(۱-۱) این کتاب راجع به دو بسته نرم افزاری به نام Tel , Tk می باشد. که با همدیگر یک سیستم برنامه نویسی برای گسترش و استفاده از واسط گرافیکی کاربرد (GUI) را فراهم می کند. Tel بیانگر زبان ابزار فرمان است و با نام "Tiche" شناخته شده است و یک زبان اسکریپتی ساده برای کنترل و توسعه برنامه های کاربردی می باشد. Tel یک برنامه نویسی کلی و عمومی را با استفاده از ابزارهای سودمند از قبیل متغیرها، حلقه ها و توابع برای برنامه های کاربردی متنوع هم می آورد. بعلاوه Tel به صورت توکا (embeobble) می باشد. یعنی مفسر آن بعنوان کتابخانه ای از توابع "C" که می تواند به راحتی داخل برنامه های کاربردی قرار داده شود پیاده سازی شود. و هر بنامه ای کاربردی می توان خصوصیات بنیادی Tel را با فرمانهای افزودنی بخصوصی با برنامه های کاربردی گسترش دهد.

یکی از مفیدترین ابزارهایی که برای گسترش Tel به کار گرفته شده است Tk می باشد. Tk یک بسته از ابزارهای نرم افزاری برای سیستم های ویندوز با ورژنهای مختلف می باشد. و تلفظ نام آن "TeEkey" می باشد. Tk وسایل و امکانات Tel را با فرمانهای افزودنی برای ساختن واسطه های کاربر گسترش می دهد. بنابراین شما می توانید ساختها واسطه های کاربر را برای استفاده از کدهای زبان C با اسکریپتهای Tcl بنویسید. Tk نیز مانند TCl بعنوان توابع کتابخانه ای C

پیاده سازی شده است. بنابراین در خیلی از برنامه های کاربردی گوناگون می تواند مورد استفاده قرار گیرئد. برنامه های کاربردی نیز می توانند براساس خصوصیات Tk با واسطه های اشیاء گرافیکی کاربرد (widgest) و مدیریت هندسی در C گسترش یابند. Tel و Tk با همدیگر ٤ مزیت را برای کاربران و توسعه دهندگان کاربردی ایجاد می کنند ابتدا اینکه Tel داشتن یک زبان اسکریپتی قدرتمند را برای هر برنامه‌ی کاربردی آسان می کند. تمام چیزی را که یک برنامه کاربردی نیاز است انجام دهد پیاده سازی تعدادی از فرامین Tel است که ویژگی های اصلی برنامه های کاربردی را تامین می کند. پس برنامه کاربردی میتواند با مفسر tel با پروسی جدها، یک تابع کامل زبان اسکریپی که شامل هر دو مورد فرمان های تهیه شده به وسیله Tel (که هسته Tel (Tel Core) نامیده می شوند) شکل ۱. ۱ را ببینید.

جای شکل

Figure (۱-۲): برای ایجاد کردن یک برنامه کاربردی جدید براساس Tel، یک گسترش دهنده برنامه کاربردی یک ساختمان داده های ویژه C برای آن برنامه

کاربردی طراحی می کند و کدهای C را برای پیاده سازی تعداد کمی از فرامین Tel پیاده سازی می کند. کتابخانه Tel هر امکانی را فراهم می کند در غیر این صورت Tel نیازمند به یک زبان فرمان گرا قابل برنامه نویسی است.

برنامه کاربردی می تواند با نوشتن اسکریپتهای TCl اصلاح شده و گسترش یابد. برای مثال، یک برنامه کاربردی برای خودداندن تابلوهای اعلانات (بولتن) الکترونیکی ممکن است شامل کدهای C باشد. که یک فرمان Tel را برای جستجو و کاوش در تابلو اعلانات و صدور یک پیغام جدید برای این کار و یک فرمان دیگر را برای پیغام داده شده پیاده سازی می کند. زمانی که از این فرمان ها خارجی می شویم اسکریپتهای Tel می توانند برای تکرار این چرخه در لابه لای پیغام های جدیدی از تمام تابلوهای اعلانات نوشته شوند و همه آنها را در یک زمان نمایش دهند یا فایل پیغامهایی که خوانده شده اند یا خوانده نشده اند را بر روی دیسک ذخیره کرده و نگهداری کنند. و یا اینکه یک یا چند تابلو اعلانات را برای موضوع ویژه ای جستجو کنند.

برنامه کاربردی تابلو اعلانات نیازی نیست هر یک از توابع افزودنی را در C پیاده سازی کند. بلکه همه این توابع می توانستند به عنوان اسکریپتهای Tel پیاده سازی

می شوند و کاربران این برنامه های کاربردی می توانستند اسکریپتهای افزودنی Tel را برای افزودن توابع بیشتر به برنامه کاربردی بنویسند.

مزیت دوم Tel و Tk قابلیت توسعه و گسترش سریع آن می باشد. برای مثال فیلمی از برنامه های کاربردی پنجره ای جالب می توانند تماماً به وسیله اسکریپتهای Tel نوشته شوند. بدون اینکه ابداً از کدهای C استفاده شود. این کار با استفاده ای یک واسط کاربرد (Shell) پنجره ای که Wish نامیده می شود انجام می شود. این کار به شما اجازه می دهد که در سطح بالاتری نسبت به وقتی در C یا C++ هستید برنامه نویسی کنید و خیلی از جزئیاتی که برنامه نویسان باد مخاطب آن باشند و از شما پوشیده است. در مقایسه ای نسبت به ابزارها وقتی شما تماماً در C برنامه نویسی می کنید یادگیری استفاده از Tel و Tk آسانتر است و نیاز به کدنویسی کمتری دارد. کاربران مبتدی Tel و Tk می توانند فقط چند ساعت پس از یادگیری آن واسطه های کاربر جالبی ایجاد کنند. و به گفته خیلی از کاربران وقتی آنها Tel و Tk را جایگزین دیگر ابزارهای نرم افزار کرده اند با کاهش قابل ملاحظه ای تا ده برابر در حجم کد و زمان لازم برای گسترش برنامه روبرو شده اند. دلیل دیگری برای توسعه سریع Tel/Tk مفسری بودن زبان Tel می باشد. وقتی شما یک برنامه کاربردی Tel از قبیل Wish را استفاده می کنید می توانید اسکریپتهای جدیدی تولید و اجرا کنید بدون اینکه لازم باشد آن برنامه

کاربردی مجدداً اجرا یا کامپایل شوند. این کار به شما اجازه می دهد خیلی سریع آن را تست کنید. و خطاهای (باس) آن را مشخص سازند. زمانی که Tel تفسیر می شود خیلی کندتر از کدهای کامپایل شده C اجرا می شود. ولی ایستگاه های کاری و سیستم های جدید به طور باورنکردنی سریع هستند. برای مثال شما می توانید اسکریپتی با صدها و حتی هاران فرمان Tel را با یک حرکت ماوس بدون اینکه تاخیر قابل ملاحظه ای داشته باشد اجرا کنید. در مورد موارد نادری که به نحوه اجرای مسأله قابل ملاحظه ای است شما می توانید بیشتر اجراها را در قسمت های مهم بحرانی اسکریپتی Tel مجدداص در C پیاده سازی می کنند.

سازماندهی کتاب: مزیت سوم Tel مزیتی است که آن را به یک «زبان چسبنده» بسیار خوب تبدیل کرده است. چون Tel قابل تعبیه شدن در برنامه های دیگر می باشد. Tel می تواند در برنامه های مختلف و به منظور تحقق اهداف مختلف به کار گرفته شود. می توان یک اسکریپت Tel را طوری نوشت که تمام خصوصیات برنامه ها را در بر داشته باشد. برای مثال هر برنامه کاربردی پنجره Tk می تواند یک اسکریپت Tel را به هر برنامه کاربردی Tk انتشار دهد. این ویژگی جلوه های چند رسانه ای (Multi- medis effects) را خیلی بیشتر در دسترس قرار م یدهد در زمانی که برنامه های کاربردی صوتی و تصویری با Tk ساخته شده اند

(و هم اکنون موجودند) برنامه Tk می تواند فرامین اجرا و ضبط را برای آنها صادر کند. صفحات گسترده می توانند خودشان را به وسیله پایگاه داده برنامه های کاربردی به روز (upobte) کنند. ویراستارهای واسطه های کاربر می توانند نمایش و رفتار برنام های کاربردی را در حالت اجرا و غیره اصلاح کنند. Tel یک Franea lingua را تهیه می کند که اجازه می دهد برنامه های کاربردی با همدیگر کار کنند. مزیت چهارم Tel راحتی کاربر است. زمانی که یک کاربر Tel و Tk را یاد می گیرد او می تواند صرفاً با یک یادگیری چند فرمان خاص برنامه کاربردی اسکریپتهایی برای هر برنامه کاربردی Tel و Tk بنویسید و در برنامه های کاربردی جدیدی ایجاد کند. این مزیت سفارشی کردن و بهبود برنامه های کاربردی را برای کاربران امکانپذیر می سازد.

۲-۱) سازماندهی کتاب

فصل ۲ از چندین اسکریپت ساده برای مروری سریع بر مهمترین ویژگی های Tel و Tk استفاده می کند و یک سیستم مناسب و راحت به شما معرفی می کند و آنها بدون هیچ توضیحی در مورد جزئیات مفیدند. باقیمانده کتاب دوباره حالت جامع تری دارد. و به ٤ بخش تقسیم یم شوند.

I) معرفی زبان اسکریپتی Tel. بعد از خواندن این بخش شما قادرید اسکریپتهایی برای برنامه های کاربردی Tel بنویسید.

II) این قسمت فرمانهای افزودنی Tel را توصیف می کند. که به وسیله Tk آماده شده اند و به شما اجازه می دهند به واسطه های کاربر اشیاء گرافیکی (widgets) مانند منوط و نوارهای پیمایشی را ایجاد کرده آنها را در پنجره برنامه کاربردی مرتب کنید. بعد از خواندن این قسمت شما قادر خواهید بود برنامه های کاربردی پنجره ای جدیدی مثل wish ایجاد کنید. و اسکریپتهایی برای بهبود برنامه های کاربردی Tk موجود بنویسید.

III) قسمت سوم در مورد پروسی جرهای C در کتابخانه Tel و نحوه استفاده از آنها برای ایجاد فرمان های Tel جدید بحث می کند. بعد از خواندن این قسمت شما قادر خواهید بود بسته های نرم افزاری (پکیج Tel(Package و برنامه های کاربردی در C بنویسید.

IV) قسمت چهارم کتابخانه پروسی خبرهای Tk را توصیف می کند. بعد از خواندن این قسمت شما قادرید اشیاء گرافیکی (widgets)جدیدی ایجاد کنید و آنها را در C مدیریت هندسی کنید.

هر یک از این بخش های اصلی شامل ۱۰ فصل کوتاه می باشند. هر فصل انتخاب شده قسمتی از سیتسم را توصیف می کند. لازم نیست تمام فصول را به ترتیب

بخوانید. من توصیه می کنم که شما خواندن فصول ۳-۹ را به سرعت شروعغ کنید. بعد از فصول XXX-YYY عبور کنید و بعد از آن فصل هایی را که به آن نیاز دارید بخوانید. در اینجا هیچ خصوصیت و ویژگی ناگفته نمی ماند. و تمام توضیحات طوری سازماندهی شده اند که یک مقدمه مرتب و روان را نسبت به مرجع مطالب ارائه دهند. یک مجموعه مجزا از مراجع با توزیع Tel و Tk موجود است. اینها خیلی مجهزترند. اما قطعاً هر ویژگی از هر دو سیستم را ارائه می دهند.

این کتاب فرض می کند که شما با زبان برنامه نویسی CANSIC استاندارد آشنا هستید و تنجربیاتی در مورد unix , Xl دارید. برای اینکه فصل چهار را بهتر درک کنید نیاز دارید خیلی از ویژگی های تهیه شده به وسیله xlib از قبیل گرافیک و خصوصیات پنجره ها را بفهممند. اما این جزئیات فقط در فصل ٤ مورد نیازند. شما قبل از اینکه خواندن این کتاب را شروع کنید نیازی نیست راجع به Tel و Tk چیزی بدانید. هر دوی اینها معرفی خواهند شد.

در تمام این کتاب من از فونت couvier برای نشان دادن هر چیزی که کامپیوتر تایپ می کند مثل نام متغیرها، پروسی جرها و نام فرمانها، اسکریپتهای Tel و کدهای C استفاده می کنیم.

مثالی از اسکریپت Tel در زیر آمده است:

```
set   Q        44
```

⇒ 44

فرمانهای Tel مثل "Set a 44" و نتیجه آن مثالی از نمایش با فونت couvier می باشد و نتیجه آن یعنی "44" که به صورت مایل نمایش داده شده است. علامت ⇒ قبل از نتیجه نشانگر مقدار برگردانده شده است. اگر خطایی در یک فرمان tel اتفاق بیفتد پیغام خطا با فونت کوریر مایل نمیش داده می شود و علامت ϕ در شروع پیغام درج می شود. تا نشان دهد که مقدار برگشت داده شده یک پیغام خطاست .55 set a 44 ϕwrong # args.Should be "set vorname? Newualve? وقتی ساختار دستوری فرمان های tel را شرح می دهیم از فونت کوریر مایل برا ینشان دادن نام آرگونامهای رسمی استفاده می کنیم. اگر یک یا چند آرگومان در درون علامت های سوال محصور شده باشند به معنی اختیاری بودن استفاده از آراگومان می باشد. برای مثال ساختار فرمان set در زیر آمده است.

Set war name? New volve?

و به معنی آن است که کلمه set باید کلمه به کلمه برای فراخوانی آن فرمان وارد شود. Varname , New value نام آرگومانهای فرمان set می باشند. هنگامی که بخواهید فرمانset را فراخوانی کنید باید یک نام متغیر به جای varname و

یک مقدار جدیدی برا یاین متغیر در عوض newline تایـپ کنیـد. آرگـون هـای newvale اختیاری می باشند.

فصل دوم:

مروری بر **Tel , Tk**

این فصل Tel , Tk را با یک سری از اسکریپتهای خواص اصلی سیستم ها را شرح می دهند معرفی می کنند. گرچه شما باید قادر باشید بعد ازخواندن این فصل نوشتن اسکریپتهای ساده را آغاز کنید. اما توضیحاتی که در اینجا آمده است کامل نیست. تمام مطالب این فصل با جزئیات بیشتر در فصل های بعدب مطرح خواهد شد و همچنین سیستم ها از جنبه های گوناگونی از قبیل واسطه های آنها در C که در این فصل به آنها هیچ اشاره ای نشده است در فصل بعدی بحث خواهد شد.

هدف از این فصل نشان دادن ساختارهای Tel , Tk و کارهای مختلفی است که آنها می توانند انجام دهند. بنابراین وقتی در مورد جزئیات خصوصیات آنها بحث می شود قادر خواهید بود ببینید که چرا آنها مفیدند.

۱-۲) شروع به کار: برای فراخوانی یک اسکریپت Tel شما باید یک برنامه کاربردی Tel را اجرا کنید. اگر Tel را روی سیستمتان نصب کرده باید باید یک واسط کاربر ساده Tel به نام Telsh بر روی سیستمتان موجود باشد که شمامی توانید با استفاده از آن بعضی از مثالهای این فصل را امتحان کنید. (اگر Tel روی سیستمتان نصب نشده باشد می توانید با رجوع به ضمیمه اطلاعات کافی راجع به نحوه فراهم کردن و نصب آنها به دست آورید) فرمان زیر را برای فراخوانی

Telsh در واسط های کاربر تایپ کنید. Telsh در حالت محاوره ای راه اندازی خواهد شد. Tel از ورودی استانداردش ورودی ها را می خواند و برای ارزیابی و پردازش به مفسر انتقال می دهد. برای شروع فرمان زیر را در Telsh تایپ کنید. Telsh expr 2+2 نتیجه را که "4" می باشد چاپ می کند. و پرامپت (Prompt) آن برای دریافت فرمان بعدی شما را راهنمایی می کند.

یک سرور از Tel , Tk:

برای مثال چندین ویژگی از Tel را شرح می دهد. اول اینک فرمان های Tel فرمی مشابه فرمان های واسط کاربر دارند. هر فرمان شامل یک یا چند کلمه است که با فضای خالی (Space) یا تب از هم جدا شده اند. در این مثال ٤ کلمه دیده می شود. 2 , 2+ ,2 اولین کلمه هر فرمانی نام آن فرمان است. این نام یک پروسی جر C را در برنامه کاربردی انتخاب می کند. که این پروسی جر تابع مربوط به این فرمان را اجرا خواهد کرد. بقیه کلمات آرگومانهایی هستند که به پروسی جر C انتقال داده می شوند. EXPr یکی از فرامین مرکزی (Core command) ساخته شده در درون مفسر Tel می باشد. بنابراین در هر برنامه‌ی کاربردی Tel وجود دارد. این فرمان آرگومانهایش را به صورت یک رشته به هم اتصال می دهد و این رشته را به صورت یک عبارت ریاضی مورد ارزیابی قرار می دهد.

هر فرمان Tel یک رشته را به عنوان نتیجه بر می گرداند. برای فرمان expr نتیجه مقدار عبارت می باشد. نتایج همیشه به صورت یک رشته برگردانده می شوند. بنابراین expr نتیجه عددی را به رشته تبدیل کرده و آن را بر می گرداند. اگر فرمانی نتیجه ای داشته باشد که معنی خاصی ندارد یک رشته خالی را به جای آن بر می گرداند.

