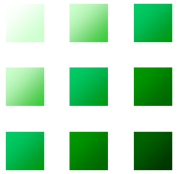


Tclについて

Tool Command Language

2013.8.2

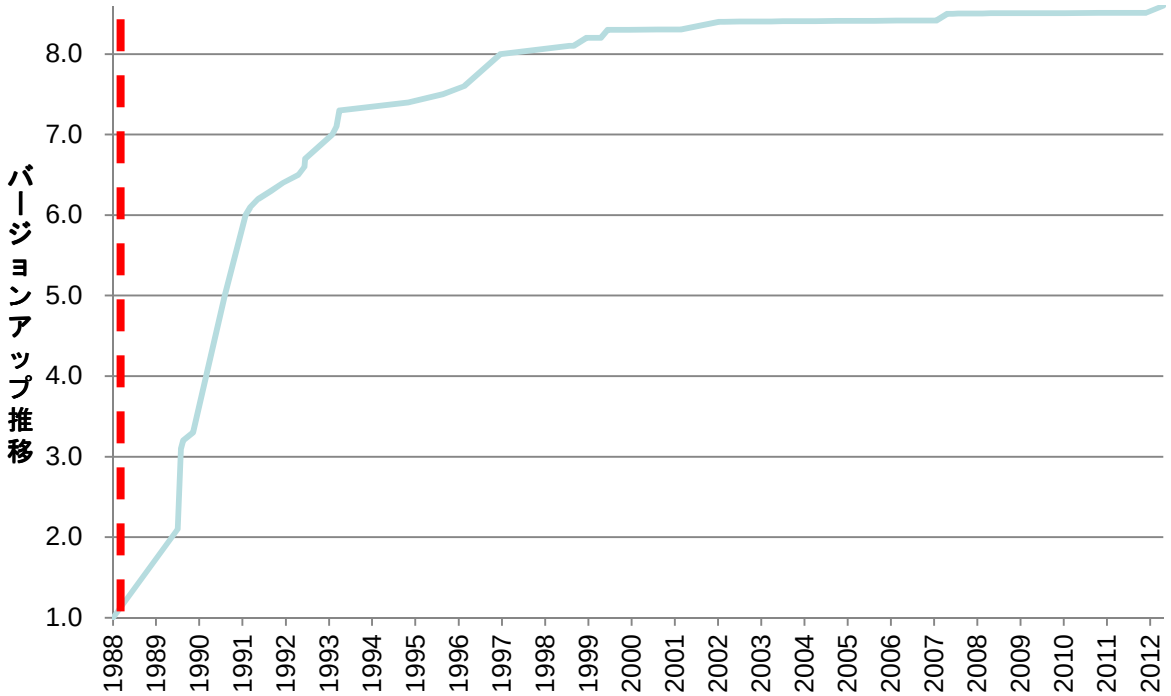
ニューラル株式会社

御品書き

- Tclの輝かしい歴史
- 基本的な構文
- 特徴的な機能

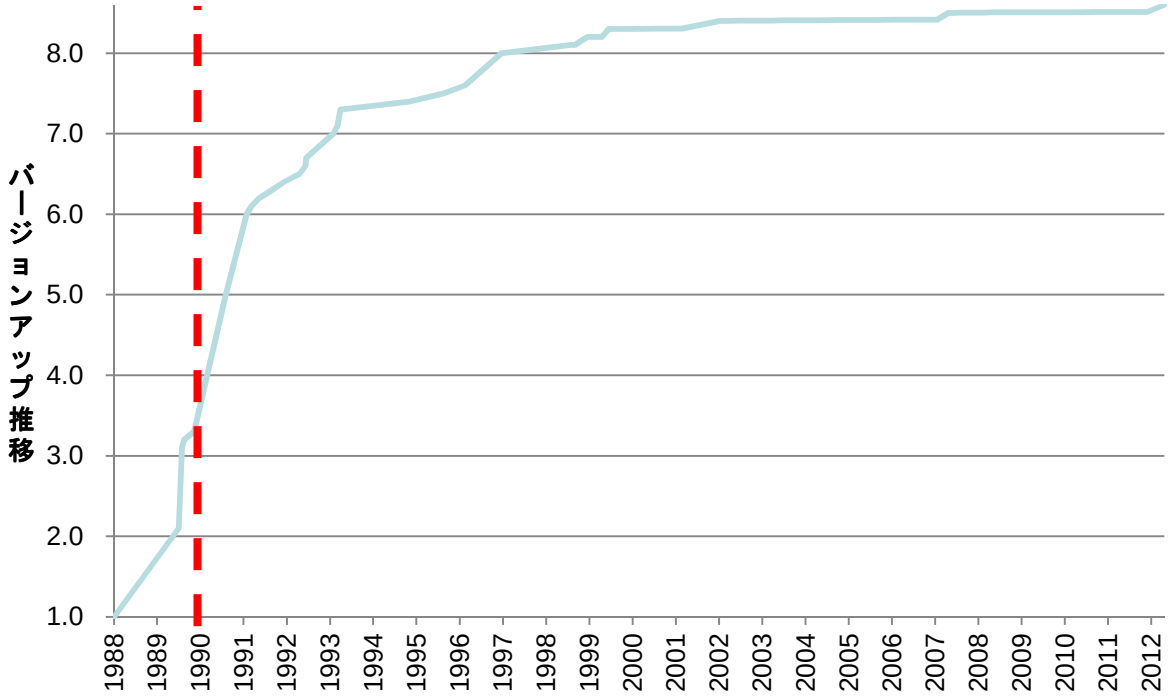
Tclの輝かしい歴史

- 1988年 : 誕生
  - オースターハウト博士により開発。
