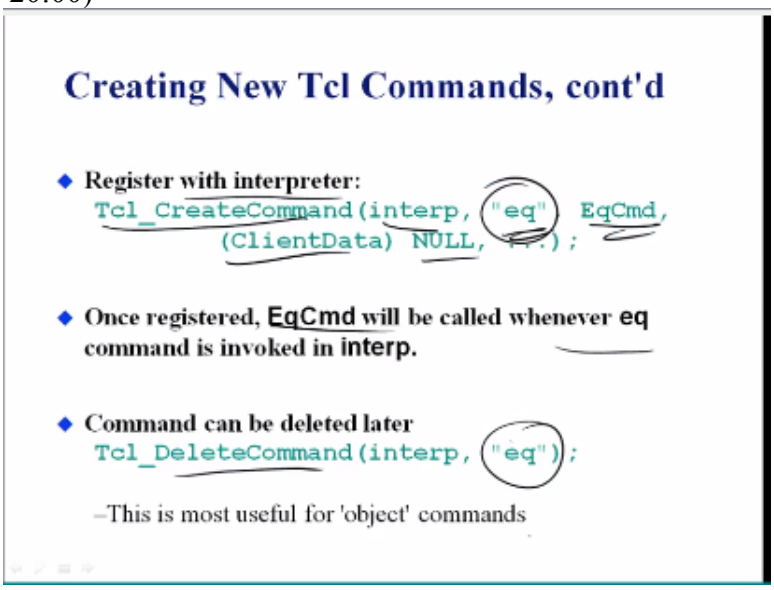
And then where do scripts come from this again you can read from the standard input read from the script file or associate X events and wait for the events invoke the Associated scripts with this the TK and then embedded in the sequel this is the fixed or private scripts that will be talked about.

(Refer Slide Time: 18:50)

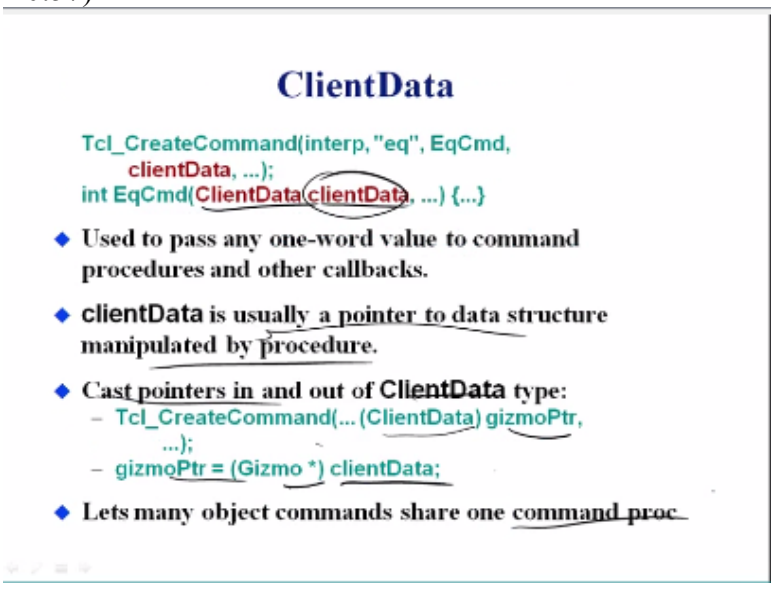


So creating this new TCL commands essentially so we write the command procedure in C EQ command client data the clan data TCL intro where we create that one basically like then we have integer R key and the char * DRG so this is not in C it is written so if it is < tree or like not = 3 then we just say like wrong arguments, otherwise we go and like compare the argv1 and argv2 of here and then if they are not the same then result is 1 otherwise result is 0.

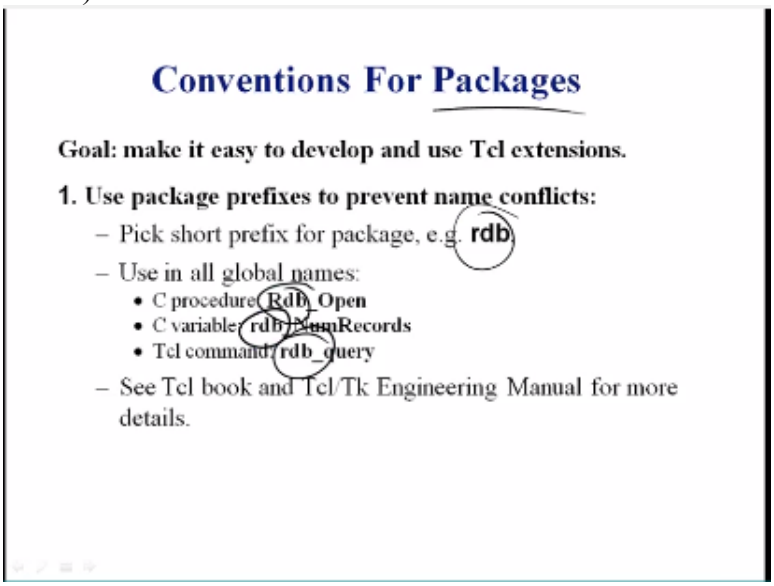
So this is basically to compare 2 commands to see whether it is the same or not and then we return the written to the mean.
(Refer Slide Time: 20:00)



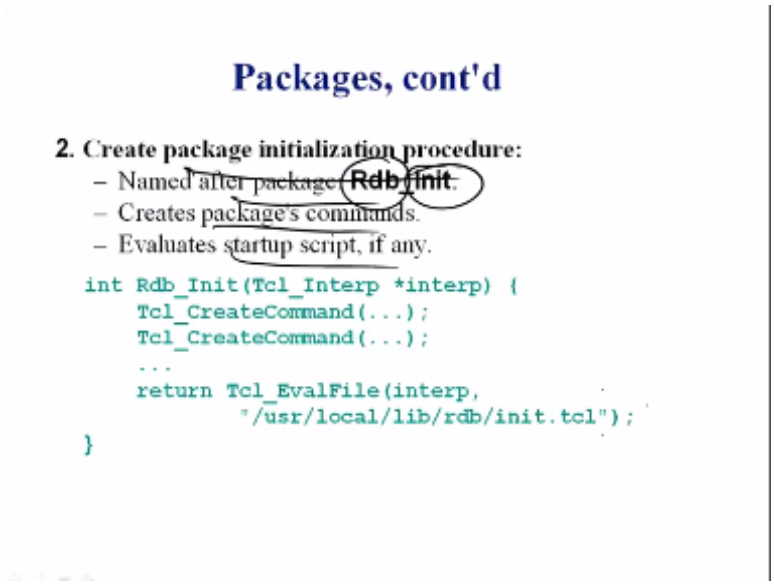
And then once we create this comment procedure then we register this with the interpreter that is TCL create some an inter EQ, EQ command client data and none so the echo command is now called EQ which is essentially a string compare now once we register this the EQ command will be called when our EQ command is invoked the intro and the interpreter you can also be delete you can delete the command by just giving the TCL delete command enter you so this is the most useful for object-oriented comments of the comments.
(Refer Slide Time: 20:57)