به این مثال توجه کنید.

```
Expr  2 + 2
      ⇒ 4
```

خط اول فرمانی است که شما تایپ کرده ایدو خط دوم نتیجه ای است که این فرمان بر می گرداند. علامت ⇒ نشان دهنده این است که این خط شامل برگردانده شده به عنوان نتیجه می باشد. ولی Telsh واقعاً این علامت "⇒" را چاپ نمی کند. من مقادیر برگشت داده ای که مهم نیستند مانند ترتیبی از فرامین که فقط نتیجه فرمان آخر مهم است را حذف می کنم. فرامین به طور طبیعی در خط جدید خاتمه یابند. بنابراین وقتی شما در Telsh درهر خط چیزی را تایپ می کنید طبیعتاً یک فرمان مجزا مصوب می شود. سمی کالون ها نیز به عنوان جدا کننده فرامین از یکدیگر عمل می کنند و د رمواردی که شما مایلید چند فرمان را در یک خط تایپ کنید مورد استفاده قرار می گیرند. التبه تایپ یک فرمان در چند

خط نیز امکانپذیر است و شما نحوه انجام آن را در آینده خواهید دید. در فرمان expr یک عبارت ساختار دستوری مشابه AMSZC دارد و بیشتر عملگرهای C را با همان قوانین تقدم عملگرها پوشش می دهد. در اینجا چند مثال آورده شده است که شما می توانید آنها را در Telsh تایپ کرده و امتحان کنید.

Expr 3 << 2

⇒ 12

Expr 14.1 *6

⇒ 84.6

Expr (3 > 4) 11 (6 ⇐7)

⇒ 1

این مثال یک عملگر بیتی را شرح می دهد که بیتهای عملوند خود را به سومت چپ شیفت می دهد. مثال دوم نشان می دهد که که این عبارت می توانند شامل مقادیر حقیقی مثل داده های صحیح باشند. مثال آخر استفاده از عملگرهای رابطه ای ⇐ و دو عملگر منطقی ۱۱ را نشان می دهد. مثل C نتایج بولین بهص ورت عددی نشان داده می شود. که ۱ نشانگر مقدار درست و ۰ نشانگر مقدار نادرست می باشند.

برای خارج شدن از Telsh فرمان exit را فراخوانی می کنیم:

Exit

این فرمان برنامه کاربردی را خاتمه داده و به واسطه کاربرد (Shell) بر می گردد.

۲-۲: نوشتن برنامه Helloword با Tk

برنامه کاربردی "Helloword". دکمه "Helloword" با مدیر پنجره mwm تهیه شده است. اگر شما از مدیر پنجره دیگری استفاده کنید ممکن است نمایش شکل شما متفاوت باشد.

گرچه Tel یک مجموعه کامل از ابزارهای برنامه نویسی از قبیل متغیرها و حلقه ها و پروسی جرها را فراهم می کند اما محیط برنامه نویسی مجزایی ندارد.Tel می تواند به عنوان بخشی از برنامه های کاربری که از فرامین Tel خودشان علاوه بر فرمان های مرکزی Tel (Tel eore) استفاده می کنند به کار می روند.

فرمان های خاصی از برنامه های کاربردی اصول جالبی را فراهم می کنند که کنار هم قرار گرفتن این اصول می تواند در توابع مفید باشد. Tel به تنهایی خیلی جالب نیست و استفاده از امکانات آن سخت است مگر اینکه شما Tel را با فرمان های خاصی از برنامه کاربردی به کار برید. Tk یک مجموعه جالب از فرامین را برای استفاده با ابزارهای برنامه نویسی مهیا می کند. بیشتر مثالهای کتاب در یک برنامه کاربردی به نام wish استفاده شده است. که شباهت زیادی به Telsh دارد. به جز اینکه شامل فرمانهای ترعیف شده به وسیله Tk می باشد. فرمان های Tk به شما اجازه می دهد Telsh دارد. به جز اینکه شامل فرمان های تعریف شده به

وسیله Tk می باشد. فرمان های Tk به شما اجازه می دهند که واسط گرافیکی کاربر ایجاد کنید. اگر شما Tk , Tel را روی سیستمتان نصب کرده باشید می توانید wish را نیز مانند Telsh فراخوانی کنید. این کار یک پنجره خالی کوچک روی صفحه نمایش شما ظاهر می کند و فرمان ها را از ورودی استاندارد می خواند. در اینجا یک اسکریپت ساده Wish آمده است.

Button .b- text "Hallo, world! "- command exit pack.b

اگر شما این دو فرمان Tel را در پنجره wish تایپ کنید شکل ۱-۲ در پنجره نمایش داده خواهد شد.

اگر شما با ماوس روی این دکمه کلیک کنید از Wish خارج می شوید. و پنجره دیگر قابل دیدن نخواهد بود. در مورد این مثال چند مطلب وجود دارد که باید توضیح دهیم. ابتدا اجازه دهید موضوعات دستوری را مورد بررسی قرار دهیم. این مثال شاخمل دو فرمان Button و Pack می باشد که هر دو به وسیله Tk پیاده سازی شده اند. گرچه به نظر می رسد که این فرمانها با فرمان expr که در بخش قبلی مطرح شد فرق دارند ولی آنها ساختاری مشابه دیگر فرمان های Tel دارند که شامل یک یا چند کلمه می باشند. که به وسلیه فضاهای خالی از هم مجزا شده اند. فرمان Button شامل ٦ کلمه و فرمان Pack شامل دو کلمه می باشد. کلمه چهارم از فرمان Button درداخل جفت کوتیشن

(“ ”) قرار گرفته است. این عمل اجازه می دهد که در داخل کلمه از کاراکترهای فضای خالی نیز استفاد کنیم. این جفت کوتیشن ها جزئی از خود کلمه نیستند. و قبل از اینکه کلمه به عنوان یک آرگومان به فرمان عبور داده شود توسط مفسر Tel حذف می شوند. برای فرمان expr ساختار کلمه زمانی که Tel ساختار کلمه خیلی مهم است. در فرمان Button آرگومان اول نامن پنجره و آرگومان بعدی که به صورت جفتی می باشد و شامل دو نام است که اولی نام یک گزینه پیکربندی و دومین آرگومان مقداری برای این گزینه است. بنابراین اگر جفت کوتیشن حذف شوند مقدار گزینه Text- عبارت “Hello” خواهد شد و با عبارت “world” بعنوان یک پیکربندی مجزا قرار رفتار خواهد شد. زمانی که هیچ انتخاب ترعیف شده ای به نام “world” وجود ندارد. این فرمان یک مقدار خطا بر می گرداند. حال بیایید رفتار این فرمان ها را بررسی کنیم. Widget یک ساختمان بلاکی عمده برای واسطه گرافیکی کاربر در TK می باشند. یک widget پنجره‌ای با رفتار و نمایشی خاص می باشد. (هر دو واژه “Widget” و “windows” به صورت مترادف در Tk استفاده می شوند. Widgect ها به کلاس هایی مانند منوها و دکمه ها و نوارهایی پیمایش تقسیم می شوند. تمام Widgect ها که دارای کلاس واحدی هستند نمایش و رفتار عمومی یکسانی دارند. برای مثال همه دکمه های Widgect ها که دارای کلاس واحدی هستند نمایش و رفتار عمومی

یکسانی دارند. برای مثال همه دکمه های Widget وقتی که با ماوس فراخوانی شوند یک رشته متنی یا Bitmap را نمایش می دهند و یا یک فرمان مخصوص از Tel سازماندهی شده اند و جایگاه آنها با نامشان در این سلسله مراتب مشخص می شوند. Widget اصلی که وقتی شما wish را آغاز می کنید. روی صفحه نمایش ظاهر می شود با نام "." مشخص می شود. نام b. به فرزند widgect اصلی اشاره دارد. نام widgect ها در Tk درست مثل نام فایل ها در unix می باشد بجز اینکه Tk از "." بعنوان جدا کننده کاراکترها به جای "/" استفاده می کند. بنابراین a.b.e. اشاره به widgect ی دارد که فرزند widgect اصلی است. Tk یک فرمان برای هر کلاس از widgect ها تهیه می کند که شما آن را برای ایجاد یک widgect از همان کلاس فرا می خوانند. برای مثال فرمان button، ویرگت button ایجاد می کند. تمام فرمان های ایجاد widgect فرم مشابهی دارند. آرگومان اول نام widgect جدیدی است که می خواهیم ایجاد کنیم و آرگومانهای بعدی گزینه های پیکربندی را مشخص می کنند. Widgect هایی که با کلاسهای متفاوت مجموعه متفاوتی از گزینه ها را پشتیبانی می کنند. Widgect ها عموماً گزینه های زیادی دارند. برای مثال در اینجا در حدود ۲ گزینه مختلف برای Button ها تعریف شده است و مقادیر پیش فرض برای گزینه هایی که شما تعیین نمی کنید فراهم شده اند. وقتی یک فرمان ایجاد widgect مثال

button فراخوانی می شود یک پنجره جدید به وسیله نام داده شده ایجاد می کند. و با گزینه مشخص شده آن را پیکربندی می کند. فرمان button در مثال دو گزینه را مشخص می کند. text– که یک رشته برای نمایش button است و – command که یک اسکریپت Tel است که وقتی کاربر آن را فراخوانی کند اجرا می شود. در این مثال گزینه command–، exit است. در اینجا چند مثال دیگر از گزینه های Button آورده شده است. که می توانید آنها را امتحان کنید.

- Back ground The background color for the button
- Foreground the color of the text in the button
- Font the name of the hont touse for the button?

Such as * -times – medium- r- normal-- * -?

120-* for a 12- point tiMES Roman font?

فرمان Pack دکمه Widget را روی صفحه نمایش قابل رویت می کند .ایجاد یک Widget به صورت خودکار باعث نمایش آن نمی شود. عناصر مستقلی که مدیران هندسی Cgeometrymoridgen نامیده می شوند. مسمولیت محاسبه اندازه و موقعیت ونمایش این پنجره ها (Widget) را بر روی صفحه نمایش بر عهده دارند این فرمان درخواست می کند که b. تمام قسمت های پنجره والد خود را پر کند. بعلاوه اگر والد (در اینجا پنجره) فضایی بیشتر از فضای مورد نیاز فرزند

در اختیار داشته باشد جمع می شود تا فقط به اندازه کافی برای نگهداشتن فرزند بزرگ شود. بنابراین وقتی شما فرمان Pack را تایپ می کنید پنجره اصلی از اندازه اصلی خودش جمع تر می شود. و اندازه آن برابر اندازه نشان داده شده در شکل ۱-۲ می شود.

۳-۲- فایهلای اسکرپیتی:

در مثالهایی که تاکنون دیده اید فرمان های Tel به صورت محاوره ای در Telsh و Wish تایپ می شدند. شما می توانید این فرمان ها را در درون فایل های اسکرپیتی به کار برده و آنها را فراخوانی کنید. درست مثل Shellscript (زیر برنامه ای که توسط مفسر فرمان (Shell) یک سیستم عامل کامل اجرا می شود. برای انجام انی مثال یعنی Hello word متن زیر را در یک فایل به نام hello قرار دهید.

```
Hello:
# ! usr / bin/ wish- f
Button. B- text “Hello, World! “- Command exit
Paek. B
```

این مثال به جز خط اول دقیقاً مثل مثالی است که پیشتر نوشتیم. تا وقتی این مثال را Wish می نوشتیم خط اول یک توضیح بود ولی اگر بخواهید این فایل را اجرایی کنید (برای مثال “chanod 775 hello” را در واسط کاربرد (shell)

تایپ کنید). شما می توانید سیستم Wish را فرا م یخواند و این فایل را مثال یـک اسکریپت تفسیر انتقال می دهد. Wish همان پنجره نشان داده شـده در Figure 2.1 نمایش می دهد و منتظر فرمان شـما خواهـد بـود. در ایـن مـورد شـما قـادر نخواهید بود فرمان ها را به طور محاوره ای در Wish تایپ کنید. تنها کای که می توانید انجام دهید کلیک کردن روی این دکمه است. این اسکریپت تنها وقتی شـما Wish را در میسر /usr/ local/ bin نصب کرده باید کـار مـی کنـد. اگـر شـما wish را در جای دیگری نصب کرده اید باید خط اول را طوری تغییر دهید تا بـه محل مورد نظر شما که در آنجا wish نصب شده رجوع کنیـد. عمـلاص کـاربران برنامه های کاربردی Tk بندرت فرمان هـای Tel را بـه کـار مـی برنـد. کـاربران معمولا توسط ماوس وصفی کلید با برنامه کاربردی در تعاملنـد مثـل برنامـه هـای گرافیکی. Tel معمولا پشت صحنه کارهایی انجام می دهد که از دید کاربران پنهان است. اسکریپت hello با یک برنامه کـاربردی کـه کـدهای C و ابـزاری از قبیـل Motif نوشته و کامپایل شده و به یک فایل باینری اجرایـی تبـدیل شـده رفتـاری مشابه دارد. باری دیباگ کردن همانگونه که معمول است گسترش دهندگان برنامـه های کاربردی فرامین Tel را بهصورت محاوره ای تایپ می کنند. برای مثال شـما می توانید اسکریپت Hello را به وسیله Wish تست کنید. (بـه جـای hello، در واسط کاربر wish را تایپ کنید) سپس فرمان زیر را تایپ کنید.

Sourree hello

Source یک فرمان Tel است که نام فایل را به عنوان یک آرگومان دریافت می کند. فایل را می خواند و به عنوان یک اسکریپت Tel مورد ارزیابی قرار می دهد. این کار همان واسط کاربردی را تولید خواهد کرد که اگر شما Hello را مستقیماً از واسط کاربر Shell فراخوانی می گردید. اما شما حالا می توانید فرمان های Tel را به صورت محاوره ای نیز به کار برید. شم می توانید فایل اسکریپتی را ویرایش کرده و فرمان command– را به command "Puts Good- bye! –exit" ; تغییر دهید. سپس بدون اینکه برنامه را مجدداً شروع کنید فرمان های زیر را در wish تایپ کنید:

Destory. B

Source Hello

فرمان اول دکمه موجود را وزن خواهد کرد و فرمان دوم مجدداً دکمه را با گزینه جدید ایجاد خواهد کرد. حالا وقتی شا روی این دکمه کلیک کنید فرمان Puts قبل از خروج از Wish یک پیغام جدید در خروجی استاندارد خواهد نوشت.

٤-٢- متغیرها و جانشینی ها:

Tel به شما اجازه می دهد مقادیری را در متغیرها ذخیره کرده و سپس این مقـادیر را در فرمان ها استفاده کنید. برای مثال اسکریپت زیر را ملاحظه کنید که می توانید آن را در wish یا Telsh تایپ کنید.

Set a 44 expr $a* 4

⇒ 44 ⇒176

فرمان اول مقدار 44 را به متغیر a نسبت می دهد. و مقدار آن را بر می گردانـد. در فرماندوم علامت $ باعث می شود که Tel عمـل جانشـینی متغیـر را انجـام دهـد.

مفسـر

Tel Q $a مقدار متغیر a را که توسـط فرمـان seqبـه آن نسـبت داده شـده و مقدار آن 44 است قرا رمی دهد. پس آرگومان expr مقدار 4*44 را دریافت مـی کند. در Tel لازم نیست متغیرها را تعریف کنیم وقتی مقداری به آنهـا نسـبت مـی دهیم به طور خودکار ایجاد می وشوند. مقادیر متغیرها به صورت رشته ذخیره مـی شوند. و این مقادیر رشته ای با هر طوری مجازند. البته در این مثال اگـر در فرمـان expr، a مقداری به جز مقادیر عددی صحیح یا عددی حقیقـی داشـته باشـد یـک خطا رخ خواهد داد. (مقادیر دیگر را امتحان کنید و نتیجه آن را ببنید).

Tel همچنین جانشینی فرمان ها را نیز پیش بینی کرده است. که به شما اجازه مـی دهد از نتیجه یک فرمان بعنوان آرگومان فمرانی دیگر استفاده کنید.

```
Set   a
        447
Set b [expr $a* 44]
⇒ 176
```

براکتها جانشینی فرمان ها را فراخوانی می کنند. هر چیزی که در درون براکتها فقرار گرفته باشد بعنوان یک اسکریپت Tel مجزا ارزیابی شده و نتیجه این اسکریپت جانشین فرمانی می شود که در بارکت قرار گرفته است. در این مثال دومین آرگومان از دومین فرمان "176" خواهد شد.