  - UNIXアプリケーション拡張用スクリプトの標準化を目指す。
  - TclはTool Command Languageの略。



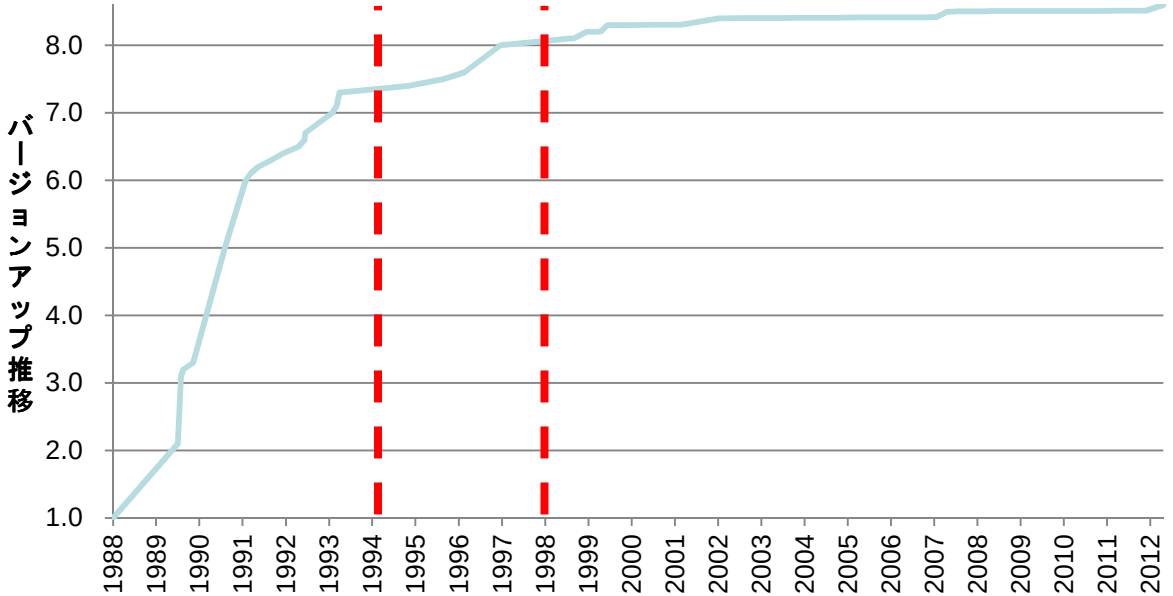
Tclの輝かしい歴史

- 1990年：Tkをバンドル
  - 容易にGUIを作成できるTkをバンドルし、一気に人気が出る。
  - 誕生当初の目的であるアプリケーション拡張言語としてではなく、Tkと合わせたTcl/Tkの形でGUI開発言語としての人気。
  - Tkはtoolkitの略。