Now so we talked about the client data , what is client data so in the TCL command file here so we talked about the client data output is client data and then no expanding so here we talk about it again it is used to pass any one word value to the command procedures and call back so the Second client data is usually the pointer to the data structure manipulated by the procedure.
So we pass the first one and then the second one is manipulated by the procedure then and then the cache pointers in and out of the client data types this is so we can say like client data is no point to and then pointer we can define that basically * and then the client data, so we can have many objects share one command procedure by using this, so this is how we write new commands and registered you commands with the interpreter.
(Refer Slide Time: 22:38)

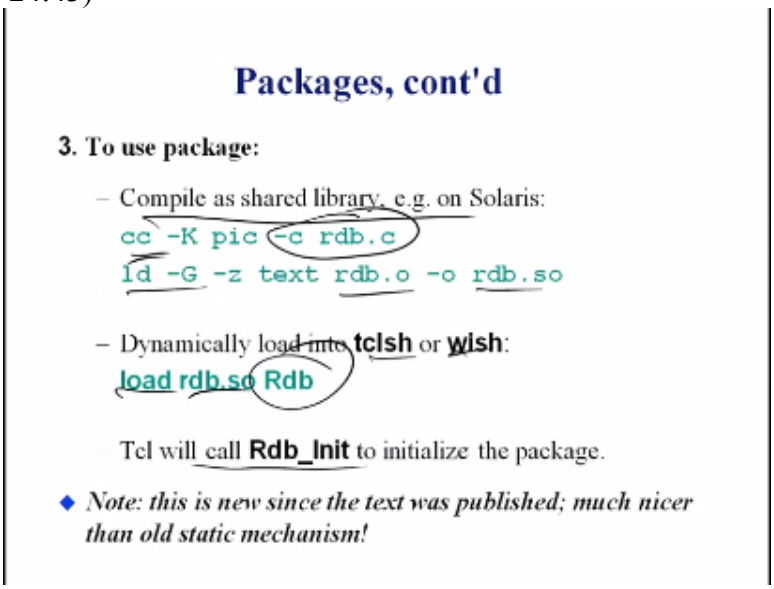


So now let us look at the packages so our goal is now to make it easy to develop and use TCL extensions so we can use the package prefixes to prevent name conflicts so there will be one of the things that we talked about earlier one right there left sides let me use a package prefix ,so we can pick a short prefix or a package for example RDB so we use this in all the global names essentially so in C procedure its uppercase RDB open C variable is RDB and then the TCL command is also.
(Refer Slide Time: 23:34)



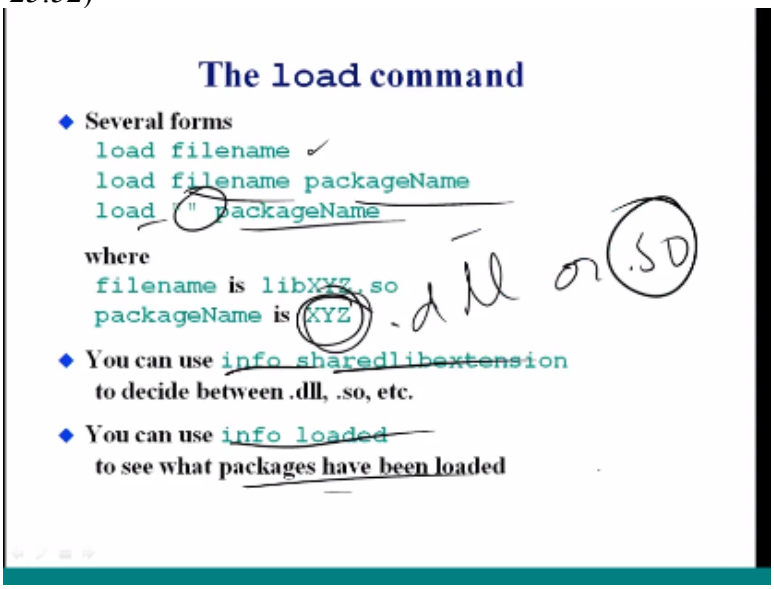
So now once we have specified on use the package prefixes for now we create the package initialization procedure and it is named after the package so our RDB- is the packaging is the initialization procedure this creates packages commands and then evaluate any startup script if we specify so here is an example with this int RDB init TCL interpreter in ERB then we create the TCL command first create command TCL command and then return to return TCL Eval file With that then what they need doc TCL.

(Refer Slide Time: 24:43)

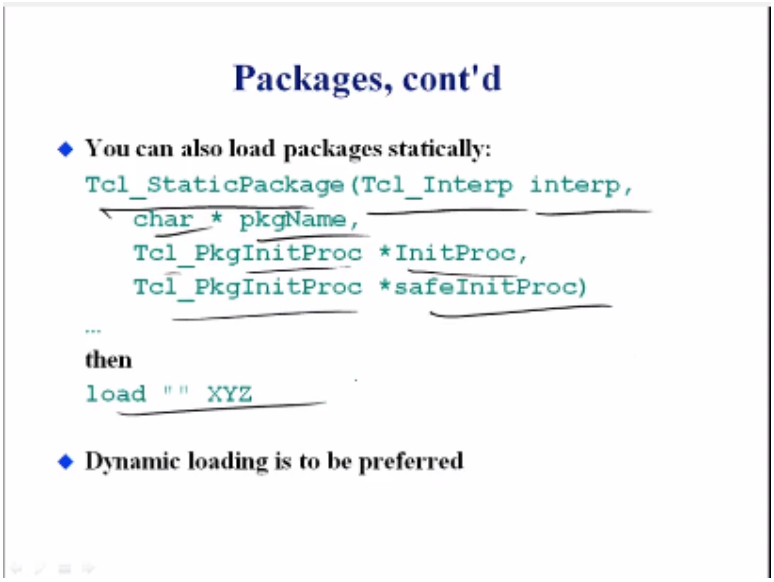


Now if you want to use the package basically we need to compile this at the shared library so here we do the CC compiled so RDV.C compile that and then we load it as RDB .O and output is RDB. SO, now we can dynamically load into the TCLSH you are wish so to load it basically

load RDB Row and then the table RDB so TCL will call the RDB init to initialize the path package.
(Refer Slide Time: 25:52)