۵-۲) ساختار کنترلی:

مثال بعدی متغیرها و جانشین های آنها را با ساختارهای کنترلی برای ساختن یک پروسی جر Tcl به نام Pouler به کار می برد که یک عدد پایه را به یک عدد صحیح می رساند.

```
Proc power {base P}
Set result 1
While {$P>0}
Set result {expr $result * $ base}
Set P [expr $ P-1]
{
return $result
}
```

اگر شما خطوط بالا را در wish , Telsh به کار ببرید یا آنها را در یک فایل قرار داده و با فرمان Source به کار ببرید فرمان جدید Power در دسترس خواهد. این فرمان در آرگومان وارد یک عدد صحیح و یک عدد پایه که باید به توان یان عدد صحیح برسد ونتیجه یم شود عدد پایه که به توآن عدد صحیح رسیده است:

Power 2 6

$\Rightarrow 6'4$

Power 1.15 5

$\Rightarrow 2.01136$

این مثال از یک ساختار دستوری دیگر به نام آکولاد b { } ϕ استفاده می کند. آکولادها مثل جفت کوتیشن می تواند در اطرف یک کلمه و فضاهای خالی قرار گیرند. اما آکولاد ها و جفت کوتیشن ها از دو لحاظ با هم تفاوت دارند. ابتدا اینکه آکولاد ها به صورت لانه ای هستند. این فرمان از وقتی در خط اول آکولاد باز میشود شروع شده و وقتی در خط وسط بسته می شود پایان می یابد و شامل مطالبی است که در بین این دو آکولاد قرار می گیرند.

مفسر Tel آکولادهایی بیرونی را حذف می کند و هر چیزی را که بین آنها باشد به انضمام چندیدن صفت از آکولادهای تو در تو را به عنوان آرگومان به پروسی جر (Proe) عبور میدهد. دومیت تفاوت این جفت کوتیشن ها و آکولادها این است که در داخل آکولادها جانشینی اتفاق نمی افتد ولی در داخل جفت کوتیشن ها

عملجانشینی انجام می شود. تمام کاراکترهای بین دو آکولاد کلمه به کلمه و بدون هیچ پردازش خاصی به پروسی انتقال داده می شوند. فرمان Proe سه آرگومان دارد. نام پروسی جر، یک لیست از نام آرگومانها که با فضاهای خالی از هم جدا شده اند و بدنه پروسی جر که یک اسکریپت Tel می باشد. Proe نام پروسی جر با به عنوان یک فرمان جدید وارد مفسر Tel می کند. هر وقت این فرمان فراخوانی شود بدنه تابع مورد ارزیابی قرار خواهد گرفت. وقتی که بدنه تابع در حال اجراست دستیابی به آرگومانها بعنوان متغیرها امکان پذیر خواهد بود که Base آرگومان اول و P دومین ارگومان را برای Power نگه می دارند. بدنه پروسی جر Power سه فرمان را شامل می شود. set , while , return. فرمان While بیشترین کار پروسی جر را انجام می دد و شالم دو آرگومان می باشد. یک عبارت مثل "$P>0" و یکی بدنه که یک اسکریپت چند خطی Tel می باشد. فرمان While این عبارت را مورد ارزیابی قرار می دهد و اگر نتیجه غیر صفر باشد بدنه به عنوان یک اسکریپت Tel مورد ارزیابی قرار می گیرد. و این فرایند بارها و بارها تکرار می شود. تا اینکه این عبارت صفر شود. در این مثال بدنه نرمان While مقدار نتیجه را در Base ضرب می کند و p را یک واحد کاهش می دهد. وقتی که p صفر شود نتیجه مقدار موردنظر یعنی Base به توان P می باشد. فرمان return حذف می شود مقدار برگشتی پروسی جر نتیجه آخرین

فرمان در بدنه پروسی جر خواهد شد. در فرمان Power نتیجه فرمان While مقدار برگشت داده خواهد وبد. که همیشه یک رشته تهی خواهد بود. ما از آکولادها در این مثال استفاده کردیم. مسأله مشکلتر در نوشتن اسکرپیتهای Tel مدیریت جانشینها می باشد. اینکه وقتی شما می خواهید روی دهند و وقتی که نمی خواهید از رخ دادن آنها جلوگیری کنید. آکولادها مانع از جانشینی ها می شوند و یا آنها را به تعویق می اندازند. بدنه پروسی جر باید در درون آکولادها قرار گیرند. چون ما نمی خواهیم زمانی که بدنه تابع بعنوان یک آرگومان به Proe انتقال می یابد. جانشینی فرمان ها و متغیرها اتفاق بیفتد. ما می خواهیم این جانشینی در زمانی دیگر یعنی زمانی که بدنه تابع بعنوان یک اسکرپیت Tel مورد ارزیابی قرار می گیرد اتفاق بیفتد. بدنه فرمان While نیز به همین دلیل در این آکولادها قرار گرفته است. بجز اینکه نمی خواهیم جانشینی ها یک بار انجام شود، می خواهیم در زمان تجزیه و تحلیل فرمان while تا زمانی که بدنه آن مورد ارزیابی قرار می گیرد این جانشینی ها با رها و بارها انجام می شود. آکولادها در مورد آرگومان {$P>0} مربوط به فرمان While نیز مورد نیازند. بدون این آکولادها مقدار متغیر P در هنگام تحلیل فرمان While جانشین متغیر P خواهد شد و این عبارت یک مقدار ثابت خواهد داشت. و حلقه while تا ابد ادامه پیدا خواهد کرد. (شما می توانید بعضی از این آرگومانها را با جفت کوتیشن جایگزین

کنید و ببینید چه اتفاق می افتد). در مثالهای این کتاب در جایی کـه آکـولاد بـرای یک آرگومان که یک اسکریپت Tel می باشد باز مـی شـود و مـا از یـک سـاختار دستوری خاص استفاده می کنیم و اسکریپت را که در خطوط بعدی مـی آینـد بـه صورت دندانه دار می نویسیم. ۱) گرچه من فکر مـی کـنم ایـن کـار و آکـولاد در انتهای اسکریپت بسته می شود. خوانایی برنامه را بالا مـی بـرد ولـی Tel بـه ایـن ساختار دستوری خاص نیازی ندارد. این قوانی دستوری آرگومانهـای اسـکریپتی و هر آرگومان دیگری به یک شکل رفتار می کنند. درحقیقت مفسر Tel حتی وقتـی که یک آرگون را تجزیه و تحلیل می کند نمی داند که این آرگومان یک اسـکریپت حقیقت مفسر Tel حتی وقتی یک آرگومان را تجزیه و تحلیل می کنـد نمـی دانـد که این آرگومان یک اسکریپت است. نتیجه مطلب ایـن اسـت کـه آکـولاد بایـد در همان خطی که قسمت قبلی فرمان قرار گرتفه است باز شود. اگر آکولاد باز به یک خط دیگر انتقال یابد خط جدید خود به خود قبل از آکولاد باز به این فرمان خاتمه می دهد. شما تاکنون تقریباً با تمام ساختار دستوری زبـان Tel را دیـده ایـد. تنهـا ویژگی دستوری که باقیمانده جانشینی یک اسلش "/" می باشد که به شـما اجـازه می دهد کاراکترهای خاص مثل دالرساین "$" ار در داخل یک کلمه به کار ببریـد. بدون اینکه لازم باشد آن را درون آکولاد قرار دهید. توجه داشته باشید که While proe , عناصر نحوی خاصی در Tel نیستند. این دو فرمان مانند بقیه فرمان هـای

Tel می باشند. و دارای آرگومانهایی هستند. تنها مورد خاص در مورد , While proe این است که آنها با بعضی آرگومانهایشان مثل اسکریپتهای Tel رفتار می کنند و باعث می شوند این اسکریپتها مورد ارزیابی قرار گیرند. خیلی از فرمانهای دیگر نیز این کار را انجام می دهند. فرمان Button مثالی در این مورد بود. (آرگومان فرمان یک اسکریپت Tel می باشد) و شما در این مورد ساختارهای کنترلی دیگری مثل for case, foreach, Eual را نیز موردمطالعه قرار خواهید داد. نکته دیگری در مورد پروسی جرها این است که متغیرهای یک پروسی جر به طور طبیعی برای آن پروسی جر محلی اند و در خارج از آن پروسی جر قابل مشاده نیستد. در مثال Base,P ,Power متغیرهایی محلی آرگومان بودند و همچنین متغیر Result. با هر بار فراخوانی پروسی جر یک مجموعه تاز از متغیرهای محلی ایجاد می شود. (یک کپی از مقدار آرگومانها انتقال می یابد) و وقتی یک پروسی جر باز می گردد متغیرهای محلی آن حذف می شوند.

Figure 2.2) یک واسط گرافیکی کاربری عدد پایه را به توان می رساند:

متغیرهایی که در خارج از پروسی جر ها تعریف می شوند متغیرهای عمومی (Global) نامیده می شوند. آنها برای همیشه باقی می مانند مگر اینکه به صورت آشکار حذف شوند. شما بعداً در می یابید که چطور یک پروسی جر می تواند به متغیرهای عمومی و متغسرهای محلی پروسی جرهای فعال دیگر دسترسی داشته باشد.

٦-٢) قید کردن رویداد:

مثال بعدی یک رابط گرافیکی برای پروسی جر Power تهیه می کند. علاوه بر شرح دادن و اثبات دو کلاس جدید از Widget ها مکانیسم انقیاد Tk

(Binding) را نیز توضیح می دهد. یک انقاید binding باعث می شود که یک اسکریپت Tel در هر جایی که یک رویداد خاص در یک پنجره خاص رخ دهد مورد ارزیابی قرار گیرد. گزینه command– برای دکمه ها یک مثال ساده از پیاده سازی بایندینگ با یک کلاس خاص از widget می باشد. Tk همچنین مکانیسم های عمومی بیشتری را شامل می شود که می تواند این رفتار را با ویدگت های اختیاری و راه های دلخواه بسط دهد. برای اجرای این مثال اسکریپت زیر را به درون یک فایل power کپی کرده و این فایل را از واسط کاربر shell فراخوانی کنید.

این اسکریپت یک صفحه نمایش مثل Figure تولید می کند. در اینجا دو ویرگت widget ورودی وجود دارد که می توانید با ماوس روی آن کلیک کنید و اعداد را تایپ کنید. اگر return را در هر یک از ورودی ها تایپ کنید نتیجه در سمت راست پنجره نمایش داده خواهد شد. شما می توانید با تغییر در هر کدام از ورودی های Basse یا P نتایج جدیدی را محاسبه کنید و با تایپ مجدد کلمه return نتیجه آن را ببینید. این برنامه کاربردی شامل ۵ (شی گرافیکی) می باشد که عبارتند از دو ورودی و سه برجسب. ورودی ها ویدگت های یهستند که یک رشته متنی را نمایش می دهند که شما می توانید به صورت محاوره ای آنها را ویرایش کنید. ورودی های -base , Power برای ورودی اعداد استفاده می شوند. هر ورودی

عرضی برای 6 واحد دارد و بدین معنی است که می تواند اعداد را تا ٦ رقم نمایش دهد ورودی های را به صورت تورفتگی نمایش می دهد. گزینه – textuarible برای هر ورودی یک متغیر عمومی را برای نگهداری متن ورودی مشخص می کند و هر تغییری که شما در ورودی اعمال می کنید در این متغیر منعکس خواهد شد. دو برچسب Lable1 , Lable2. دو متن نمایشی را نگه می دارند و برچسب سوم یعنی result. نشان دهنده محاسبه نتیجه توان می باشد گزینه textuariable– باعث می شود که Result. هر رشته ای را که در متغیر عمومی Result وجود دارد نمایش دهد در صورتی که Lable1 , Lable2. رشته های ثابتی را نمایش می دهند. فرمان Pack هر پنج ویدگت را در یک ردیف از چپ به راست مرتب می کند. این فرمان دو خط را در اسکریپت اشغال کرده است و یک اسلشی که در انتهای خط اول انی فرمان قرار گرفته است یک کاراکتر خطی دنباله دار است که باعث می شود خط جدید به عنوان فضای خالی تلقی شود. گزینه Side– باعث می شود که هر «ویدگت» در سمت چپ فضای خالی باقمیانده در ویدگت اصلی قرار گیرد: ابتدا base. در لبه سمت چپ پنجره اصلی قرار می گیرد سپس lable1. در سمت چپ فضایی که توسط base. اشغال نشده است قرار می گیرد. و به همین ترتیب ادامه می یابد. گزینه های – padx , pady با قرار گرفتن ۱ میلی متر فضای اضافی در سمت چپ و راست

هر ویدگت و ۲ میلی متر فضای اضافی در بالا و پایین نمایش آن را کمی جالبتر و بهتر می کنند.

پسوند "m" میلی متر را شمخص می کند. شما همچنین می توانید "C" را برای سانتی متر "?" را برای اینچ و "P" را برای پوینت به کار برید و برای پیکسل هیچ پسنودی لازم نیست. فرمان brind واسط کاربر را به پروسی جر power متصل می کند. هر فرمان brind سه ارگومان دارد: نام یک پنجره، یک رویداد ویژه و یک اسکریپت Tel که وقتی آن رویداد در پنجره اتفاق بیفتد فراخوانی می شود.

<Reture> رویدادی را مشخص می کند که با تایپ یک کلید برگشتی (return) در صفحه کلید توسط کاربر اتفاق می افتد. در اینجا تعداد دیگری از رویدادها مشخ شده اند که ممکن است مفید واقع شوند:

????

این اسکریپتها برای انتیاد (bindings) Poewer را فراخوانی می کنند. و مقادیر دو ووردی را به آن انتقال می دهند. و نتیجه را در Result ذخیره می کنند. بنابراین این نتیجه در ویدگت result. نمایش داده خواهد شد. این انتیادها (Brindgs) و پیوندها رفتار عمومی تعبیه شده ورودی ها (ویرایش رشته های

متنی) با رفتار برنامه های کاربردی خاص (محاسبه یک مقدار براساس دو ورودی و نمایش این مقدار در ویدگت سوم) را توسعه می دهد.

انتیاد (binding) اطلاعاتی را راجع به رویدادها در اختیار ما می گذارد مثلا اینکه وقتی رویداری اتفاق می افتد پوپنتر در چه محلی قرار می گیرد به عنوان مثال wish را راه اندازی کنید. فرمان زیر را تایپ کنید:

bind. <Any- Motion> {puty “pointer at %x , %y”}

حالا مکان نما به بالای پنجره حرکت می کند. هر زمانی که مکان نما حرکت می کند یک پیغام در خروجی استاندارد نوشته می شود که محل جدید مکان نما را نشان می دهد. وقتی که مکان نما حرت می کند یک رویداد رخ می دهد، TK علامت ٪ و نوشته هایی را که به دنبال آن می آید را اسکن کرده و قبل از آنکه آنها را به Tel منتقل کنند و این اسکریپتها مورد ارزیابی قرار گیرند آنها را با اطلاعاتی راجع به آن رویداد جایگزین می کنند. %x با موقعیت مکان نما در محور عرض ها جایگزین می شود.

۷-۲- زیر پروسه ها:

معمولا Tel هر فرمانی را با فراخوانی یک پروسی جر C در یک برنامه کاربردی و انجام یک تابع اجرا می کند. این اجرا با یک برنامه shen مثل sh تفاوت دارد و برنامه های Shell به طور معمول در کی زیر فرایند اجرا می شوند. Tel هم به

شما اجازه می دهد زیر فرایندهایی را ایجاد کنید و از فرمان exee استفاده کنید.

در اینجا مثال ساده ای از Tenel آورده ایم:

exel grep # in elude Tk,h

⇒ # include <Tel.h>

include <X11/ XX lib.h>

include <Stddek.h>

فرمان exec با آرگومانهایش رفتاری مشابه کلمات خط فرمان شل Shell دارد. در این مثال exec یک فرایند جدید برای اجرای برنامه Grep ایجاد می کند و "#inelude" , "tk.h" را به عنوان آرگومانهایش انتقال دهد. به شرط اینکه خط زیر را در Shell تایپ کرده باشدی.

Grep # inckude tk.h

برنامه grep فایل tk.h را در خطوطی که شامل رشته # include مس یاد جشتجو کزده و این خطوط را در خروجی استاندارد می نویسد. اما exec خروجی استاندارد را به Tel بر می گرداند. Exec همچنین از آدرس دهی ورودی و خروجی با استفاده از علائم استاندارد در اسکریپتهای Tel (Shell) از قبیل >> < , و <، پردازش موازی با "1" و فرایندهای پس زمینه (Back ground) با استفاده از & حمایت می کند. مثال زیر یک واسط کاربر ساده برای ذخیره سازی

و فراخوانی مجدد با استفاده از فرامین واسط کاربر (shell) ایجاد می کند.

اسکریپت زیر را در یک فایل به نام redo تایپ کرده و آن را فراخوانی کنید:

شکل

figure 2.3. برنامه کاربردی Redo. کاربر می تواند یک فرمان در پنجره تایپ کند. مثل شکل (a) وقتی که کاربر retum را تایپ می کند. این فرمان بعنوان یک زیر برنامه که exee استفاده می کند فراخوانی می کند فراخوانی شده و یک دکمه جدید ایجاد می شود. که می تواند بعداً برای فراخوانی مجدد فرمان مورد استفاده قرار گیرد. (b) فرمان های دیگری که می توانند برای ایجاد دکمه های بیشتر استفاده شوند از بالا پنج فرمان دیگر می باشند.

```
# ! /usr/ local/ bin/ wish – F
set id o
entry. Entry- width 30- relief sunken- textuariable cmd.
Pack. Entry- Padx im- Pady 1m
```

Bind. Entry <Return> {

Set id [expr $ id + 1]

If {& id >5} {

Destory. B [expr $ id- 5]

Button. B$ id- command “exee < @ stdin> @ stdout $ emd –

text $emd

- Pack. B $ id- fill x
- Entry delete o end

در ابتدا این اسکریت یک واسط کاربر با یک ویدگت تک ورودی ایجاد مـی کنـد. شما می توانید یک فرمان shell مثـل LS را در ورودی تایـپ کنیـد. وقتـی شـما retum را می زنید Redo فراخوانی شده و فرمنانی را کـه در shell نمـایش داده می شود بعلاوه این اسکریپت یک دکمه ویدگت جدید ایجاد مـی کنـد کـه فرمـان رانمایش می دهد. ((see Figore 2.3) و شما می توانید با کلیک کردن روی این دکمه فرمان را مجدداً فراخوانی کنید. هر چه شما فرمان های بیشتری را تایپ کنید و دکمه های بیشتری نمایان می شوند. از بـالا پـنج فرمـان را در Figure 2.3 بـه خاطر بسپارید.