Tclの輝かしい歴史

- 1994~1998年：サン・マイクロシステムズが開発
  - 作者がサンに勤めたこの期間は同社が開発。
  - Webクライアント環境制覇の戦略の一つとして精力的に開発された。
    - UNICODE化、バイトコンパイル、Tclet、など。
  - しかし他の技術と競合し人気を拡大させることはできなかった。
  - また、デスクトップ環境でもVisual Basicによる開発が増え、Tcl/Tkは徐々に使われなくなる。



Tclの輝かしい歴史

一方その頃
日本では・・・

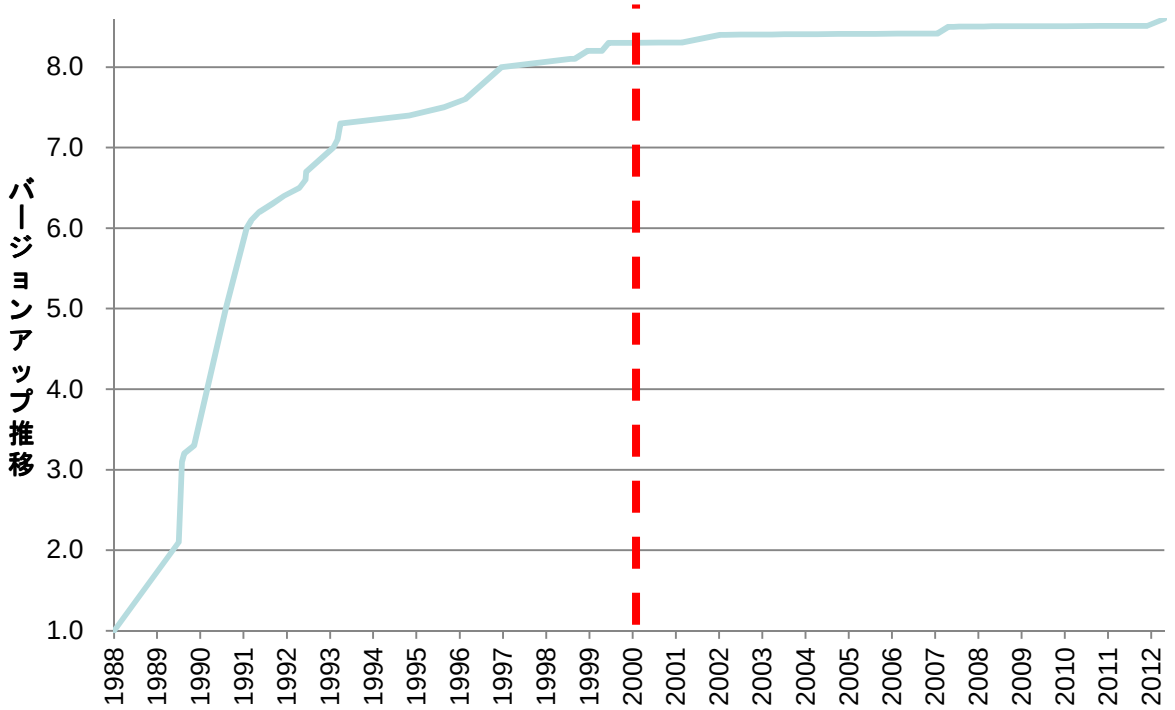
Tclの輝かしい歴史

TKサウンドが流行りましたが・・・

Tcl/Tkとは全く関係ありません。

Tclの輝かしい歴史

- 2000年～:オープンソース開発
  - オープンソースに開発の場を移す。
  - TkはPerl、Python、Rubyからも使えるように移植される。
  - Tclは本格的に使われなくなる。