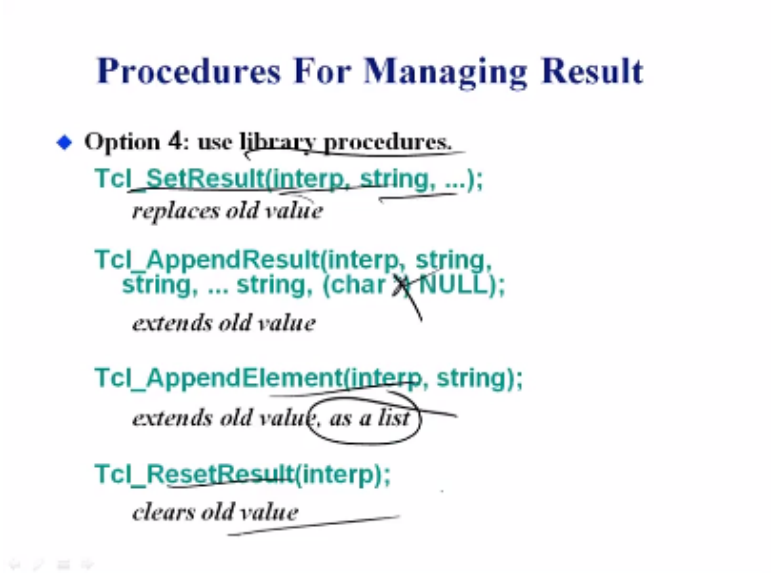


So this dynamic initialization is fairly new I mean you ask in the versions we will talk about today and we are TCL/ TK what versions are there what we could use, so now let us talk about the load command the load has several forms load file name load file name package name load nothing package name, here the file name is the name of the file basically here you have the letters RTB. So, that will be the file name the package that will be the file name and then the package name is the top-level package specification that is inside.
Which is in this case is RDB so in this case it be the x value, so we whether we load it as a dll dynamic link library or FO. SO what I saw is we can use info shared life extension to decide between systems we can also use info load and to see what packages have been loaded into the system.
(Refer Slide Time: 27:45)



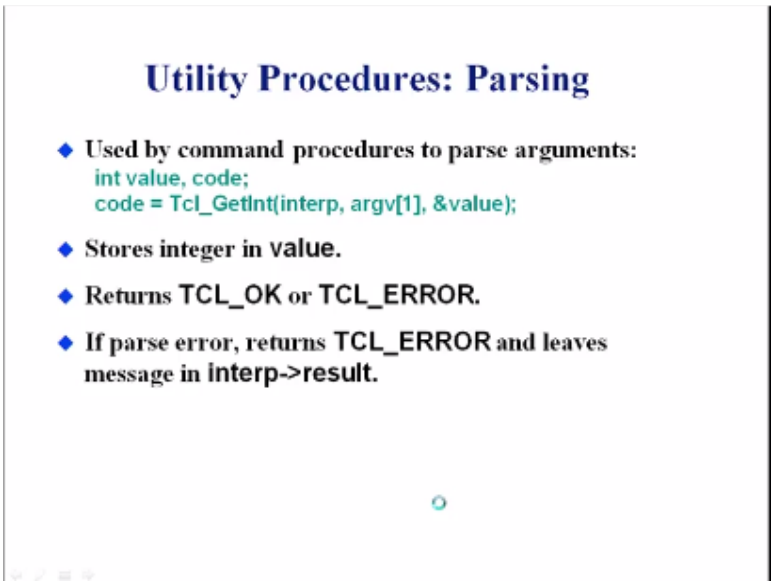
If you want to load a program strategy then we specify that TCL static package is TCL prefer interpret and character * packaging so that points to the thing and then we also specify the TCL package unit proc this is the star init proc and then TCL academic proc star state info once we specify this then we can say load expecting, of course in this scenario the dynamic loading is preferred.

(Refer Slide Time: 28:31)



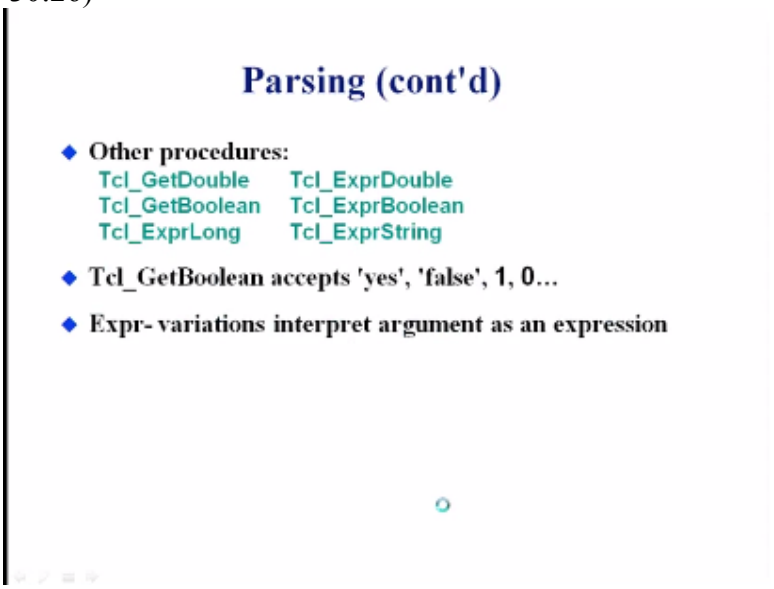
Now the fourth option for managing results if you use library procedures so TCL result set result interpreter strain and this will replace the old value and if you pay like to append a result then that extends the old value and if you do append element then it extends the old value but now as a list if you do use the reset result now that clears the old value.

(Refer Slide Time: 29:19)



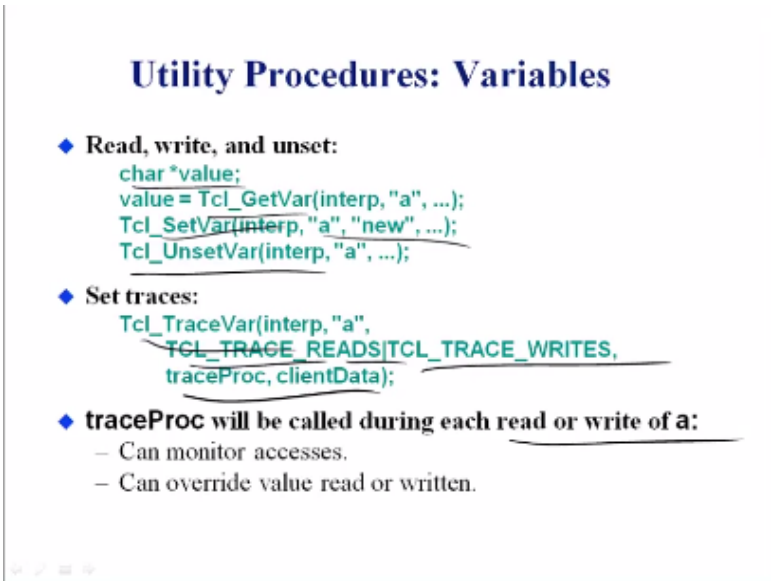
Now let us look at the utility procedures and one of them is parsing this is used by command procedures to parse arguments usually it takes in a value and a code the code is just to TCL get init interpreter RB1 and then value , it stores the integer in value and it returns TCL okay our paper you know if there is a parse error then it returns a TCL and leave message in interpreting result.