جالبترین قسمت اسکریپت redo در فرمان bind مـی باشـد. انتیـاد <Return> باید فرمانی را که در متغیر cmd ذخیره شده اجرا کند و یک دکمه ویدگت جدیـد ایجاد کند. ابتدا یک ویدگت ایجاد می کند. دکمه های ویدگت نامهایی مثل , b2. b1. دارند و اعداد در متغیر 2d قرا رمی گیرند. Id از صفر شروع می شـود و قبـل ازایجاد هر دکمه جدید افزایش می یابد. "b$jd." یـک ویـدگت بـه نـام "b." و مقدار jd تولید می کند. قبل از ایجاد ویدگت جدید اسکریپت چک مـی کنـد. کـه آیا هر پنج فرمان در حال حاضر ذخیره شده اند که اگر پاسـخ مثبـت بـود قـدیمی ترین دکمه موجود حذف می شود.

”b [expr $ jd-5]“ نام آخرین دکمه را با کم کردن عدد ۵ از تعداد دکمه های جدید والحاق آن با ”b.“ تولید می کند. فرمان تولید دکمه جدید exee را فراخوانی کرده و از ورودی و خروجی استاندارد برای زیر پروسه ها به ورودی و خروجی استاندارد wish تغییر مسیر می دهد. به همان ترتیبی که wish از واسط کاربرد (shell) فراخوانی شده بود، و این باعث می شود که خروجی زیر پروسه ها به جای اینکه به wish برگردند در پنجره واسط گر (Shell) بر نمایش داده داده شوند.

فرمان "Pack. B$jd- fill X“ دکمه ای جدید می سازد که در انتهای پنجره نمایش داده می شود. ”fillX-“ با کشش دکمه به طور افق نمایش

آن را بهتر می کند. تا فضای کافی برای متن آن دکمه وجود داشته باشد سعی کنید fill– را حذف کنید تا ببینید چه اتفاقی بدون آن رخ می دهد. دو فرمان اخیر در مورد این کار (birding) اسکریپت widget command نامیده می شوند. هر وقت که یک ویدگت جدید ایجاد می شود یک فرمان Tcl جدید نیز با همانت نام بعنوان ویدگت ایجاد می شود و شما می توانید این فرمان را برای ارتباط با ویدگت فراخوانی کنید. اولین آرگومان یک فرمان ویدگت یکی از چندین عملگر را انتخاب می کند و آرگومانهای دیگر به عنوان پارامتری برای این عملگر استفاده می شوند. در اسکریپت Redo اولین فرمان ویدگت باعث می شود دکمه ویدگت

اگر شما با ماوس روی آن کلیک کرده باشید فرمانش را فراخوانی می کند. فرمان ویدگت دوم ورودی ویدگت را پاک می کند تا فرمان جدید تایپ شود.

هر کلاس از ویدگت در ویدگت فرمانهایش مجموعهع متناوتی از عملگرها را پشتیبانی می کند. اما خیلی از عملگرها از یک ویدگت دیگر مشابه اند. باری مثال هر ویدگتی از یک پیکربندی ویدگت فرمان پشتیبانی می کند که می تواند برای اصلاح هر پیکربندی از ویدگت به کار گرفته شود. اگر شما بخواهید به صورت محاوره ای اسکریپت Redo را اجرا کنید می توانید خط فرمان زیر را تایپ کرده و رنگ زمینه تصویر را به رنگ زرد تغییر دهید.

- entry configure- back ground yello

یا می توانید بنویسید:

.b1 configure – foreground brown

.b1 flash

برای تغییر رنگ متن در دکمه b1. به قهوه ای و سپس نورانی شدن دکمه می توانید خطوط بالا را تایپ کنید. یکی از مهمترین چیزها درباره Tel , Tk این است که آنها از هر لحاظ یک برنامه کاربردی در هنگام اجرا قابل دسترسی و قابل تغییر می کنند. برای مثال اسکریپت Redo واسط خودش را اصلاح می کند. بعلاوه Tk فرمان هایی را فراهم می کند که شما می توانید برای پرس و جوی

ساختار سلسله مراتبی ویدگت استفاده کنید. و شما می توانید از پیکربندی فرمان های ویدگت برای پرس و جو و اصلاح پیکربندی ویدگتهای مجزا استفاده کنید.

۸-۲) ویژگی های دیگری از Tk , Tcl

مثالهای این فصل از هر جنبه ای دستور زبان Tel را بررسی می کند و خیلی از مهمترین ویژگی های Tk , Tcl را شرح می دهد. Tk , Tcl شامل خیلی از ابزارهای دیگری هستند که در این فصل مورد استفاده قرار نمی گیرند. همه اینها بعداً در همین کتاب بررسی می شوند. ویژگی های مفید تری نیز وجود دارند که هنوز ذکر نشده اند:

آرایه ها و لیست ها: Tel آرایه هایی برای ذخیره سازی کلیدها و مقادیرشان به صورت کالا فراهم می کند و لیستها را برای مدیریت انبوهی از داده ها در اختیار قرار می دهد.

ساختار های کنترلی: Tcl فرمانهای دیگری باری کنترل جویان اجرا تهیه کرده است. مثل switch, foreach , for , eval.

دستکاری رشته ها: Tel فرمان هایی برای دستکاری رشته ها شامل می شود از قبیل اندازه گیری طول رشته ها و انجام دادن الگوی تطابق برای جانشینی عبارات منظم.

دسترسی به فایل: شما می توانید در درون اسکریپتها از فایل ها بخوانید و در فایل ها بنویسید و اصلاح خصوصیاتی از قبیل طول و زمان اجرا را انجام دهید.

ویدگتهای بیشتر: Tk کلاسهای زیادی از ویدگتها را شامل می شود. بعلاوه اینها نمایش داده شده اند. مثل، منوها، نوارهای پیمایشی و ویدگتهای ترسیمی که canvas نامیده می شود و یک ویدگت متنی که اجرای جلوه های عبارات فوق متن را آسانتر می کند.

دسترسی به امکانات دیگر X: Tk فرمانهایی برای دستیابی به همه امکانات سیستم پنجره X از قبیل فرمانهایی برای ارتباط با مدیر پنجره (بعنوان مثال برای تنظیم عنوان پنجره)، و زمانی برای اصلاح انتخاب و فرمانی برای مدیریت فوکوس تهیه می کند.

واسطهای C: Tcl کتابخانه ای از پروسی جرهای e تهیه می کند که مشا می توانید برای تعریف فرمانهای Tel در C استفاده کنید. TK کتابخانه ای تهیه می کند که شما برای ایجاد کلاسهای ویدگت و مدیریت هندسی در C مورد استفاده قرار دهید:

۹-۲) بسط و توسعه برنامه های کاربردی

Tcl و Tk رشدی فعال و سریع دارند. خیلی از افراد برنامه های کاربردی براساس Tcl و Tk ساخته اند. و بسته های نرم افزاری که براساس توابع Tcl و Tk گسترش یافته‌اند. چندین برنامه کاربردی و بسته نرم افزاری در دسترس اند و در بسیاری از جاها در Tcl و Tk استفاده می شوند. در این کتاب مجال بحث بر روی همه نرم افزارهای موجود در Tcl/Tk با جزئیات وجود ندارد. اما این بخش مروری سریع بر روی ۵ تا از متداولترین برنامه های کاربردی و گسترش های آن دارد. ضمیمه A برای به دست آوردن اطلاعاتی راجع به نرم افزارهای Tcl و Tk بینید.

۱-۹-۲) expect

expect یکی از قدیمی ترین برنامه های کاربردی و همچنین یکی از متدوالترین آنها می باشد. این برنامه ای است که بهص ورت محاوره ای با برنامه ها در ارتباط است. در اسکریپت زیر، expect می داند. چه خروجی می تواند از یک برنامه مورد انتظار باشدو چه پاسخی صحیح است می تواند برای کترل اتوماتیک برنامه هایی مانند: ftp, Telnet, rlogin, crypt, fsek, tip که می توانند بطور خودکار از یک اسکریپت شل (shell) هدایت شوند. چون نیاز دارند با ورودی در محاوره ای باشند استفاده شود. Expect همچنین اجازه یم دهد کاربران به طور

مستقیم کنترل برنامه را به دست گیرند. هر زمانی کـه بخواهنـد آن محـاوره کننـد.

بعنوان مثال اسکریپت expect زیر به وسیله برنماه rlogin با یک ماشـین راه دور مرتبط می شود. و کار را با ماشین منبع هماهنگ می کند. سپس کنترل را در اختیار کاربر قرار می دهد:

```
# !/ usr/ loeal/ bin/ expect
Spawn rtogin [Lindex $aragv  1]
Expext- re “(% | #)”
Send “cd [pwd]\r”
interact
```

فرمان های spawn, expect, send, inteteract با expect پیاده سازی شده اند و Lindex , pwd فرمان های توکـار هسـتند. فرمـان rlogin spwan را بـا استفاده از یک خط فرمان و آرگوکمانی بعنوان یک ماشین راه دور راه انـدازی مـی کنند. فرمان expect منتظر می ماند که خروجی rlogin یک prompt ایجاد کند (% یا # که به دنبال آن یک فضای خالی قرار مـی گیـرد) سـپس خروجـی هـای send یک فرمان برای تغییر دایرکتوری کاری می باشد فقط در صورتی کـه کـاربر فرمان را به صورت محاوره ای تایپ کرده باشد.

سرانجام interact باعث می شود expect مسیر را به بسرعت طی کرده و کاربری که اسکریپت expect را فراخوانی کرده بتواند به طور مستقیم با rlogin صحبت کند. Expext می تواند برای خیلی از اهداف از قبیل خطا یابها و برنامه هایی که برای پست الکترونیکی به کار می روند. و برنامه های دیگری که زبان اهی اسکپریتی برای خودشان ندارند استفاده شوند. این برنامه ها نیازی به تغییرات هدایت شده به وسیلهexpect ندارند expect همچنین برای امتحان برنامه های محاوره ای مناسب است. Expect می تواند با Tk یا گسترش های دیگری از Tel ترکیب شود. برای مثال با استفاده از Tk ساختن یک رابط گرافیکی برای برنامه محاوره ای موجود بدون تغییر آن برنامه کاربردی امکان پذیر است.

Expect به وسیله دان لیبز (Don Libes) ایجاد شده است.

Tel توسعه یافته:

Tel توسعه یافته (TCLX) یک بسته نرم افزاری کتابخانه ای می باشد که فرمان های توکار Tel با خیلی از فرمان ها و پروسی جر های دیگر در سیتم های برنامه نویسی به آن افزوده شده است که می تواند با هر برنامه کاربردی Tel مورد استفاده قرار گیرد. در اینجا چند تا از متداولترین خصوصیات TCLX آورده شده است:

- دستیابی به سیستم های POSIX
- امکانات خواندن یک فایل به صورت عملیاتی شبیه برنامه awk.
- فرمان هایی برای دستکاری زمان و تاریخ ها و تغییر آنها به کداسکی و بالعکس.
- امکانات help به صورت برخط (online)
- امکاناتی برای خطایابی و تجزیه و تحلیل فایل و گسترش برنامه ها.

خیلی از بهترین ویژگی های TCLX قسمتی از TCLX نیستند. ثابت خواهد شد که انها در جاهایی سودمندند که با هسته Tcl ترکیب شده باشند. در بین خصوصیات فراهم شده توسط TCLX که به Tcl اضافه شده فایل های ورودی و خروجی آرایه متغیرها، محاسبات حقیقتی و توابع مافوق، فراخوانی خودکار و فرمان UPVAR , XPEG بین المللی.

Tcl توسعه یافته به وسیله کارل لنبارو Ckorl lenbaro و مارک دیکنز (Diekhens mark) ساخته شده بود.

ساختن و ایحاد واسطهای کاربر گرافیکی را با نوشتن اسکریپتهای Tcl ساده می کند ولی XF باز هم بر سهولت ان می افزاید. XF یک سازنده واسط کاربر محاور ای است. شما یک واسط کاربر با دستکاری اشیا بر صفحه نمایش طراحی می

کنید. سپس XF یک اسکریپت Tcl ایجاد می کند که واس کاربردی را که شما طراحی کرده اید تولید خواهد کرد. (Figure 2.4 را ببنید).

XF ابزارهایی برای ایجاد و پیکربندی ویدگتها، مرتب کردن آنها با مدیر هندسی Tk ایجاد انتیاد رویدادها و غیره را فراهم می کند. XF یک برنامه کاربردی در حال اجرا را دستکاری می کند. بنابراین از تمام لحاظ واسط کاربر می تواند مورد مشاهده قرار گیرد. و فوراً آزمایش شود.

XF از تمام کلاسهای ویدگت های توکار پشتیبانی می کند و به شما اجازه می دهد کلاسهای جدیدی از ویدگت با نوشتن اسکریپت های خاصی از Tel برای اداره کردن این کلاسها توسط XF را اضافه کنید. شما نیازی ندارید منحصراً از XF استفاده کنید: شما می توانید واسط کاربری طراحی کنید که قسمتی از آن با XF و بخشی دیگر از ان با اسکریپت های Tcl نوشته شده باشد. XF از بیشتر گسترش های موجود در دسترس اخیر که برای TK , Tcl ایجاد شده است پشتیبانی می کند و XF خودش در Tcl نوشته شده است.

XF توسط اسون دلماس (sven Delmas) ایجاد شده است. و اساس ان بر ساختن سریع واسط کاربر برای Tk می باشد که BYO نامیده می شود و در دانشگاه ویکتوریا در شهر ولینگتون نیوزلند توسعه یافته است.

٤-٩-٢) برنامه نویسی توزیع شده Tel (Tel-DP) مجموعه ای از فرمان های Tel است که گسترش و توسعه برنامه های توزیع شده را ساده تر می کند. یکی از مهمترین ویژگی های Tel-DP یک پروسی جر راه دور است. که اجازه می دهد برنامه های کاربردی Tcl با تبادل اسکریپتهای Tcl در ارتباط باشند. برای مثال اسکریپت زیر از Tel- DP برای پیاده سازی یک "idserver" ناچیز که شناسه های منحصر به فردی را در پاسخ به درخواست GeTID بر می گرداند استفاده می کند.

```
                                        Set myZD  0

Proc Get Id { } {
        Global myId ;
        Set myId [expr $myId + 1]
        Return $ myId
        }
        Make PRC Server  45 45
```

Figure 2-4: یک صفحه نمایش که پنجره اصلی XF را نشان می دهد. یک سازنده برنامه های کاربردی محاوره ای برای Tel , Tk.

تمام کدهای این اسکریپت به جز خط خر کدهای عادی Tel می باشند که یک متغیر عمومی myId و یک پروسی جر به نام Get Id که متغیر را افزایش داده و مقدار جدیدی آن را بر می گرداند تعریف می کند. فرمان Make PrcServer به وسیله Tel-DP پیاده سازی شده است و باعث می شود که برنامه کاربردی به درخواست های TCP در سوکت 45 45 گوش فرا دهد.

برنامه های Tcl دیگر نیز وجود دارند که می توانند با این Server با استفاده از اسکریپتهایی مانند اسکریپت زیر ارتباط برقرار کنند.

Set server [Make Preelient server. EmPany.com 45 45]

PRC $server GetId

فرمان اول یک اتصال با سرور ایجاد می کند و یک شناسه را برای این اتصال ذخیره می کند. آرگومان Make PRCelient میزبان سرور را شناسایی کرده و سوکتی را که به سرور گوش می دهد را تشخیص می دهد. فرمان Pre یک پروسی جر راه دور را اجرا می کند. آرگومانهای آن کی شناسه اتصال و یک اسکریپت دلخواه Tcl می باشند PRC این اسکریپت را به سرور می فرستد سرور این اسکریپت را اجرا کرده و نتیجه آن را باز می گرداند (در این مورد یک شناسه جدید) که نتیجه فرمان Pre می شود. هر اسکریپتی می توانست جانشین فرمان Get Id شود.

Tcl-DP همچنین ویژگی های دیگری را نیز شامل می شود. مثل فراخوان پرسی جرهای غیر همزمان از راه اور که کارگزارد (client) نیاز ندارد برای تکمیل فراخوانی انتظار بکشد، یک سیستم شیء توزیع شده که اشیاء آن می تواند در چندین برنامه کاربردی کپی شده و به روز رسانی شود. و به صورت خودکار به همه کپی ها انتقال یابد. و کی نام ساده برای این سرویس Tel-DP برای برنامه

های کاربردی از قبیل پخش سیستم های ویدئوئی، گروه افزارها و بازیها استفاده شده است. Tel- DP از بیشتر پروسی جر های فراخوان سیستم های راه دور انعطاف پذیرش است. زیرا آن براساس واسط کاربر کامپایل شده بین کلاینت و سرور نیست. در Tel- DP اتصال یک کلاینت به یک سرور جدید بدون کامپایل مجدد یا راه اندازی آن کلاینت آسان است.