Tclの輝かしい歴史

- 2013年7月17日：2chに3スレ目が立つ
  - 十年にわたり熱い議論が交わされていたようです。

●●●●TCL/TKなら俺に聞け 3●●●●

2 ：**デフォルトの名無しさん** ：sage ：2013/07/17(水) 18:28:57.71
前々スレは4年半も持ったので、前スレも2012年くらいまではいけるという予言
↓
前スレは6年2か月も持ったので、このスレも2020年くらいまではいけるという予言

Tclの輝かしい歴史

- 2013年7月現在のTclの人気順位は36位
  - TIOBE Programming Community Index for July 2013 より

順位	プログラミング言語	インデックス
1	C	17.63%
2	Java	15.91%
3	Objective-C	10.25%
4	C++	8.75%
5	PHP	7.19%
6	C#	6.21%
7	(Visual) Basic	4.34%
8	Python	4.04%
9	Perl	2.15%
10	JavaScript	1.84%
	～	
35	Scratch	0.30%
36	Tcl	0.25%
37	F#	0.24%
	～	
50	RPG (OS/400)	0.16%

基本的な構文

- コマンド行を逐次実行するシンプルな構文
- コマンド行は文字列のリスト
  - リスト先頭の文字列をコマンドとする。
  - コマンドより後ろの文字列リストを引数とする。
- コマンド行はセミコロンか改行で区切られる

```
puts "Hello world!"; puts “Hello newral!”

# 構文上に型は存在せず全て文字列。“～”で括らなくても文字列となる。
set a Tcl最高！

# $+変数名で設定されている文字列が展開される。スペースを含む文字列は“～”で括る。
puts “$a と思っていた時期が私にもあります“; # → Tcl最高！と思っていた時期が私にもあります

# {～}で括ると変数は展開されない。
puts {$a と思っていた時期が私にもあります}; # → $a と思っていた時期が私にもあります
```

基本的な構文

- 制御構文にあたるものはコマンドで実装されている
  - if, for, foreach, while, switch, break, continue

```
foreach {i j} {1 2 3 4} {
        puts "$i * $j = [expr $i * $j]"
}
```

構文上の解釈	コマンド行の各文字列
コマンド名	foreach
引数1	{i j}
引数2	{1 2 3 4}

引数3	{
	puts "$i * $j = [expr $i * $j]"
	}

特徴的な機能

- 異なるスタックフレームでスクリプトを実行できる
  - いわゆるクロージャのようなものを実現するために必要。

```
# 引数で渡された処理を実行し、その処理時間を測る。
proc stopwatch {script} {
    set start [clock clicks -milliseconds]
    eval $script
    set end   [clock clicks -milliseconds]
    puts "経過時間: [expr $end - $start]ms“
}
# 上記のプロシージャを使い合計計算の処理時間を測る。
proc main {} {
    set count 100
    set sum 0
    stopwatch {
        for {set i 1} {$i <= $count} {incr i} { # エラー！変数 $count を参照できない
            set sum [expr $sum + $i]
        }
    }
}
```

特徴的な機能

- uplevelコマンドにより1つ上のスタックで処理を実行
  - 呼び出し元のスコープでやりたい放題もできる。

```
# 引数で渡された処理を実行し、その処理時間を測る。
proc stopwatch {script} {
    set start [clock clicks -milliseconds]
    uplevel 1 $script
    set end   [clock clicks -milliseconds]
    puts "経過時間: [expr $end - $start]ms“
}
# 上記のプロシージャを使い合計計算の処理時間を測る。
proc main {} {
    set count 100
    set sum 0
    stopwatch {
        for {set i 1} {$i <= $count} {incr i} {
            set sum [expr $sum + $i]
        }
    }
}
```

まとめ

まとめ

- Tclは構文が単純なため習得が容易。
- 独自の構文(もどき)を実装できる柔軟性。
- Tkが頑張ればTclはもう少し生き長らえる。