(Refer Slide Time: 30:26)



Other useful procedures it a TCL get double TCL expression double it will get Boolean expression, Boolean expression long and expressions, so the TCL Boolean accepts s false 1 0 extra and expR is essentially like variations interpret argument as an exception.

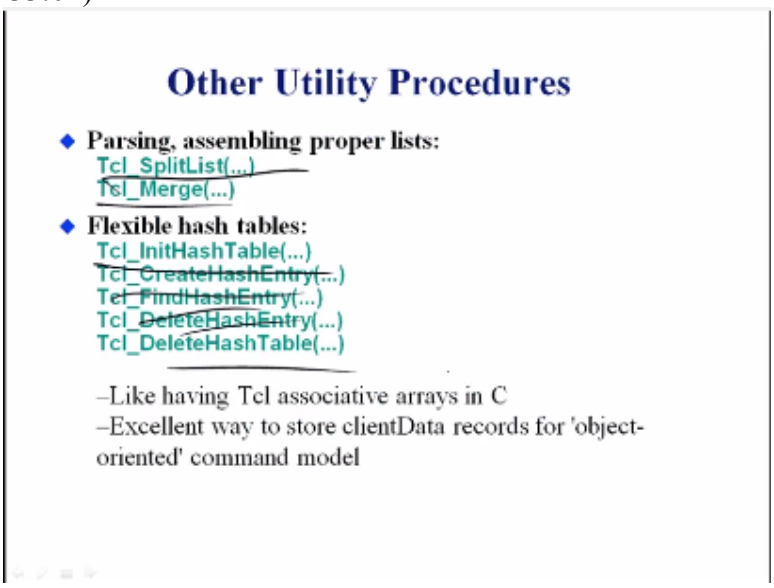
(Refer Slide Time: 31:23)



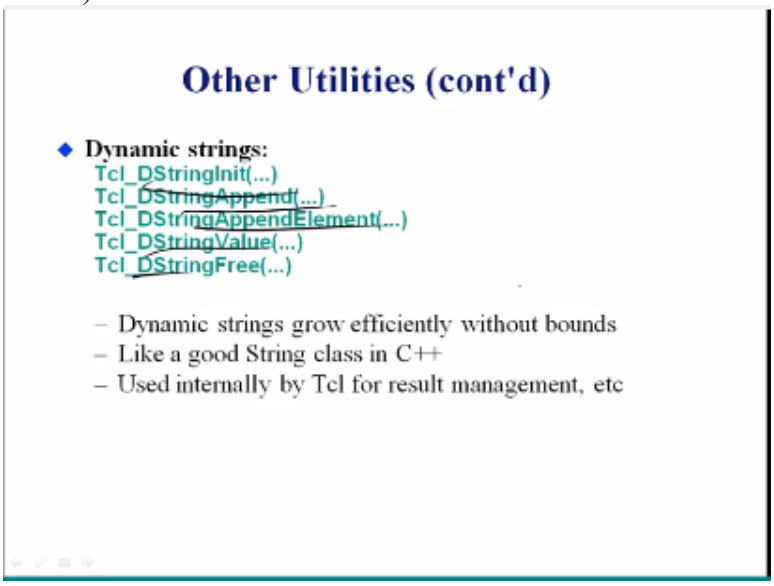
So in utility procedures what are variables essentially so basically they can be read Wright or unset, so you had an example basically like we have a character value, Value is TCL get war interpreter a something and then TCL set to are interpret something and then unset something else so we can use get set and unset if you can pretty much rewrite in upset then you can also set traces basically.

So that is can face a variable so that is trace war and then we can say okay TCL trace weeds to play wipes trace procedure and planned it so we can trace it through how the value actually is changing broader problem throughout the script so and then the trade prog is it will be called basically to doing each read or write of a they can you can monitor the accesses and override value and read or the value are you red orbit.

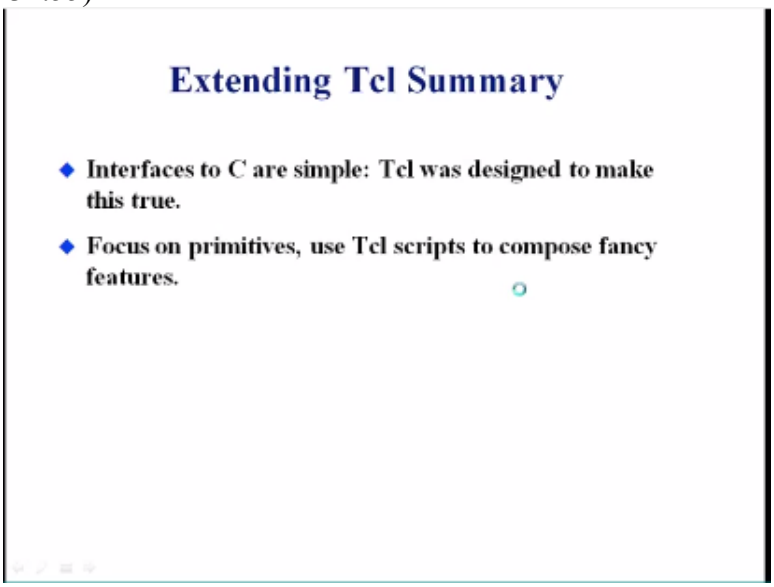
(Refer Slide Time: 33:02)



Now for passing assembling proper lists we use the TCL straight list and TCL merge the hash tables are quite flexible basically it actually will create identity finder gents will attach entry and delete the whole hash table, the hash table is just like having TCL associative arrays in C it is also excellent way to store go find data records for object oriented command.
(Refer Slide Time: 33:59)