AK (۲-۹-۵:

AK یک گسترش صوتی برای Tcl است. AK در راس یک فایل صوتی ساخته شده است، یک شبکه ناپیدا، سیستم صوتی که مستقل از سخت افزار بر روی انواع پلت فرم ها اجرا می شود. AK فرمانهایی از Tcl برای پخش و اجرای فایل (صوتی)، ذخیره سازی، کنترل تلفن و همزمانی مهیا کرده است. موضوع اصلی در AK اتصال به سورر فایل تی و وسایل زمینه و درخواست هایی از قبیل پخش فایل می باشد. برای مثال در اینجا یک اسکریپت برای پخش فایل صوتی در یک ماشین راه دور آورده شده است:

Avdioserver remore “Server. Com pany. Com: O”

Remote eontext room- device 1.

Root create Play “annoueement- file. Au”

فرمان اول یک اتصال به سرور صوتی روی ماشین Server. Com pany. Com ایجاد می کند. و به این اتصال نام remote را اختصاص می دهد. AK همچنین یک فرمان زمینه ای به نام room برای ارتباط با زمینه ایجاد می کند. آخرین فرمان پخش یک فایل صوتی را آغاز می کند. AK یک مدل زمانی منحصر به فرد را پیاده سازی می کند که اجازه می دهد کلاینت ها نمونه های صوتی را که در حال رخ دادن می باشند را به دقت تعیین مکانیسمی برای اجرای اسکریپتهای Tel در زمان های صوتی معین شده تهیه می کند. که می تواند برای به دستآوردن انواع آثار فوق رسانه ای از قبیل نمایش تصاویر یا ویدئو یا یک پخش صوتی استفاده شود.

Ak یک سیستم اسکریپتی قدرتمند و منطق برای گسترش و توسعه برنامه های کاربردی چند رسانه ای از قبیل سیستم های آموزشی و استعلام تلفنی تهیه می کند.

فصل سوم:

دستور زبان Tcl:

برای نوشتن اسکریپتهای Tcl شما باید دو نکته را بیاموزید. اول اینکه شما باید نکات دستوری Tcl را بیاموزید که شامل یک نیم دوجین از قوانینی است که چگونگی تجزیه فرامین Tcl را مشخص می کند. نکات دستوری Tcl برای هر فرمانی یکسان است. دوم اینکه شما باید چیزهایی در مورد فرامین مجزای Tcl که در اسکریپتهایتان استفاده می کنید بیاموزید. Tel در حدود ۶۰ فرمان نو کار در اختیار دارد، TK نیز فرمان های دیگری به آن می افزاید و هر برنامه کاربردی براساس Tel , Tk تعدادی از فرمان های خودش را به آن می افزاید. شم نیاز دارید تمام نکات دستوری و قواعد را بدانید اما فرمان هایی که نیازی دارید به تدریج می آموزید. این فصل نکات دستوری زبانی Tel را شرح می دهد. فصول باقیمانده در قسمت اول فرامین توکار را توصیف می کند و قسمت دوم رمانهای Tk را شرح می دهد.

۱-۳) اسکریپتها، فرمانها، لغات:

یک اسکریپت، Tel شامل یک یا چند فرمان می باشد. فرمان ها به وسیله خط جدید و یا سمی کالون از هم مجزا می شوند. برای مثال:

```
Set a 24
```

Set b 15

اسکریپت بالا شامل دو فرمان است که با کاراکترهای خط جدید از هم جدا شده اند. همین اسکریپت می تواند در یک خط نوشته شود. و فرمان ها با سمی کالون از هم جدا می شوند:

set a 24; set b 15

هر فرمان شامل یک یا چند کلمه می باشد که کلمه اول نام فرمانو کلمات دیگر آرگومانهای آن فرمان هستند. کلمات با فضاهای خالی از هم جدا شده اند. هر فرمان در مثال بالا ۳ کلمه دارد.

(Figure 3.1 فمانهای Tel در دو مرحله ارزیابی می شوند. ابتدا مفسر Tel فرمان را به کلماتی تجزیه می کند. و جانشینی ها را انجام می دهد. سپس یک پروسی جر فرمان از این کلمات نتیجه نهایی را تولید می کند هر فرمانی چندین پروسی جر فرمان دارد.

ممکن است تعدادی از کلمات در یک فرمان قرا رداشته باشند و هر کلمه ممکن است یک مقدار رشته ای دلخواه داشته باشد . فضاهای خالی که کلمات را از یکدیگر جدا می کنند جزئی از کلمات نیتسد. خط جدید و سمی کالون که خاتمه دهنده فرامین هستند نیز جزئی از کلمه به حساب نمی آیند.

۳-۲) ارزیابی یک فرمان:

Tel یک فرمان را در دو مرحله مورد ارزیابی قرار می دهد که در Figure 3.1 نشان داده شده است. تجزیه و تحلیل و اجرا در مرحله تجزیه و تحلیل مفسر Tel قواعد توصیف شده در این فصل را به کار بسته و فرمان ها را به کلمات تقسیم یم کند و جانشینی ها را انجام می دهد. تزیه و تحلیل دقیقاً برای تمام فرامین انجام می شود. در طی مرحله تجزیه وتحلیل مفسر Tel مقدار دهی مقدار دهی مقادیر کلمات را در متغیر a به جای "$a" را انجام می دهد. Tel نمی داند و یا اصلاً توجهی نمی کند که آیا a یا نتیجه کلمه یک عدد است. Tcl با کلمه اول مثل نام فرمان رفتار می کند. و بررسی می کند اگر فرمان تعریف شده باشد یک پروسی جر فرمان را برای اجرای تابعش مکان یابی می کند. اگر این فرمان تعریف شده باشد، سپس مفسر Tcl پروسی جر فرمانش را فراخوانی کرده، همه کلمات را به پروسی جر فرمان انتقال می دهد. پروسی جر های فرمان باری تفسیر کلمات

آزادند که از راهی که مناسبتر است اقدام کنند. و فرمانهای مختلفی را به آرگومانهایشان نسبت می دهند.

یادداشت: من واژه هایی مثل "word" , "argument" را متبادلاً به مقادیری که به پروسی جر فرمان انتقال می یابند نسبت داده ام. تنها اختلاف بین این دو این است که اولی آرگومان است و دومی کلمه است.

فرمان های زیر بعضی از معانی ای که به طور متداول به آرگومانها نسبت داده می وشند را شرح می دهد. Set a122

در خیلی از موارد از قبیل فرمان set آرگومانها ممکن است هر فرمی داشته بانشد. فرمان Set آرگومان اولی را به عنوان نظم متغیر و آرگومان دوم را به عنوان مقدار آن متغیر در نظر می گیرد. فرمان "set 122 a" نیز معتبر است. این فرمان یک متغیر به نام "122" ایجاد می کند که مقدار آن "a" می باشد.

expr 24/3-2

آرگومان expr باید یک عبارت ریاضی باشد که از قوانین توصیف شده در فصل ۵ تبعیت می کند. برای فرمان های دیگر این عبارت آرگومان محسوب می شوند.

Eval {set a 122}

آرگومان eval یک اسکریپت Tel است. Eval آرگومانش را به مفسر Tel ارسال می کند. و در آنجا این آرگومان تجزیه و تحلیل شده و اجرا می شود. فرمان های کنترلی مانند While , if نیز اسکریپت هایی به عنوان آرگومان ها می پذیرد.

Lindex {red green blu Puple} 2

اولین آرگومان lindex لیستی شامل ٤ مقدار می باشد که به وسیله فضاهای خالی از هم جدا شده اند. این فرمان دومین منحصر (“blue”) را از لیست انتخاب کرده و بر می گرداند. فرمان های TCl برای دستکاری لیست ها در فصل ٦ توضیح داده شده اند.

String length abraedabra

بعضی از فرمان ها مانند string و فرمان های ویدگت TK فرمان هایی هستند که در یک فرمان قرار می گیرند. آرگومان اولی فرمان یکی از چندین عملگر را برای اجرا انتخاب می کند و معانی آرگومانهای باقیمانده را تعیین می کند. بعنوان مثال “String compare” به یک آرگومان دیگر نیاز دارد و طول آن آرگومان را محاسبه می کند در صورتی که "string compare“ به دو آرگومان نیاز دارد.

Button . b- text Hello- fgred

آرگومانهای رشته ای با –text مقادیر جفتی منتخابی هستند که به شما اجازه می دهند گزینه ها را مشخص کنید و به مقادیر پیش فرض توجه کرده و ازآنها استفاده کنید.

در نوشتن اسکریپتهای Tcl یکی از مهمترین چیزهای یکه باید به یاد داشته باشید این است که پارسی Tcl وقتی که یک فرمان را تحلیل می کند لغات آن را معنی نمی کند و تمام معانی بالا توسط پروسی جرهای فرمانو نه پارسر Tcl شده است. به بیان دیگر می توانیم بگوییم که آرگومانها با مقادیر پیش فرض بیان شده اند و اگر شما می خواهید آنها را ارزیابی کنید باید این درخواست را به صورت آشکار بیان کنید. این رهیافت برای بیشتر زبان های لایه ای (shell language) یکسان است. اما بیشتر زبان های برنامه نویسی اختلاف دارد. باری مثال برنامه زر را که به زبان C نوشته شده است ملاحظه کنید.

```
X= 4;
Y = X+10;
```

در دستور اول مقادیر اینتیجر4 در متغیر X ذخیره می شود. در دستور دوم عبارت "X+10" مورد ارزیابی قرار می گیرد. مقدار متغیر X با عدد 10 جمع شده و حاصل در متغیر y ذخیره می شود و در انتهای اجرا متغیر y مقدار 14 را در بر دارد. اگر شما بخواهید یک رشته ثابت را بدون اینکه ارزیابی شود مورد استفاده

قرار دهید باید آن را در درون جفت گیومه بگذارید. حال برنامه ای مشابه برنامه بالا را در Tel ملاحظه کنید:

```
Set  X  4
Set  Y  X+10
```

فرمان اول رشته "4" را به متغیر X نسبت می دهد. مقداری متغیر نیازی نیست که فرم خاصی داشته باشد. فرمان دوم به سادگی رشته "X+10" را به عنوان مقداری جدید در Y ذخیره یم کند. در انتها اسکریپت Y دارای مقدار رشته ای "X+10" و نه مقدار اینتیجر 14 می باشد. در Tcl اگر شما بخواهید این عبارت را ارزیابی کنید باید این درخواست را به صورت آشکار بیان کنید.

```
Set  X  4
Set  Y  [expr $X+10]
```

در این مثال دو ارزیابی درخواست شده است. ابتدا کلمه دوم در فرمان دوم که در براکت قرار گرفته است به پارس Tcl می گوید که کاراکترهایی را که در براکت قرار گرفته اند را به عنوان یک اسکریپت Tcl در نظر گرفته و نتیجه آن را به عنوان مقدار این کلمه مورد استفاده قرار دهد. دوم اینکه علامت $ که قبل از X قرار گرفته است باعث می شودپارسی Tcl در فرمان expr مقدار متغیر x را در $x قرار دهد. اگر علامت $ از ابتدای x حذف شده بود آرگومان expr شامل یک

رشته x بود و در نتیجه یک خطای دستوری رخ می داد. در انتهای اسکریپت متغیر y مقدار رشته ای "14" دارد که تقریباً مثل مثال C می باشد.

۳-۳) جانشینی متغیرها:

Tcl سه فرم جانشینی را پیش بینی کرده است. جانشینی متغیرها، جانشینی فرمان و جانشینی بک اسلکش. هر جانشینی باعث می ود تعدادی از کاراکتر های اصلی یک کلمه با بعضی مقادیر دیگر جایگزین می شود. جانشینی ها در هر کلمه ای از فرمان ممکن است اتفاق بیفتد که این شامل نام فرمان نیز می شود و هر تعداد از جانشینی ها ممکن است در یک کلمه رخ دهد. اولین فرم از جانشینی، جانشینی متغیر می باشد. ماشه این جانشینی کاراکتر $ می باشد که سبب می شود مقدار یک متغیر در یک کلمه Tcl بجای آن متغیر قرار گیرد. برای مثال فرمان زیر را مورد ملاحظه قرار دهید:

```
Set kgrams 20
Expr $kgram S*2. 2046
⇒ 44.092
```

فرمان اول مقدار متغیر Kgrams را برابر 20 قرار می دهد. فرمان دوم مقدار وزن را برحسب پاوند محاسبه یم کند بدین ترتیب که مقدار Kgrams را در 2.2046 ضرب می کند. این عمل با استفاده از جانشینی متغیر انجام می شود: رشته $

Kgrams با مقدار متغیر Kgrams جایگزین یم شود بنابراین آرگومان دریافت شده توسط پروسی جر فرمان expr مقدار "20*2.2046" می باشد. جانشینی متغیر در هر جایی در یک کلمهو به هر تعداد زمانی می تواند اتفاق بیفتد. فرمان زیر را ملاحظه کنید:

```
expr $result *$base
```

نام متغیر همه اعداد و متن ها و زیر واژه ها را که به دنبال $ می آیند را شامل می شود. بنابراین نام متغیر اول (resolt) به * محدود می شود. و نام متغیر دوم (Base) نیز به انتهای کلمه محدود می شود. مثال های بالا ساده ترین فرم جانشینی متغیرها را نمایش می دهد. دو فرم دیگر برای جانشینی متغیر وجود دارد که برای آرایه های انجمنی (شرکتپذیر) استفاده می شوند و کنترل آشکارتری را بر روی توسعه نام متغیرها فراهم می کند. این فرم ها در فصل ٤ بحث می شوند.

٤-٣) جانشینی فرمان:

دومین فرمان جانشینی که توسط Tel فراهم شده است جانشینی فرمان می باشد جانشینی فرمان باعث می شود که قسمتی از یک کلمه فرمان و یا تمام آن با نتیجه فرمان دیگری از Tel جایگزین می شود. جانشینی فرمان با قرار دادن یک فرمان تو در تو در داخل براکت فراخوانی می شود:

```
set kgrams 20
```

```
set tbs [expr $kgrams*.2046]
⇒ 44.092
```

کاراکترهایی که در بین براکتها قرار گرفته اند باید یک اسکریپت Tel معتبر تشکیل دهند. این اسکریپت ممکن است هر تعداد از فرامین مجزا شده بـه وسـیله خطـوط جدید یا سمی کالون را شامل شود. این براکتها و تمام کاراکتهایی کـه بـین ایـن دو براکت قرار می گیرند. با یک نتیجه نهایی جایگزین می شوند. بنابراین در مثال بـالا فرمان expr زمانی که کلمات seg تجزیه و تحلیل می شوند اجرا شده و نتیجه آن که رشته "44.092" می بشد بعنوان دومین آرگومان set در نظر گرفته می شـوند درست مثل جانشینی متغیر، جانشینی فرمان نیز می تواند در هر جایی از یک کلمـه رخ دهد و در یک کلمه ممکن است بیش از یک جانشینی فرمان انجام شود.

۵-۵) جانشینی بک اسلش:

آخرین فرم جانشینی در Tel جانشینی بک اسلش می باشد. جانشینی بـک اسـلش برای وارد کردن کاراکترهای خاص از قبیل خطوط جدیـد در کلمـات و همچنـین وارد کردن کارکترهایی مانند I , $ بدون اینکه پـارس Tcl رفتـار خصـی بـا آنهـا داشته باشد.

برای مثال فرمان زیر را ملاحظه کنید:

```
set msg Eggs: $2.18/ dozen:\ nGasoline:\\ $1.49/ gallon
```

⇒ Eggs: $2.18/ dozen

Gasoline: $1.49/ gallon

در اینجا دو دنباله متوالی از یک اسلش ها وجود دارند که به دنبال آنها فضاهای خالی قرار گرفته اند. هر کدام از این دنباله ها با یک تک فضای خالی در کلمه قرار گرفته اند و این کاراکتر فضای خالی به عنوان جدا کننده کلمات تلقی نمی شوند.

در اینجا دو دنباله از یک اسلش ها وجود دارند که به دنبال آنها علامت $ قرار گرفته است که هر کدام از اینها در کلمه با تک تک علامت $ جایگزین می شوند و این دالرساین ها ($) بعنوان کاراکترهای عادی محسوب می شوند (آنها بعنوان ماشه جانشنی متغیر عمل نمی کنند). یک اسلشی که بدنبال آن n قرار گرفته است با یک خط جدید جایگزین می شود.

جدول ۱-۳ لیستی از توالی یک اسلش ها را که Tel از آنها پشتیبانی می کند آورده است. که تمام تولی هایی را که در ANSIC تعریف شده مثل \t برای وارد کردن یک کاراکتر تب و \xd4 برای وارد کردن کاراکتری که مقدار آن در مبنای ۱۶ برابر oxd4 می باشد را شامل می شود. اگر به دنبال یک اسلش کاراکتری قرار گیرد که در جدول قید نشده است، مثل \$ یا ? \ سپس یک اسلش از کلمه حذف شده و کاراکتری که به دنبال آن می آید بعنوان یک کارکتر عادی در کلمه قرار می گیرد. این به شما اجازه می دهد هر کارکتر خاصی از Tcl را در یک کلمه

به کار برید بدون اینکه پارس Tcl رفتار خاصی بااین کارکتر داشته باشد. دنبالـه \\ یک بک اسلش تکی را در کمله قرا رمی دهد. توالی یک بک اسلش و خط جدیـد می تواند برای انتشار یک فرمان طولانی در چندین خط مورد اسـتفاده قـرار گیـرد، به مثال زیر توجه کنید:

Paek. Base. Lable 1. Poerw. Lable 2 . result\

- side left- Padx 1m- pady 2m

یک اسلش و خط جدید بعلاوه هر فضای عمده ای در خـط بعـدی بـا یـک تـک کاراکتر فضای خالی در کلمه جایگزین می شود. بنابراین هر دو خـط بـا همـدیگر بعنوان یک تک فرمان محسوب می شوند.

یادداشت: توالی بک اسلش- خط جدید و جایگزینی آنها در مرحله پیش پـردازش قبل از اینکه مفسر Tel این فرمان را تجزیه کند غیر معمول است. این به معنـی آن است که برای مثال ، فضای خالی که جایگزین یک اسلش خـط جدیـد مـی شـود بعنوان یک جدا کننده کلمه تلقی خواهد شد مگر اینکـه درون جفـت کوتیشـن یـا آکولاد قرار گیرد.

۳) نقل قول با جفت کوتیشن:

Tel راه های مختلفی را برای جلوگیری از تفسـیر خـاص کارکترهـایی مثـل & و سمی کالون توسط پارس پیش بینی کرده است. این تکنیـک هـا نقـل قـول قـوای

(quoting) نامیده می شود. شما هم اکنون یکی از فرم های qouting را در جانشینی یک اسلش دیده اید. برای مثال $ \ باعث می شود یک علامت $ بدون اینکه یک جانشینی متغیر را موجب شود در داخل کلمه قرار گیرد. بعلاوه جانشینی بک اسلش در Tcl دو فرم دیگر از نقل قول (qouating) پیش بینی کرده است: جفت کوتشن ها و آکولادها.

جدول ۱-۳) جانشینی های بک اسلش که در Tel فراهم شده است. هر یک از توالی ها در ستون اول با کاراکتر نظیرش در ستون دوم جایگزین خواهد شد. اگر به دنبال بک اسلش کاراکتر دیگری به جز کاراکتر های قید شده در ستون اول قرار گیرد سپس هر دو کاراکتر با کاراکتر دوم جایگزین خواهند شد.

جفت کوتیشن ها جد کننده های فرامین و لغات را بی اثر می سازند. آکولادها تقریباً تمام کاراکترهای خاص را بی اثر می سازند. اگر یک کلمه در درون جفت کوتیشن قرار گیرند سپس فضاهای خالی، تب ها ، خطوط جدید و سمی کالون ها بعنوان کاراکترهای عادی در کلمه محسوب می شوند. مثالی که قبلاً مطرح شد می تواند به صورتی زیباتر و تمیزتر مطابق زیر نوشته شود:

aet msq “Eggs: $2.18/ dozen”

Gasoline: \$1.49/gallon

اما من فکر می کنم که این اسکریپت با \n خواناتر است. جانشینی متغیرها و جانشینی فرمان و جانشینی یک اسلش همگی مثل همیشه در داخل جفت کوتیشن اتفاق می افتد . برای مثال اسکریپت زیر یک رشته را که شامل نام یک متغیر، مقدارش و مجذور آن مقدار می باشد را به imsg اختصاص می دهد:

set a 2.1

set msg "a is $a ; the square of a is [expr$a * &a]"

⇒ a is 2.1 ; the square of a is 4.41

اگر شما بخواهید یک کلمه شامل جفت کوتیشن باشد و خود کلمه نیز در درون جفت کوتیشن قرار گیرد باید از جانشینی یک اسلش استفاده کنید:

set name a. out

set msg "could,'t open file\ "$name"\ " "

⇒ couldn't open file "a. out"

۷-۳) نقل قول با آکولاد:

آکولادها فرم ریشه ای تری از نقل قول (quoting) را پیش بینی کرده اند که درآن تمام کاراکترهای خاص در معنای خود را ا زدست می دهند. اگر یک کلمه در درون آکولاد قرار بگیرند تمام کاراکترهایی که در بین آکولادها قرار گرفته اند، به

عنوان مقدار آن کلمه در نظر گرفته می شوند. هیچ جانشینی در کلمه انجام نمی شود و فضای خالی، تب ها، خطوط جدید و سمی کالون ها بعنوان کاراکترهای عادی در نظر گرفته می‌شوند. مثال صفحه 30 می تواند با آکولادها به صورت زیر نوشته شود.

Set msg {Eggs” $1.49/ gallon}

دالرساین بعنوان ماشه جانشینی متغیر در کلمه عمل نمی کند. و خط جدید به عنوان جدا کننده فرمان محسوب نمی شود. در این مورد ۱n نیز نمی تواند بار یدرج یک خط جدید در یک کلمه مثل مثالهای قبل عمل کند. چون د راین آرگومان جانشینی یک اسلش انجام نمی شود:

Set msg {Eggs: $2.18/ dozen\ n Gasoline: $1.49/ gallon

⇒ Eggs= $2.18/ dozen\ nGasoline: $1.49/ gallon

یکی از مهمترین استفاده های آکولادها بتعویق انداختن ارزیابی باشد. ارزیابی که به تعویق افتاده است باعث می شود پارس Tel کاراکتر های خاص را فوراً تجزیه و تحلیل نکند. به جای آن کاراکترهای خاص بعنوان قسمتی از آرگومان به پروسی جرهای فرمان منتقل می شوند و پروسی جر فرمان خودش بر کاراکترهای خاص تاثیر می گذارد. اکولادها تقریباً همیشه وقتی اسکریپتها به فرامین Tcl منتقل می شوند. استفاده می شوند مثل مثال زیر که فاکتوریل عدد ۵ را محاسبه می کند:

```
set result 1
set I 5
While {$i > 0 }
        Set result [expr $result* $I]
        Set i [expr $i – 1]
```

بدنه حلقه while برای تعویق جانشینی ها در درون آکولاد قرار مـی گیـرد. حلقـه While اسکریپت را برای ارزیابی به Tel بر می گرداند و در هر بار تکـرار حلقـه جانشینی ها انجام می شوند. د راین مورد به تعویق انداختن جانشینی ها مهم است برای اینکه این جانشینی ها هر بار که بدنه حلقه While مورد ارزیـابی قـرا رمـی گیرد از نو انجام می شوند. تا اینکه این جانشینی یک بار انجام شود. به آکولادهـای تو در تو در مثال زیر توجه کنید:

```
Proc power {base P}
        Set result 1
        While {$P>0}
        Set result [expr $result* base]
        Set P [expr &P-1]
        }
        return $result
}
```

در این مثال سومین آرگومان Proe دو جفت آکولاد پی در پی می باشـد (بیرونـی ترین آکولادها به وسیله پارس Tel حذف می شوند). جانشـینی فرمـان در مـورد

[expr $P-1] وقتی که فرمان Proe توسط پارس تجزیه و تحلیل می شود انجام نمی گیرد و یا حتی وقتی که فرمان While دومین آرگومانش را برای اجرای حلقه بازیابی می کند جانشینی انجام خواهد شد.

یادداشت: اگر یک آکولاد با یک اسلش استفاده شود برای این آکولاد باز شده لازم نیست به دنبال یک آکولاد بسته باشیم. وقتی که کلمه توسط پارس تجزیه و تحلیل می شود یک اسلش حذف نمی شود.

یادداشت: تنها فرم جانشینی که در بین آکولاد ها اتفاق می افتد یک اسلش- خط جدید می باشد. طبق مباحث مطرح شده در بخش ۵-۳ توالی های یک اسلش- خط جدید، در مرحله پیش پردازش و قبل از تجزیه و تحلیل فرمان توسط پارس، حذف می شوند.

۸-۳) توضیحات:

اگر اولین کالراکتر یک فرمان # و تمام کاراکتر هایی که به دنبال آن قرار می گیرند بعنوان توضیحات محسوب می شوند و مورد پردازش قرار نمی گیرند توجه کنید که نشان در هم باید در موقعیتی که Tcl انتظار اولین کاراکتر فرمان را دارد رخ دهد. اگر یک نشان در هم در هر جایی غیر از جای مقرر قرار گیرد بعنوان یک کاراکتر عادی تلقی می شود که بخشی از یک کلمه فرمان را می سازد:

```
# this is a comment
```

```
set a 100                    # Not a comment
wrong #args: Should be “set vor Name? New value?”
set b 101 ;                  # this is a comment
⇒ 101
```

کاراکتر # در خط دوم بعنوان یک توضیح تلقی نمی شود. چون در میانه یک فرمان ظاهر شده است. اولین فرمان set 6 آرگومان دارد و یک نتیجه خطا تولید خواهد کرد. آخرین کاراکتر # بعنوان یک کاراکتر توضیح محسوب می شود و وقتی این اتفاق می افتد که فرمان با یک سمی کالون خاتمه یافته باشد.

۹-۳) بازگشت های استثنائی و طبیعی:

یک فرمان Tcl با راه های مختلفی می تواند خاتمه یابد. متداولترین مورد یک بازگشت طبیعی می باشد. بدین معنی که فرمان با موفقیت تکمیل شده و یک رشته را به عنوان نتیجه بر می گرداند. Tel همچنین از بازگشت های استثنائی از فرمان ها پشتیبانی می کند. متداولترین فرم بازگشت استثنائی یک خطا می باشد. وقتی که یک خطا رخ می دهد بدین معنی است که فرمان نمی تواند تابع مربوطه اش را تکمیل کند. فرمان متوقف شده و هر فرمانی که به دنبال آن در اسکریپت می آید نیز در نظر گرفته نمی شود. یک بازگشت خطا شامل رشته ای است که نشان می دهد چه اشتباه و خطایی رخ داده است. این رشته معمولا توسط برنامه کاربردی

نمایش داده می شود. برای مثال فرمان set که در زیر آورده شده است یک خطا تولید می کنند چون دارای آرگومانهای زیادی است:

set state west virginal

Q wrong # args: should be “set var name? New value?”

فرمان های مختلف تحت شرایط متفاوت خطاهای گوناگونی ایجاد می کنند. بعنوان مثال فرمان expr هر تعداد آرگومان می تواند داشته باشد ولی این آرگومانها باید از لحاظ دستوری شرایط خاصی داشت باشند. بعنوان مثال پرانتز ها باید با هم هماهنگ باشند یعنی هر پرانتزی که باز می شود بسته شود و پرانتزی که بسته می شود قبلا باز شده باشد وگرنه یک خطا پیش می آید:

expr 3* (20 + 4

unmathed parentheses in expression “3* (20 + 4”

مکانیسم بازگشت استثنائی به صورت تکمیلی در فصل ۹ بحث خواهد شد. که تعدادی از بازگشت های استثنائی از خطاها را پشتیبانی می کند، اطلاعات بیشتری راجع به خطاها و پیغام های خطا علاوه بر آنچه که در بالا ذکر شده است ارائه می دهد. و اجازه می دهد که خطاها رسیدگی شود. در حال حاظر گرچه تمام چیزی که شما نیاز دارید بدانید فرمان هایی هستند که رشته هایی معمولی را به عنوان

نتیجه بر می گردانند اما گاهی اوقات خطاهایی را بر می گردانند که باعث توقف تفسیر یک فرمان Tel می شوند.

یادداشت: شاید در این مواقع متغیر error Info مفید باشد. بعد از بروز خطا Errorinfi Tcl برای نگهداری یک پشته و ردیابی و تعیین محل وقوع خطا استفاده می کند. شما می توانید این متغیر را با فرمان "set errorIndo" به کار ببرید.

نکات بیشتری راجع به جانشینی ها:

شاید متداولترین مشکلی برای کاربران جدید Tcl درک این نکته باشد که جانشینی ها کجا اتفاق می افتد. و کجا اتفاق نمی افتد. سناریوی واقعی این است که کاربران از رفتار اسکریپتهای Tcl متعجب می شوند. چون وقتی که انتظار دارند که جانشینی اتفاق نیفتد جانشینی رخ می دهد. یا هنگامی که انتظار دارند رخ دهد. رخ نمی دهد. اما من فکر می کنم اگر شما دو قاعده را به خاطر بسپارید مکانیسم جانشینی های Tcl ساده و قابل پیش بینی خواهد شد.

۱. Tcl فرمان ها را در هر گذر از چپ به راست تجزیه و تحلیل می کند. هر کاراکتر دقیقاً یک بار خوانده می شود.

۲. منتها یک لایه از جانشینی برای هر کاراکتر اتفاق می افتد، نتیجه یک جانشینی باری جانشینی های دیگر در نظر گرفته نمی شود.

جانشینی های Tcl ساده تر و منظم تر از آن هستند که شما از شل (Shell) مربوط به UNIX برای برنامه نویسی استفاده کرده باشید (مخصوصاً csh). زمانی کاربران تازه کار با مسائل مختلفی در مورد جانشینی ها روبرو می شوند که مسائل را پیچیده تر از واقعیات موجود فرض می کنند برای مثال فرمان زیر را در نظر بگیرید:

```
set X [format {Earning for july: $%.2f} $earnings]
⇒ Earnings for July: $ 1400.26
```

کاراکترهایی که در بین براکت ها قرا رگرفته اند دقیقاً یک بار در طی جانشینی فرمان set را برای جانشینی متغیر ها بخواند. سپس یک گذر دیگر برای جانشینی فرمان انجام دهد. بلکه همه چیز در یک بار خواندن اتفاق می افتد. تمام فرمان format کلمه به کلمه به عنوان آرگومان دومن به فرمان Set منتقل می شود بدون اینکه هیچ خواندن اضافی صورت گیرد. (برای مثال دالرساین ($) بعنوان مثال ماشه جانشینی متغیر عمل نمی کند).

یک دنباله از قواعد جانشینی می گوید مرزهای کلمات در یک فرمان باید فوراً مشخص شود و این مرزها متاثر از جانشینی ها نیست. برای مثال اسکریپت زیر را ملاحظه کنید:

```
set city “los mgeles”
set bigeity $city
```

دومین فرمان set بدون توجه به مقدار متغیر city تضمین شده که سه کلمه دارد. در این مورد City شامل یک کاراکتر فضای خالی می باشد اما این فضای خالی به عنوان جدا کننده کلمات محسوب نمی شود. در بعضی از مواقع یک لایه تکی از قواعد جانشینی بجای کمک کردن می تواند مانع کار شود. برای مثال اسکریپت زیر به طور اشتباه سعی می کند فایلهایی را که نام آنها به ”.O“ ختم می شود. را حذف می کند:

```
exec rm [glob *.O]
rm: Q.O  B.O  C.O  nonexistent
```

فرمان glob یک لیست از نام فایل هایی را که نام آنها با الگوی ”.O“ مطابقت دارد را بر می گرداند. مثل ”a.o b.o c.o". سپس فرمان exec سعی می کند با فراخوانی برنامه rm تمام این فایل ها را حذف می کند. چون نمی تواند یک فایل به نام ”a.o b.o c.o" پیدا کند. برای اینکه rm به درستی کار کند باید نتیجه glob در چندین کلمه ظاهر شود.

خوشبختانه افزودن لایه های اضافی تجزیه و تحلیل (Parsing) کار ساده ای است. به یاد آورید که فرمان های tel در دو فاز مورد ارزیابی قرار می گیرند: تجزیه و تحلیل (Parsing) و اجرا. قوانین جانشینی فقط در فاز تجزیه و تحلیل (Parsing) به کار بدره می شوند. زمانی که Tcl کلمات فرمان را برای اجرا به پروسی جر فرمان انتقال می دهد، پروسی جر فرمان هر کاری که بخواهد می تواند با آنها انجام دهد. بعضی از فرمان ها کلماتشان را مجدداً مورد تجزیه و تحلیل قرار می دهند. باری مثال با انتقال مجدد کلمات به مفسر Tcl. Eval یکی از این قبیل فرمان ها است و می تواند برای حل مسائلی به rm استفاده شود.

Eval exec rm [glob *.O]

Eval تمام آرگومانهایش را با فضاهای خالی که در بین آنها قرار دارد به هم الحاق کرده و نتیجه را به عنوان یک اسکریپت Tcl مورد ارزیابی قرار می دهد و بعد دور تجزیه و تحلیل و ارزیابی اتفاق می افتد. در این مثال فرمان Eval سه آرگومان دارد. "a.o b.o c.o" و "exec" , "rm". این رشته وقتی بعنوان یک اسکریپت Tcl تجزیه و تحلیل می شود. به پنج کلمه تقسیم می شود: هر نام فایلی به exec و سپس بعنوان یک آرگومان مجزا به برنامه rm منتقل می شود. بنابراین تمام فایلها با موفقیت جابجا می وشند.

یک یادداشت نهایی: استفاده از جانشینی ها را با راه های پیچیده امکان پذیر است. اما من اصرار می کنم که این کار را انجام ندهید. جانشینی ها مفیدند در صورتی که خیلی ساده استفاده شوند. مثل "set a $b". اگر شما از جانشینی های بی شما و مخصوصاً تعداد زیادی از بک اسلش در یک فرمان استفاده کنید کد شما احتمالاً ناخواناب و نامطمئن خواهد شد. در جانشینی هایی مثل این من پیشنهاد می کنم این فرمان را به چندین فرمان تقسیم کنید که از آرگومانهای ساده تر تشکیل شده باشد. Tcl چندین فرمان از قبیل format و list را فراهم کرده است که این کار را راحت می کند.