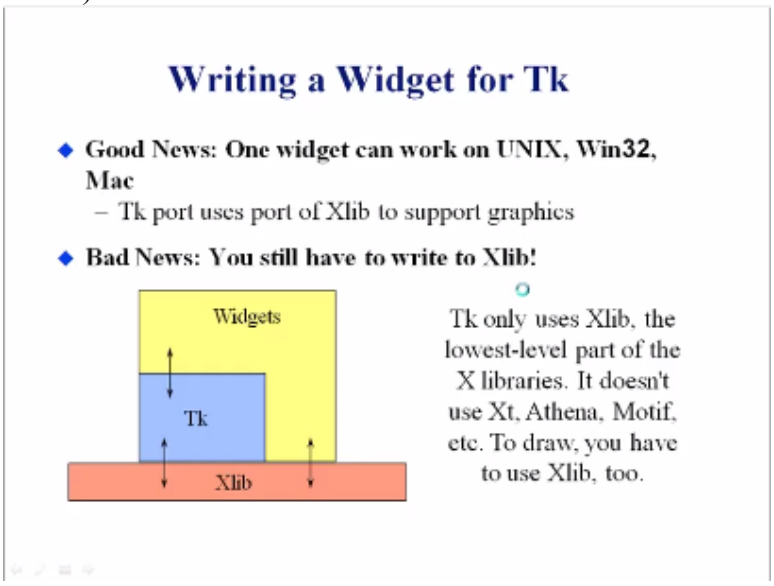
And then dynamic strings is through this these bridges these two minute this should happen this , this thing B string method element reaching value and then finally do simply the dynamic strings go efficiently without bonds like a good string class in C++ it is used internally by TCL for result management so this is again another data structure within.
(Refer Slide Time: 34:55)



So in terms of extending TCL interfaces to see itself are visible So TCL is designed to make this true and when we write when the extent TCL we focus on the primitives and use to go script

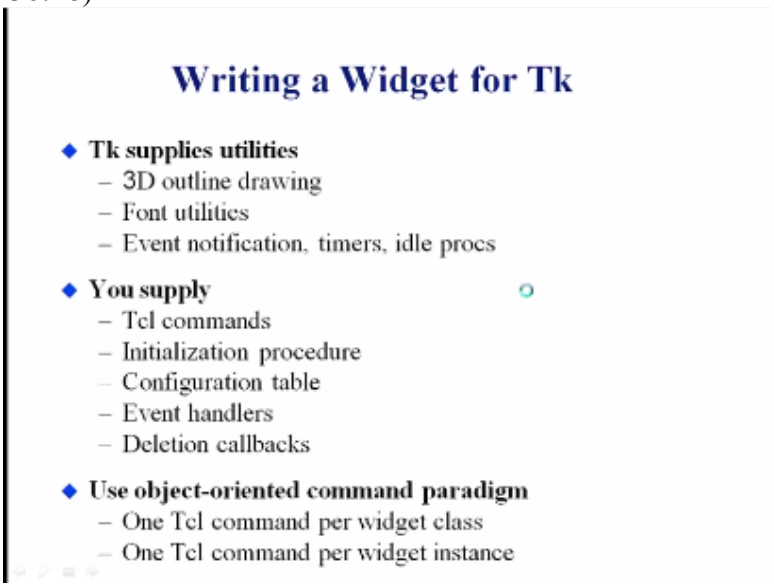
compost fancy pages so we write the most primitive procedures first and then we basically combine the features of TCL to commend multiple primitives into a movement all that the larger comment.

(Refer Slide Time: 35:27)



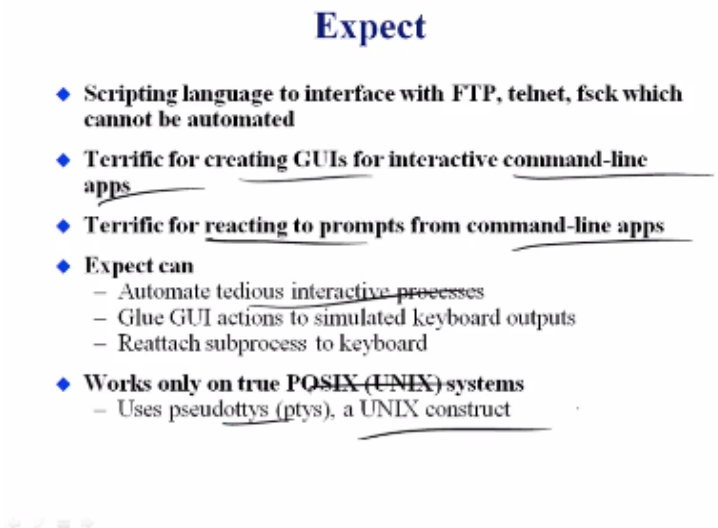
So if you are writing a widget for TK the good news is that one wage in a one widget that can work on UNIX windows etc so that is your this one but so the TK port uses the port of exit to support that so it is ago but what is the bad news is that you will have to write to excellent so TK actually uses only excellent this is the lowest level part of the X libraries it does not use XP yet now motive for etcetera so to draw you have the X Lib also.

(Refer Slide Time: 36:18)

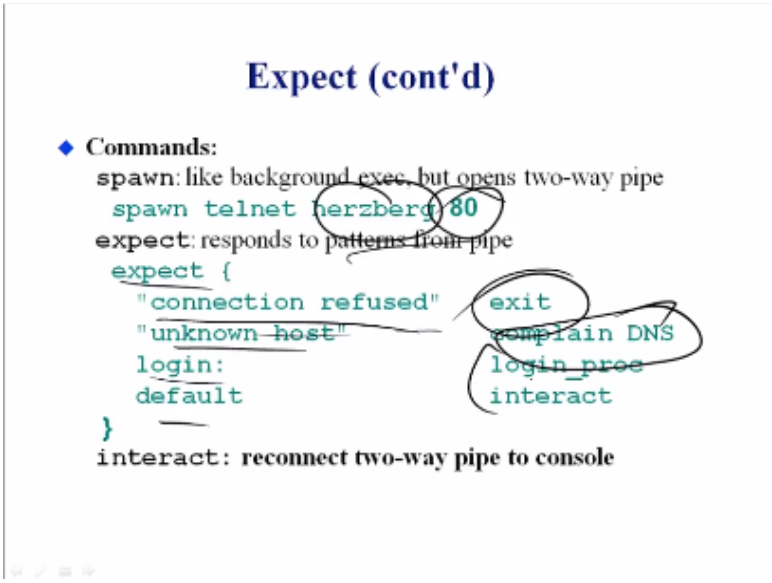


So TK supplies until things for example the 3d drawing the font utilities even notification timers I have the idol proc successful and then you have to supply the TCL commands the initialization

procedure configuration table even handlers and deletion callbacks then we use object-oriented command paradigm that is one TCL command per widget class and one TCL command per widget instance.
(Refer Slide Time: 37:05)