فصل چهارم:

متغیرها:

Tcl از دو گونه متغیر پشتیبانی می کند متغیرهای ساده و آرایه های انجمنی. این فصل فرمان های اصلی دستکاری متغیرها و آرایه ها را توصیف کرده و همچنین شرح کاملتری از جانشینی متغیرها را ارائه می دهد. جدول ۱-٤ را برای خلاصه از فرمان های بحث شده در این فصل مشاهده کنید.

۱-٤) متغیرهای ساده و فرمان set

یک متغیر اسده Tcl دو چیز را شامل می شود. یک نام و یک مقدار. نـام و مقـدار هر دو می توانند رشته هایی دلخواهی از کاراکتر باشند. برای مثال یک متغیر ممکن است نام "XYZ" یا "XYZ! #22" یا "March earnings. $100 , 472" داشته باشد. در محل نام متغیر معمولا با یک حرف شـورع مـی شـود و ترکیبـی از حروف و ارقام و زیر واژه ها می باشد که جانشـینی متغیـر را سـاده تـر مـی کنـد متغیرها توسط فرمان ser، ایجاد شده، خوانده و اصلاح می شوند. این فرمـان مـی تواند یک یا دو آرگومان داشته باشد. اولین آرکومان نام متغیـر و دومـین آرگومـان اگر وجود داشته باشد مقدار جدید متغیر می باشد:

set a {Eggs.$2.18/ dozen}

⇒ Eggs: $ 2.18/ dozen

set a

⇒ Eggs: $ 2.18/ dozen

set a 44

⇒ 44

append var Name vakue? Value …?

هر مقدرای از آرگومـان را بـه Var Name الحـاق مـی کنـد. اگـر Var Name موجود نباشد قبل از الحاق ایجاد میشود و با یک رشته تهیه مقـداردهی مـی شـود. مقدار برگشتی مقدار جدید Var Name می باشد.

Iner Var Name? Inerement?

Inerement به مقدار متغیر Var Name اضافه می شد. Inerement و مقدار قبلی

Var Name باید هر دو رشته اینتی جر باشند. (دسی مال، هگزادسیمال یا اکتال) اگر Inerement حذف شود سپس مقدار پیش فرض ۱ خواهد بود. مقدار جدید بعنوان یک رشته دهدهی در Var Name ذخیره شده و بعنوان نتیجه برگردانده می شود.

Set Var Name? Value?

اگر Value مشخص شده باشد مقدار متغیر Var Name به Value نسبت داده می شود. در هر مورد این فرمان یک مقدار متغیر جدید بر می گرداند.

Uset Var Name? Var Name Var Name …?

متغیرهای Var Name را حذف می کدن. یک رشته خالی بر می گرداند.

جدول ۱-٤) خلاصه ای از فرمان های اصلی برای دستکاری متغیرها. آرگومان هایی که در درون علامت سوال قرار گرفته اند نشان دهنده اختیاری بودن آرگومان هستند.

اولین فرمان بالا یک متغیر جدید به نام a را اگر وجود نداشته باشد ایجاد می کند و مقدار آن را برابر دنباله ای از کاراکترهای "Eggs: $ 2.18/ dozen" قرار می دهد. نتیجه این فرمان مقداری جدید این متغیر می باشد. دومین فرمان set فقط یک آرگومان دارد. و آن a می باشد. در این مورد این فرمان به سادگی مقدار چند متغیر را بر می گرداند. سومین فرمان سشف مقدار متغیر a را به 44 تغییر می دهد و این مقدار جدید را بر می گرداند گرچه به نظر می رسد مقدار نهایی متغیر a یک عدد اینتیجر دهی باشد ولی به صورت یک رشته کاراکتری ذخیره می شود. متغیرهای Tcl می توانند برای نمایش خیلی چیزها مثل اینتی جرها، اعداد اعشاری ممیز شناور، اعداد، نام ها، لیست ها و اسکریپتهای Tcl به کار روند . ولی همیشه به صورت یک رشته ذخیره می شوند. این ذخیره سازی یکسان برای تمام مقادیر اجازه می دهد مقادیر مختلف با روش یکسانی دستکاری شوند. و به راحتی به هم مرتبط شوند. متغیرها Tcl وقتی مقدار دهی می شوند به صورت اتوماتیک ایجاد می شوند. متغیرها نوع ندارند بنابراین لازم نیست اعلان شوند.

۲-٤) آرایه ها:

علاوه بر متغیرهای ساده Tcl آرایه ها را نیز پیش بینی کرده است. یک آرایه یک مجموعه ای از عناصری است که هر عنصر یک متغیر با نام و مقدار می باشد. نام

هر عنصر آرایه دو بخش دارد. نام آرایه و نام مختصر آرایه. نام آرایه ها و نام عناصر هر دو ممکن است رشته های دلخواه باشند. به همین دلیل آرایه ها گاهی اوقات آرایه های انجمنی نامیده می شوند. تا از آرایه های زبانهای دیگر که نام عناصر باید اینتی جر باشند مشخص شوند.

به عواصر آرایه می توان با یک نشان مثل earning (January) رجوع کرد که نام آرایه (در این مورد (earning) می باشد که به دنبال آن نام عنصر در پرانتز قرار گرفته است (در این مورد January). آرایه ها در هر جایی که متغیرهای ساده ممکن است استفاده شوند می توانند به کار روند مثل فرمان setکه در مثال زیر آمده است:

set earnings (January) 87966

⇒ 87966

Set earnings (February) 95400

⇒95400

set earnings (January)

⇒ 87966

اولین فرمان یک آرایه به نام earnings می سازد البته در صورتی که این آرایه وجود نداشته باشد.

سپس یک عنصر به نام January را در صورتی که وجود نداشته باشد در آرایه می سازد و مقدار آن را برابر 87966 قرار می دهد. دومین فرمان یک مقدار را به عنصر Februory از آرایه نسبت می دهد و فرمان سوم مقدار عنصر January را بر می گرداند.

۳-٤) جانشینی متغیرها

فصل سوم استفاده از کلمات $ را برای جانشینی مقادیر در متغیر در فرامین Tcl معرفی کرد. این بخش این مکانیسم را با جزئیات بیشتری بررسی می کند. جانشینی متغیر با یک علامت $ که دردرون کوتیشن قرار نگرفته است در یک فرمان Tcl فعال می شود. کاراکترهایی که به دنبال $ می آیند به عنوان یک نام متغیر تلقی میشوند و علامت $ و این نام متغیر در یک کلمه با مقادیر این متغیر تعویض می شوند. Tcl سه فرم جانشینی متغیرها را پیش بینی کرده است. تاکنون شما فقط فرم ساده جانشینی متغیر را دیده اید که در این مثال آمده است:

```
expr $a + 2
```

در این فرم یک نام متغیر که شامل حروف، ارقام و زیر واژه ها می باشد و به دنبال یک $ قرار می گیرد. اولین کاراکتر یک حرف یا رقم یا زیر واژه (به عنوان مثالف "+") که این نام را خاتمهمی دهد نیست.

دومین فرم از جانشینی متغیرها اجازه می دهد که جانشینی در مورد عناصر آرایه صورت پذیرد. این فرم شل حالت اول است بجز اینکه به دنبال نام متغیر بلافاصله نام عنصر در درون پرانتز قرار می گیرد.متغیر فرمان، و جانشینی بک اسلش نیز به همین روش بر روی نام عناصر مثل کلمه فرماندر جفت کوتیشن اجرا می شود و فضاهای خالی در نام عناصر به عنوان بخشی از نام عنصر محسوب می شوند. نه جدا کننده کلمات. باری مثال اسکریپت زیر را ملاحظه کنید.

```
Set year Total O
Foreach mount {san feb Mar Apr may Jun Jul Aug sep\
                Oct NOV Dec}  {
Set year Total [expr $year Total + $earnings ($month
  }
```

در فرمان "$earnings ($month)" با یک مقدار عنصر آرایه earnings جایگزین می شود. در نام عنصر مقدار o کنعیر month قرار می گیرد که درهر تکرار تغییر می یابد. سومین فرم جانشینی برای متغیرهای ساده در جاهایی استفاده می شود که به دنبال نام متغیر یک حروف یا عدد و یا یک زیر واژه گرفته باشد.

برای مثال فرض می کنیم شما بخواهید یک مقداری مثل "1.5m" را بعنوان یک آرگومان به یک فرمان انتقال دهید اما این عدد در متغیر size قرار دارد (در TK ممکن است انی کار را برای مشخص کردن یک اندازه در میلی متر انجام دهید)

اگر شما سعی کنید که مقدار متغیر را به فرمی مثل "$sizem" جانشینی کنید سپس Tel با m بعنوان بخشی از نام متغیر رفتار خواهد کرد. برای حل این مسأله شما می توانید نام متغیر را مثل فرمان زیر در آکولاد قرار دهید.

.Canvas configure – with $ {size} m

شما همچنین می توانید از آکولاد ها باری مشخص کردن نام متغیرهایی که شامل کاراکترهای دیگری از حروف و یا اعداد یا زیر واژه ها می باشند استفاده کنید.

یادداشت: آکولاد ها می توانند برای تعیین حدود متغیرهای ساده استفاده شوند. اما برای آرایه ها زمانی که پرانتز ها انتهای نام متغیر را نشان می دهند ضرورتی ندارد.

مکانیسم جانشینی متغیرهای Tcl فقط برای رسیدگی به جانشینی های متداول می باشد، سناریوهایی وجود دارد که با هیچ کدام از فرم های جانشینی که قبلاً مطرح شده نمی توان نتیجه مطلوب را به دست آورد. جانشینی های پیچیده تری می توانند با دنباله ای از فرمان ها انجام شوند. برای مثال فرمان Format می تواند برای تولید یک نام متغیر از هر فرم قابل تصوری استفاده شود، فرمان set می تواند برای خواندن یا نوشتن متغیر با نام استفاده شود و فرمان های جانشینی می تواند برای جانشینی مقدار یک متغیر به درون فرمان های دیگر استفاده شود.

٤-٤) حذف متغیرها: unset

فرمان unset متغیرها را از بین می برد. این فرمان می تواند به هر تعدادی آرگومان داشته باشد که هر یک نام یک متغیر می باشد و این فرمان تمام این متغیرها را حذف می کند. بعد از آن هر تلاشی برای خواندن متغیرها یک نتیجه خطا در بر خواهد داشت گویی چنین متغیری هرگز توسط فرمان سئف مقدار دهی نشده است. آرگومانهای فرمان unset ممکن است متغیرهای ساده، عناصر آرایه و یا کل آرایه باشند مثل مثال زیر:

unset a earningsl (January) b

در این مورد متغیرهای a , b تماماً حذف می شوند و عنصر January از آرایه earnings نیز حذف می شود. آرایه earnings بعد از فرمان unset همچنان وجود خواهد داشت. اگر a یا b آرایه باشند تمام عناصر این آرایه ها حذف می شوند.

۵-٤) آرایه های چند بعدی:

Tel فقط آرایه های یک بعدی را پیاده سازی می کند اما آرایه های چند بعدی نیز می توانند با الحاق اندیس های چند گانه بهدورن عناصر یگانه شبیه سازی شوند.

برنامه زیر یک آرایه دو بعدی را با اندیس های اینتی جر شبیه سازی می کنند.

Set matrix (1,1) 140

Set matrix (1,2) 218

```
Set   matrix  (1,3) 84
Set   i     j
Set   j     2
Set   cell  $matrix ($i , $j)
```

$\Rightarrow$ 21

٦-٤) فرمان های append , incr:

incr و append راههای ساده ای برای تغییر مقدار متغیر فراهم می کنند. Incr در آرگومان دارد که نام یک متغیر و یک مقدار اینتی جر می باشد. این فرمان مقدار اینتی جر را به مقدار متغیر اضافه می کند و نتیجه را به صورت یک رشته دهدهی درون متغیر ذخیره می کند ومقدار جدید متغیر را به عنوان نتیجه بر می گرداند.

```
Set  X  43
Iner X  12
```

این عدد می تواند مقدرای منفی یا مثبت داشته باشد، این مقدار می تواند حذف شود که در این صورت مقدار پیش فرض ۱ خواهد بود:

```
Set  X  43
Iner X
```

$\Rightarrow$ 44

هر دو مقدار اصلی متغیر و مقداری که به متغیر افـزوده مـی شـو بایـد یـک رشـته صحیح باشد، خواه در مبنای ده باشند ای در مبنای هشت (که بـا O مشـخص مـی شود) یا در مبنای شانزده (که با OX مشخص می شود)

فرمان append یک متن را به انتهای متغیر می افزاید. این فرمان دو آرگومان دارد که شامل نام متغیر و متن جدیدی می باشد که باید به آن افزوده شود. این فرمان متن جدید را به متغیر ضمیمه می کند و مقدار جدید متغیر را باز می گرداند. مثال زیر از فرمان append برای محاسبه مجذور یک جدول استفاده می کند.

```
Set msg “ “
Foreach i {1    2   3   4   5}  {
      Append msg “$i squred is [expr $ i* $i] \n”
      }
set msg
⇒ 1       squared  is   1
    2           squared  is   4
    3           squared  is   9
    4           squared  is   16
    5           squared  is   25
```

iner و append عملیاتی جدید را به Tcl اضافه نمی کنند زمانی که نتایج حاصل از این فرمان ها از راه های دیگر قابل حصول باشد. اما انیدو فرمان راه های ساده ای را برای عملیات متداول ارائه می دهند. بعلاوه append طوری پیاده سازی شده است که از کپی کردن کاراکتر اجتناب می کند. اگر شما نیاز دارید یک

رشته بزرگ را از ترکیب تکه هایی دیگر تشکیل دهید می توانید به صورت کاراکتر از فرمان زیر استفاده کنید:

append x $piece

به جای اینکه از فرمان Set x "xPiece" استفاده کنید.

۷-٤) مروری بر امکانات دیگر متغیرها:

Tcl تعداد دیگری از فرمان ها را برای دستکاری متغیرها فراهم می کند. این فرمان ها به طور کامل وقتی شما چیزهای بیشتری راجع به زبان Tcl آموختید معرفی خواهند شد. اما این بخش مروری کوتاه بعضی از امکانات داشت. فرمان Trace می تواند برای نایش متغیری که یک اسکریپت Tcl با فراخوانی set مقداری را به آن نسبت داده یا آن را خوانده یا با فرمان unset حذف کرده است به کار رود. ردیابی متغیر گاهی اوقات برای خطایابی سودمند است و به شما اجازه می دهد متغیرهای فقط خواندنی را ایجاد کنید.

شما می توانید از ردیابی برای انتشار استفاده کنید. به طوری که برای مثال یک پایگاه داده یا صفحه نمایش با هر تغییر مقدار متغیر به روزرسانی می شوند. فرمان array می تواند برای فهمیدن نام تمام عناصر آرایه استفاده شود. با استفاده از فرمان info فهمیدن اینکه چه متغیرهایی موجودند امکان پذیر خواهد بود. فرمان

های global , upvar می توانند برای دسترسی به متغیرها خارج از حوزه محلی تعریف شده توسط پروسی جرها به کار روند.

فصل ۵

عبارات:

عبارت مقادیر (عملوندها) را با عملگرها باری تولید مقادیر جدید ترکیب می کنند. باری مثال، عبارت "2+4" در عملوند را شامل می شود، "4" , "2" و یک عملگر + که حاصل از ارزیابی این عبارت " ، " می شود. خیلی از فرمان های Tel یک آرگومان یا بیشتر از یک آرگومان نشان را یک عبارت تشکیل می دهد. مثالی از ساده ترین این فرمانها، فرمان expr است که آرگومانهایش را به عنوان یک عبارت مورد ارزیابی قرار می دهد و نتیجه را به صورت یک رشته بر می گرداند:

```
expr  (8 + 4) * 6.2
⇒ 74 .4
```

if مثال دیگری می باشد که اولین آرگومانش را به عنوان یک عبارت مورد ارزیابی قرار می دهد. و از نتیجه برای تعیین اینکه آیا آرگومان دوم بعنوان یک اسکریپت Tel مورد ارزیابی قرار بگیرد یا نه استفاده می کند:

if $ x < 2 then {set x 2}'

این فصل از فرمان expr برای تمام مثالهایش استفاده می کند. نکات دستوری مشابه، جانشینی و ارزیابی قوانین برای تمام استفاده ها از عبارت نیز به کار برده می شود. جدول ۱-۵ را برای خلاصه ای از فرمان expr ببنید.

۱-۵) عملوندهای عددی:

عملوندهای عددی عبارت معمولا اعداد صحیح یا اعداد حقیقی می باشند. اعداد صحیح (اینتی جر) معمولا به صورت دهدهی نمایش داده می شوند. اما اگر کاراکتر اول ۰ (صفر) باشد پس عدد در مبنای 8 می باشد و اگر دو کاراکتر اول (ox) باشند. سپس این عدد به صورت شانزدهی (در مبنای ۱۶) خوانده می شود. برای مثال ۳۳۵ یک عدد دهدهی می باشد، 0517 یک عدد اکتال (در مبنای ۸) با همان مقدار می باشد و ox14f یک عدد شانزدهی (در مبنای شانزده) با همان مقدار میباشد. 092 یک مقدار صحیح معتبر نمی باشد:

- باعث می شود این عدد در مبنای ۸ خوانده شود اما عدد ۹ و یک کاراکتر معتبر در مبنای ۸ نیست. عملوندهای حقیقی ممکن است برای استفاده بیشتر فرمانهای تعریف شده در ANSIC مثل مثال زیر مشخص شده باشد:

2 – 1

3.91e + 16

634

3.