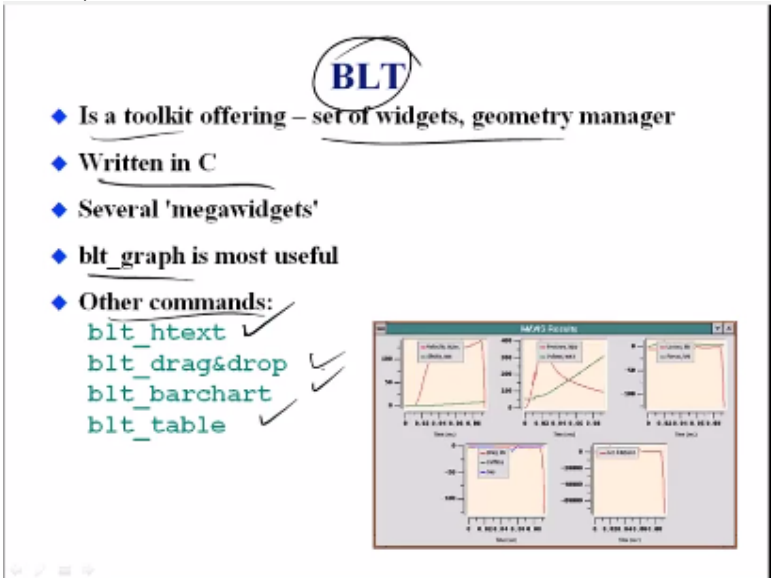


The now I will talk about a couple of other meta languages or meta scripts basically one such thing is called the expect the expect is really the scripting language to interface with FTP telnet fsck etcetera which cannot be automated , so one way to it is also a very good way to create the GOI or interactive command-line applications it's also good way to react to prompts from the command line and usually they expect and automate tedious directed processes.
Glue GUI actions to simulate keyboard outputs and reattach the subclasses to ego and one thing notice it works only on true topics or the unit systems so uses the pseudo TK this EVP device the unit constructed that is the reason why it does not work well in others.
(Refer Slide Time: 38:27)



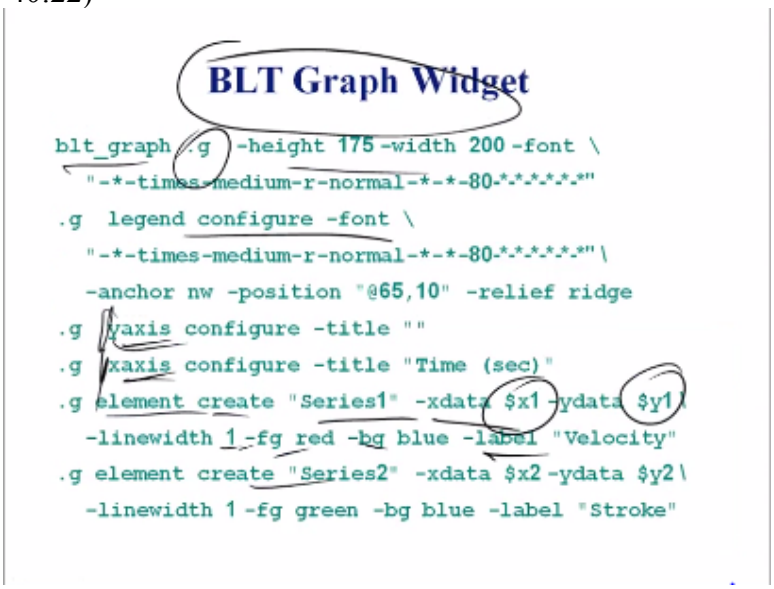
So expect the commands are like this basically will be the second called spawn you can say basically it is telnet and which machine it is the server and then what is the port number , so they expect responds to patterns from pipe for example they expect can say electronic connection refused unknown host login default so when connection that use this exact means that exit unknown host is the DNS for complain Then we have the login reconnect two-way pipe to the console.

(Refer Slide Time: 39:22)

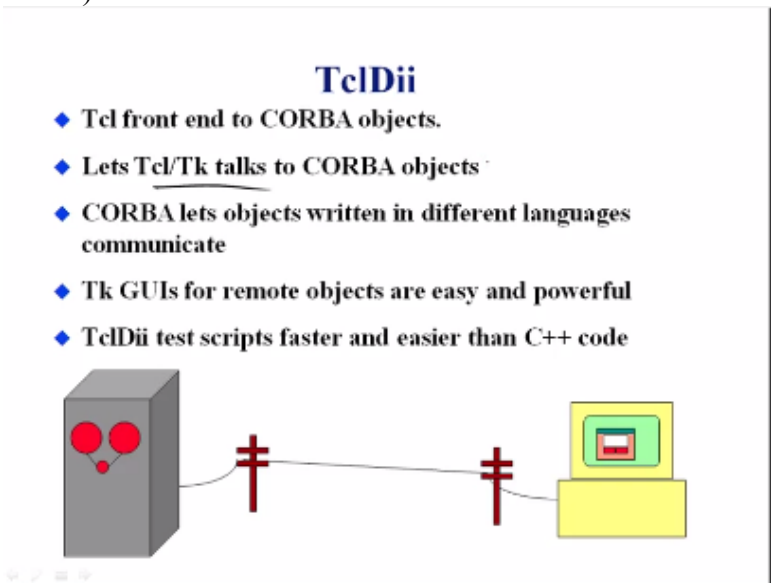


So the other one is the BLT essentially this is also it's a toolkit offering set of widgets and geometry manager so this helps you to brought pictures display graphs very easy and this particular toolkit offering is written in C it has several mega widgets the beauty graph is one of the sub one and that is the most useful one and more value one that is for very popular also then

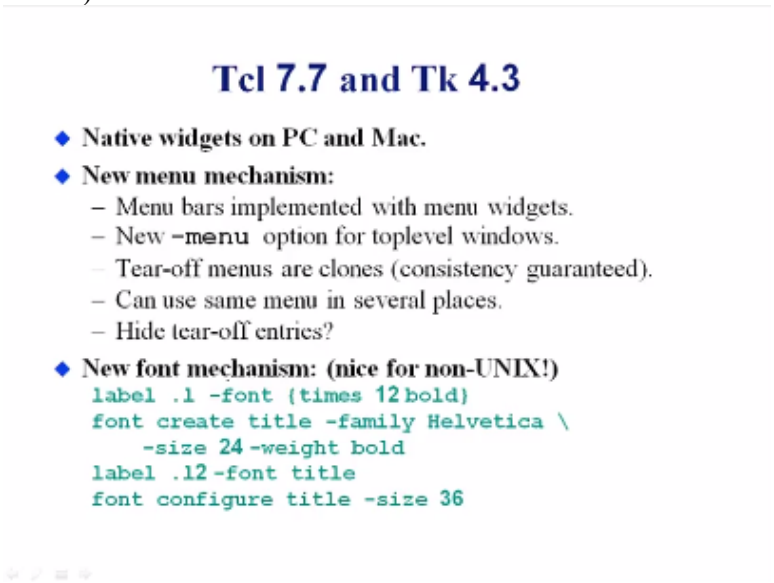
other commands basically like highlight text drag and drop bar chart table things like that can easily built using it.
(Refer Slide Time: 40:22)



So to invoke a BLT graph widget we specified a BLT graph command with the G as the widget name and then we specify all the other things then here we are dis considering the .G and then we can configure the x-axis and y-axis as to what the value should be here the x-axis is pretty much its time and y-axis one thing , so once we create the configure this and then we create the weight axis the x axis and the y x axis and the y axis.
We then create the elements again enable pH c1 X data is dollar X 1 Y data is y 1 and then what is the line width or ground background and then similarly we can plot another set of data as well so that is how the BLT graph with this is used which is much very common.
(Refer Slide Time: 41:48)

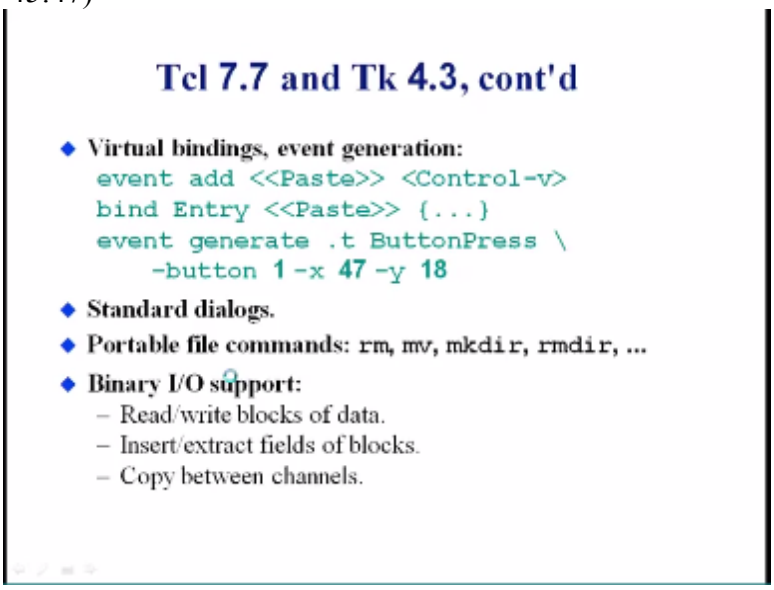


Now let us look at one more thing which is the TCL I sincerely this is usually the front end to cover projects the it let us TCL/ TK talks directly to the core object the Cobra lets objects written in different languages communicate extension so code by two sitting in the center and it gets the objects in from various languages and it computes, the TKDI is TK GUI for remote objects are easy and very powerful TK the scripts are faster and easier than cheapest than C++ so okay so this, this is pretty much the technical aspect of it of the TCL/ TK.
(Refer Slide Time: 42:55)

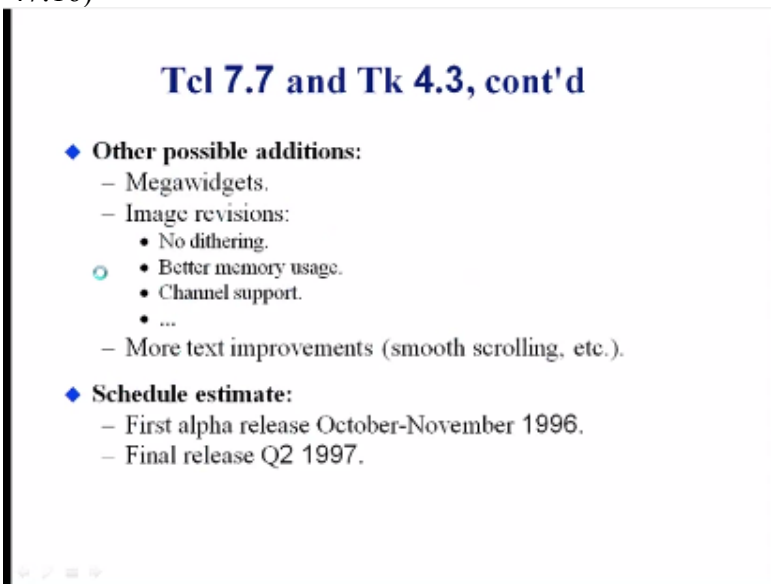


Now let us look at some of the releases and want painting to the lathe and what is the latest in position to 7.6 and TK 4.2 was originally there and this one has revision of critter needed for spectacle FCS CAPI change for, actually the debate original happened so move on statement there and then 7.7+TK4.3 which is which is have features needed for Netscape plug-in and you can just use the plug-in information.
And then safety of them in already read in plug-in binaries , and no source release until TCL 7.7 and TK four which is also like pasta so 7.3 is essentially 7.5 is the latest greatest sorry not this we will talk about that put after 7.6 the 7.7 came into being and tipping TK was 4.4 became 4.3 here basically now we had native widgets on PCs and Mac there is a new make a mini mechanical essentially.
The menu bars are implemented with many widgets there is a new menu option for the top levels the widows of menus are clones and it can use the same menu in certain places and hiding the pair of entries in some other issue so the new font mechanism basically like this is very nice or non-use machines basically we specify label and then $ L I mean first. L and font times twelve

board and then we say font create idol family Helvetica size 24 and various code Then the label basically to title and then font configure Peggy and size is 36.
(Refer Slide Time: 45:47)

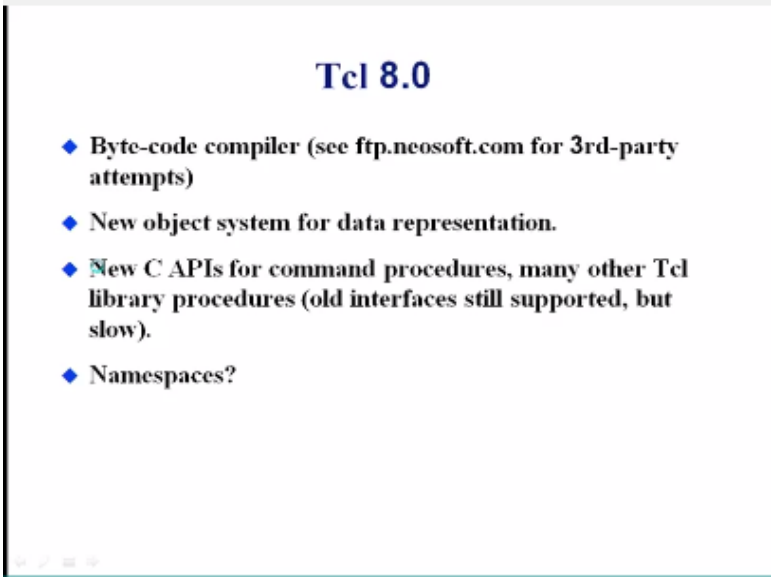


So the other things we have basically virtual bindings essentially in even generation so this is something that we didn't touch upon the TK windows can create a virtual bindings essentially meaning like virtual events which can be bound inside so you don't really need to do all the action that, the Tool kit expecting, now it also supports the standard dialogue the portable file commands are , make the etcetera it also supports the binary i/o So read or Wright blocks of data instead extract fields of blocks and then copy between channels.
(Refer Slide Time: 47:16)