یادداشت: این فرمان های مشابه نه فقط در عبارات بلکه درهر جایی در Tcl مـورد نیاز باشند. مجازند که استفاده شوند. عملوندهای عبارات می توانند رشته های غیـر عددی باشند. عملوندهای رشته ای در بخش ۵۵ بحث می شود.

Expr arg? arg arg
تمام مقادیر arg را با هم ملحق می کند (با فضاهای خـالی بـین آن) و نتـیج را بـه عنوان یک عبارت ارزیابی می کند و یک رشـته را مطـابق مقـدار عبـارت بـر مـی گرداند.

جدول ۱-۵) خلاصه ای از فرمان expr

۲-۵) عملگرها و تقدم ها:

جدول ۲-۵ همه عملگرهایی را که Tcl در عبارت پشتیبانی می کند. لیسـت کـرده است. این عملگرها شبیه عملگرهای عبارات در ANSIC می باشند. گروه هـایی از عملگرها را که حاوی حق تقدم یکسانی هستند خطوط افقی از هم جدا می کنـد و عملگرهایی با تقدم بالای جدول قرار می گیرند و عملگرهای با تقدم پایین تـر در انتهای جدول قرار می گیرند. برای مثـال 7>2 *4 یـک مقـدار صـفر (۰) بـر مـی

گرداند. زیرا عملگر * نسبت به عملگر < تقدم بالاتری دارد. بجز موارد آشکار و ساده بهتر است برای نشان دادن و گروه بندی تقدم عملگردها از پرانتز استفاده کنیم این روش از بروز خطا توسط شما یا کسی که این برنامه را اصلاح می کند جلوگیری می کند. عملگرهایی که از نظر تقدم در یک گروه قرار می گیرند. از چپ به راست مقدم ترند. برای مثال 3-4-14 مثل عبارت 3-(4-10) ارزیابی می شود. و نتیجه حاصل عدد ۳ می باشد.

۱-۲-۵)عملگرهای حسابی

عبارت Tcl از عملگرهای حسابی مثل 1+, - ,* ,/ پشتیبانی می کند. عملگر – ممکن است بعنوانع ملگرد دودویی باری تفریق مثل ۲-٤ یایک عملگر یگانه باری متنی کردن مثل (6*$i)- به کار گرفته شود. عملگر / نتیجه اش رااگر هر دو عملوند صحیح (اینتی جر) باشند به یک عدد صحیح گرد می کند.

عملگر ٪ یک عملگر قدر مطلق است. نتیجه این عملگر باقیمانده تقسیم عملوند اول بر عملوند دوم می باشد. برای ٪ هر دو عملوند باید صحیح (اینتی جر) باشند.

یادداشت: عملگردهای / و ٪ رفتار سازگارتری در Tcl نسبت به ANSIC دارند. در Tcl باقمیانده همیشه مثبت است و قدر مطلق آن کمتر از مقدار قدر مطلق مقسوم علیه می باشد. ANSIC فقط خصوصیات دوم راتضمین می کند. Tcl و

ANSIC در هر دو خارج قسمت این خاصیت را دارند که برای تمام x و y ها داریم.

$$(x/y) *y + x\% y = x$$

جدول صفحه ٤٥

جدول ۲-۵) خلاصه ای از عملگرهای مجاز در عبارات Tcl. تمام این عملگرها رفتاری مشابه به عملگرهای ANSIC دارند به جز اینکه بعضی از آنها می توانند عملورند های رشته ای داشته باشند. گروه هایی از عملگرها که در بین خطوط افقی قرار گرفته اند دارای تقدم یکسانی هستند. گروه های بالاتر تقدم بالاتری دارند.

۲-۲-۵) عملگردهای رابطه ای:

عملگرهای کوچکتر <، کوچکتر مساوی =< ، بزرگتر مساوی = >، مساوی == و نامساوی = ! برای مقایسه در مقدار به کار می روند. هر عملگری یک نتیجه (درست) تولید می کند. اگر عملوندهایی که در دو طرف عملگر قرار گرفته اند به همراه آن عملگر یک عبارت درست را تشکیل دهند. و در غیر این صورت یک مقدار صفر (نادرست) برگردانده می شد.

۳-۲-۵) عملگرهای منطقی

عملگرهای منطقی &&, 11, !, به طور نمونه برای ترکیب نتیجه عملگرهای رابطه ای به عنوان یک عبارت استفاده می شوند:

($x > 4) $$ ($x < 10)

هر عملگری یک نتیجه ۰ یا ۱ تولید می کند. && («و» منطقی) یک عدد ۱ به عنوان نتیجه تولید می کند. اگر هر دو عملوندش غیر صفر باشند، ۱۱ («یا» منطقی) یک عدد ۱ به عنوان نتیجه تولید می کند اگر یکی از عملوندهایش غیر صفر باشند»! («نه» منطقی) یک عدد ۱ به عنوان نتیجه بر می گرداند. اگر تنها عملوندش صفر باشد در Tcl مثل ANSIC، یک مقدار صفر به عنوان نادرست تلقی می شود. و هر مقدار دیگری بجز صفر یک مقدار درست تلقی می شود. هر وقت Tcl یک مقدار درست/ نادرست تولید کند از یک برای درست و از صفر برای نادرست استفاده می شود.

٤-۲-۵) عملگرهای منطق بینی:

Tcl شش عملگر برای دستکاری بیتهای اعداد صحیح پیش بینی کرده است. &, |, ^, ~, <<, >>. عملوندهای این عملگرها باید اعداد صحیح (اینتجر) باشند. عملگرهای ^, |, & عمل and «و»، or «یا» و exelusive or «یا بی انحصاری» را روی بیتها انجام می دهد. هر بیت نتیجه از تاثیر این عملگرها از

چپ به رسات بر روی بیت های نظیر به نظیر عملوند ها به دست می آیـد. توجـه کنید که & و | همیشه نتایج مشابهی با عملگر های && و || در بر ندارند:

Expr 8 && 2

$\Rightarrow 1$

expr 8 & 2

$\Rightarrow 0$

عملگرهای << و >> از عملوندهای سمت راست خود برای تعداد انتقال بیت هـا به سمت چپ یا راست استفاده می کنند. و نتیجه حاصل از آنها عدد سـمت چـپ می باشد که بیتهای آن به تعداد عدد سمت راست به چپ یا راست انتقـال (shift) داده شده است. در طی انتقال به چپ به جای بیت های کم ارزشتر صـفر وارد مـی شود. در انتقال به راست (انتقال به راست حسابی) همیشه برای اعداد مثبـت بـرای بیتهای انتقال داده شده صفر و برای اعداد منفی یک وارد می شـود. ایـن رفتـار بـا انتقال به راست (Right shift) در ANSIC که به ماشین بسـتگی دارد متفـاوت است. عملگر ~ (مکمل ۱) فقط یک عملوند دارد. و تمام بیت های عملوند خـود را معکوس کرده و حاصل را به عنوان نتیجه بر می گرداند. یعنی صـفرها را تبـدیل به یک و یک ها را تبدیل به صفر می کند.

۵-۲-۵) انتخاب عملگر:

عملگر سه گانه: ؟ ممکن است برای انتخاب یکی ا ز دو نتیجه به کار رود:

```
expr {C$Q < $Q : $b}
```

این عبارت از بنی مقادیر $a , $b مقدار کوچکتر را بر می گرداند. عملگر انتخاب اولین عملوند را از قطر درست یا غلط بودن بررسی می کند. اگر درست باشد (غیر صفر) سپس آرگومانی که بعد از ؟ قرار دارد به عنوان نتیجه بر گردانده می شود، اگر اولین عملوند نادرست (صفر) باشد سپس عملوند سوم بعنوان نتیجه برگشت داده می شود. فقط یک از عملوندهای دوم و سوم ارزیابی می شوند و نتیجه حاصل خواهند بود.

۵- توابع ریاضی:

عبارت Tcl از تعدادی از توابع ریاضی مثل sin , exp پشتیبانی می کنند. توابع ریاضی با علائم عملیاتی استاندارد فراخوانی می شوند:

```
expr 2* sin ($x)
expr hypot ($x , $y) + &z
```

آرگومانهای توابع ریاضی می توانند عبارات دلخواهی باشند، و آرگومان های چند تایی با کاما از یکدیگر جدا می شوند. لیست توابع توکار را در جدول ۳-۵ مشاهده کنید.:

جدول

٤-۵) جانشینی ها:

جانشینی ها در مورد عملوند ها می توانند به دو روش اتفاق بیفتند. روش اول براساس مکانیسم طبیعی پارس Tcl مثل فرمان زیر می باشد:

```
expr    2*sin ($x)
```

در این مورد پارسی Tcl مقدار متغیر x را قبل از اجرای فرمان جایگزین آن می کند. بنابراین اولین آرگومانهای expr مقداری مثل "2*sin (0 , 8)" خواهد داشت. دومین راه اجرای جانشینی فرمان در عبارتی است که مورد ارزیابی قرار می گیرد. برای مثال فرمان زیر را ملاحظه کنید:

```
expr   {2 * Sin ($x)}
```

در این مورد آکولادها مانع از جانشینی متغیر x توسط پارسی Tcl می شوند، بنابراین آرگومان علامت expr, "2*sin ($x)" می باشند. وقتی که ارزیاب (evaluator) عبارت با علامت $ مواجه می شود. خودش جانشینی متغیر را انجام می دهد. و مقدار متغیر x به عنوان آرگومان تابع sin مورد استفاده قرار می گیرد. داشتن دو لایه از جانشینی باعث بروز اختلاف برای فرمان expr نمی شود. اما این مسأله برای فرمان های دیگر مثل while که یک عبارت را به طور مکرر ارزیابی می کنند و در هر بار نتیجه متفاوتی به دست می آورند خیلی مهم و حیاتی است. برای مثال اسکریپت زیر را که یک عدد پایه را به توان می رساند ملاحظه کنید:

```
set result 1
while  {$Power > 0} {
set result [expr $result * $base]
iner power –1
}
```

عبارت "$Power" در شروع هر تکرار توسط حلقه while ارزیابی می شود. برای اینکه مشخص شود که آیا حلقه پایان یابد یا خیر. این نکته ضروری است که ارزیابی عبارت در هر بار مقدار جدید power را مورد استفاده قرار می دهد. اگر جانشینی متغیر زمانی که تجزیه و تحلیل فرمان while انجام می شد صورت گرفته بود. برای مثال "While $ Power> 0...." سپس آرگومان فرمان while مقدار ثابتی مثل "0<5" می شد و حلقه یا اجرا نمی شد و یا برای همیشه اجرای آن ادامه می یافت و هیچ گاه متوقف نمی شد.

جدول ص ٤٩

جدول ۳-۵) توابع ریاضی که در عبارت Tcl استفاده می شوند. در بیشتر موارد توابع رفترای مشابه کتابخانه پروسی جرهای ANSIC با همان نام دارند.

یادداشت: وقتی که ارزیاب عبارت (expression evalautary) جانشینی متغیر یا فرمان را انجام می دهد. مقدار جانشین دشه باید یک عدد صحیح یا حقیقی باشد. (یا رشته ای که در زیر توضیح داده می شود).

این مقدار نمی تواند یک عبارت اختیاری بادش.

۵-۵) دستکاری رشته ها:

برخلاف عبارت AVSIC، عبارات Tcl چند عملگر رشته ای ساده دارند مثل فرمان زیر:

```
if {$x = = “New York”} {
   }
```

در این مثال ارزیابی کننده مقدار متغیر x را با رشته “New York” مقایسه می کند. اگر مقدار متغیر x هر رشته “New York” برابر باشند بدنه if اجرا خواهد شد. برای مشخص کردن یک عملوند رشته ای باید آن را درون جفت کوتیشن یا آکولاد قرار دهید و یا از جانشینی متغیر یا جانشینی فرمان استفاده کنید.

قرار گرفتن عبارت مثال بالا در آکولاد مسئله مهمی است. برای اینکه ارزیابی کننده عبارت د رمورد متغیر x مقدار آن را جانشنی می کند، اگر از آکولاد ها صرف نظر می کردیم. آرگومان if رشته ای مثل رشته “New York” = los Angeles می شد و پارسی قادر به تجزیه و تحلیل “tas” نبود (این یک عدد نیست) هم تابع نیز به نظر نمی رسد، بعنوان یک رشته نیز می تواند تفسیر شود چون حدود آن تعیین نشده است. بنابراین یک خطای دستوری رخ می داد.

اگر رشته ای درون جفت کوتیشن قرار گیرد سپس ارزیابی کننده عبارت جانشینی فرمان و متغیر بک اسلش را بر روی کاراکترهایی که درون کوتیشن قرار گرفته اند انجام می دهد. اگر رشته ای درون آکولاد قارر گیرد هیچ جانشینی ای بر روی آن انجام نمی شود. آکولادها برای رشته ها به صورت لانه ای (تو در تو) قرار می گیرد. همانگونه که کلمات یک فرمان به صورت تو در تو قرار می گیرند.

عملگرهایی که عملوند هایشان می توانند یک رشته باشد عبارت اند از $<$, $>$, $<=$, $>=$, $==$, $!=$ و دیگر عملگرها باید عملوند عددی داشته باشند. عملگرهایی مثل $<$ مثل رشته ها را به صورت لغوی با تابع کتابخانه ای سیستمی stremp مقایسه می کنند، مرتب سازی ممکن است از سیستمی به سیستم دیگر تغییر کند.

۵) انواع و تبدیل ها:

Tcl تا آنجا که میسر باشد عبارت را به صورت عددی مورد ارزیابی قرار می دهد. عملگرهای رشته ای فقط در مورد عملگرهای رابهط ای و فقط وقتی یکی یا هر دو عملوند غیر عددی باشند قابل اجرا خواهند بود. بیشتر عملگرها عملوندی صحیح و حقیقی را مجاز می دانند. اما تعدادی از انها مثل $<<$, $\&$ فقط بر روی عملوندهای صحیح قابل اجرا خواهند بود. اگر عملوندهای یک عملگر دارای انواع

متفاوتی باشند Tcl به طور خودکار یکی از آنها را به نوع دیگر تغییر می دهد. اگر یکی صحیح (اینتی جر) دیگری حقیقی (Real) باشد. عملوند صحیح به حقیقی تبدیل می شود. اگر یک عملوند یک رشته غیر عدید باشد و دیگری یک عدد صحیح یا حقیقی آن عدد صحیح یا حقیقی به یک رشته تبدیل می شود. نتیجه یک عملگر همیشه با عملوندهایش ممنوع است بجز عملگرهای رابطه ای ک یک عدد صحیح ۰ یا ۱ را به عنوان نتیجه بر می گردانند. شما می توانید از توابع ریاضی مثل dowble برای تبدیل کی عدد صحیح به حقیقی و از int , round برای تبدیل یک مقدار حقیقی به صحیح با گرد کردن و بریدن استفاده مکنید.

۷-۵) دقت:

در هنگام ارزیابی، Tcl اعداد صحیح را با نوع اینتی جر C که در بیشتر ماشین ها با دقتی ۳۲ بیت نمایش داده می شود، نشان می دهد. اعداد حقیقی با نوع دابل C (double) نشان داده می شود که با مقدار ٦٤ بیتی برای اعداد ممیز شناور قابل نمایش خواهد بود.

اعداد به فرم داخلی در مدت ارزیابی یک عبارت نگهداری می شود و فقط وقتی لازم باشد به رشته تبدیل می شوند مثل زمانی که فرمان expr نتیجه اش را بر می گرداند. اعداد صحیح تبدیل به رشته های ده دهی علامت گذاری می شوند. بدون

اینکه دقت خود را از دست بدهند. وقتی که یک مقدار حقیقی به یک رشته تبدیل می شود. فقط شش رقم با معنی آن به صورت پیش فرض باقی می ماند:

expr 1.111111111 + 1.11111111

⇒ 2.22222

اگر شما بخواهید اعداد با معنی بیشتری وقتی که یک عدد حقیقی به یک رشته تبدیل می شود باقی بماند می توانید تعداد ارقام با معنی که می خواهید را درون متغیر Tcl-Precision قرار دهید.

Set Tcl-Precision 12

Expr 1.111111111 + 1.11111111

⇒ 2.22222222

متغیر Tcl-Precision نه فقط برای فرمان expr بلکه هر جایی که برنامه کاربردی Tcl یک عدد حقیقی را به رشته تبیدیل می کند استفاده می شود.

یادداشت: Tcl-Precision را در ماشینی که از استاندارد ممیز شناور IEEE استفاده یم کند برابر ۱۷ قرار دهید شما تضمین می کنید که در تبدیل یک نوع به رشته هیچ نوع اطلاعاتی از بین نمی رود.

- اگر حاصل یک عبارت به یک رشته برگردانده شود و سپس در عبـارت دیگـری مورد استفاده قرار رگیرد. فرم داخلی آن قبل از تبدیل به رشته با فرم داخلی آن بعد از تبدیل به رشته یکسان خواهد بود.