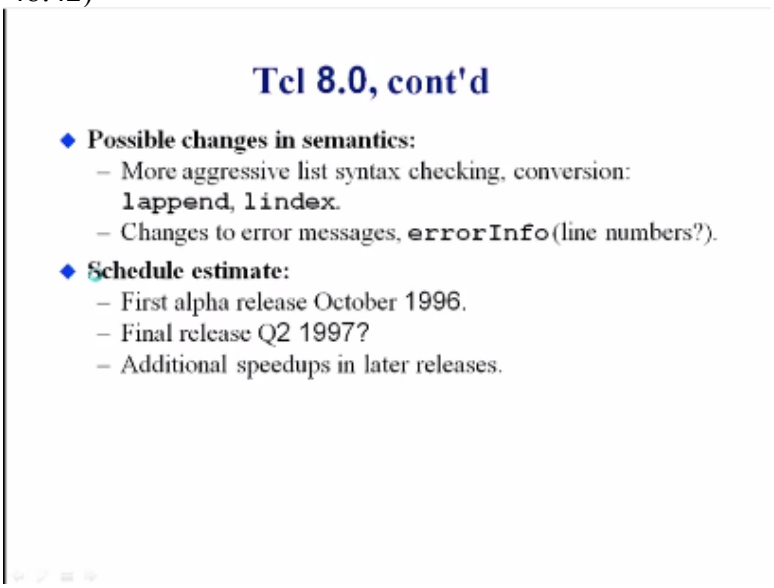
And then other possible extensions that 10.7 and TK4.3 mega widgets image revisions, in the channel support and then finally more text improvements, so these are the entire original dates essentially.

(Refer Slide Time: 47:45)

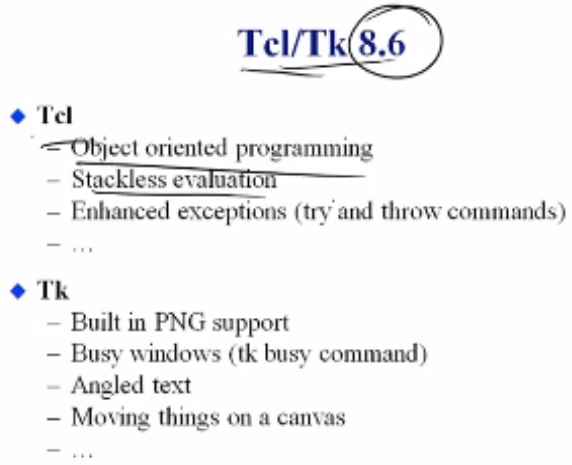


But then, the next one is the TCL 8.0 this one has a byte code compiler and this is the FTP dot command of calm they should have it and then we have a new object system for data representation the new seed CAPI or a command procedures many other single library procedures the old interface is still not supported but with a slow I mean it is still supported but it is more and then what about the namespaces basically like that kind of a thing that we need strong we need to worry about, okay.

(Refer Slide Time: 48:42)

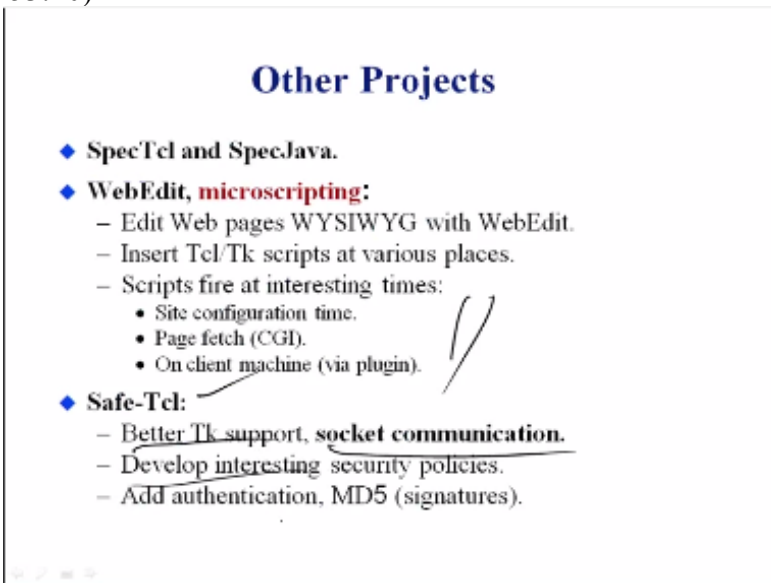


So in TCL 8.0 and also made some possible changes to the semantics the more a gratitude list syntax checking are enabled in TCL and then the conversion for error info and the box.
(Refer Slide Time: 49:58)



So for the TP 8.2 you have the blade code compiler essentially and you can go to the FTP neosoft.com for the third-party items there is a new object system for the data representation the new C CAPI for command procedures and many others I see this so here the old addresses are still supported but they are very slow and what about the namespaces and then there is also like some changes to the semantics a more aggressive list syntax ticking.
Basically than the L index and then changes to the error messages error info and then line numbers so the view schedule basically like these are all different so the latest one is the TCL is TCL /TK 8.6 TCL supports object-oriented programming it has a stack less evaluation so when the procedures are called and procedures that nested the nested procedures are all in the sequence previously it used to use a stack or the execution.
Basically it stores all the state in a stack and then it executes the next one and if that also nests another procedure it retains everything in to understand and then go on so in the new one that particular thing has changed basically like that that is fall that was found to be like family inefficient so it is changed and then it's a stack of execution in TCL 8.6 it also has enhanced exception handling the commands called try and throw.
Which attempts to actually catch an exception and then actually also like to change the course when then exception happens ,okay and then there are many other enhancements to, to the TCL 8.6 ,on the TK front there is a built-in support or PNG format for the graphics there is also a new

command for busy windows essentially it's called TK busy command there is support for angled text and moving things on came as.
(Refer Slide Time: 53:10)



So and other projects basically the spectacle spec Java debated micro scripting basically like we can use we can edit web pages this what you see what you get kind of the mode using the web edit we can insert TCL TK scripts maybe spaces and then this script fire at interesting times essentially and then the TCL basically which is so as a better PK it has little PK support the socket communication and developing in interesting and interesting security policies.
In add authentication for example the md5 signatures so that's pretty much what I wanted to cover in terms of PK so today mainly like I mean we looked at several areas basically so in once again like I just wanted to summarize four things that we did for today , so mainly like I mean we talked about the extending TCL with the T we also looked at the third party TCL extensions that expect PLT and TCL and then we talked about the past and the future and the presence of TCL and TK which is essentially like which versions what is what are supported what can you expect in the future so I think this concludes this lecture once again thank you very much for listening thank you.