LTPL : Loki Text Processing Language Manual

Michael

2/7/22

Contents

1 Introduction

1.1 What Is This Exactly ?

LTPL is a declaritive text processing and manipulation utility that specializes in making common text processing tasks as easily and tersely as possible.

1.2 Who Is This For ?

This is a tool mainly for programmers and users who require ridgit formated text output. Specifically targeting those who would traditionally use awk to do so.

1.3 Why Does This Exist ?

I wanted a utility for text processing that was more flexible than Sed but more tearse than awk that that didnt contain the C-isms that was carried over as bagage from the Bell Labs Unix style thought process when designing awk.

1.4 What This Tool Is

1.4.1 It is a Utility For Text Manipulation

LTPL is more a utility along the lines of sed and awk than an actual language you would write in a file. It ment to be embedded in bash scripts and to be used either along side or be more versitle than more traditional tools.

1.5 What This Tool Is Not

1.5.1 It Is Not *Complete* Replacement For Awk.

I am only comparing my language periodically to awk as the closest parallel of the language that exists and is in relitive popular use today. I am aware that awk is a more traditional language with a much more familar syntax that is ment to write programs and scripts that may be run from a file or

piped in from another program. however the purpose of LTPL is explicitly for reading in and displaying data.

1.5.2 It Is Not A Scripting Language

This tool is not for all intensive purposes is not a scrihpting language nor should it ever be used as one. LTPL is a tearse and simple text processing utility and nothing more

1.6 Why The Cryptic Syntax

LTPL's syntax and grammar is very simple aiming and being simple to use in a terminal enviroment. With this consideration taken a APL or Regular expression like syntax was considerd appropriate for a speedy interactive usage that I was trying to achieve.

2 Essentals

There are four core concepts exibit themselves in LTPL. The language its self is rather simple which makes it a very powerful tool for parsing text.

- Reader Directives
- Actions
- Objects
- Regular Expressions

2.1 Reader Directives

2.1.1 What Is A Reader Directive

A Reader Directive is a command to the intepreter that directs the manner in which the input file will be parsed.

There are two main Reader Directive types in LTPL.

- Parsing Directives
- Range Specifiers

1. **Parsing Directives** Parsing Directives which controls how the file is able to be read. there are a variety of different ways text can be formatted. It is not always appropriate to read file left to right.

   and yes im sorry ltpl is 1 indexed throught the language, but there is a very good reason

Reader Directives	Parse Description	$FS Default	Starting Cursor Position
==	parse left to right	" " Space	(1,1)
||	column by column	"" Newline	(1,1)
^|	column by column	"" Newline	(1,n)
–	Read a single line	" " Space	(1,n)
<=	read right to left	" " Space	(n,1)

where n is the number of elements in that row or column

Here is an example on how reader directives can be used with some formated input

```
$ ls -l

drwxr-xr-x 2 user user  4096 Jan 20 19:42 Desktop
drwxr-xr-x 2 user user  4096 Feb  4 00:36 Documents
drwxr-xr-x 3 user user  4096 Feb  6 23:16 Downloads
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Music
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Pictures
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Public
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Templates
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Videos

$ ls -l | ltpl "==$5[*:1024][p]."
Output:

4194304
4194304
4194304
4194304
4194304
4194304
4194304
4194304
```

where we are getting the 5th element and multiplying it by 1024 and printing the output

Another way to achieve the same behavior but more efficently would be to do

```
$ ls -l

drwxr-xr-x 2 user user  4096 Jan 20 19:42 Desktop
drwxr-xr-x 2 user user  4096 Feb  4 00:36 Documents
drwxr-xr-x 3 user user  4096 Feb  6 23:16 Downloads
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Music
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Pictures
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Public
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Templates
drwxr-xr-x 2 user user  4096 Jan 20 19:42 Videos

$ ls -l | ltpl "||$5[*:1024][p]."

4194304
4194304
4194304
4194304
4194304
4194304
4194304
4194304
```

The only difference here from the example above is the way that the interpter reads the information. instead of reading every single field starting with *drwxr-xr-2* and ending when we find $5 which in this case is 4096. we can get entire columns of text just by reading by column.

Examples of the other reader directives being used can be found at .... (havent made a place for it yet)

2. **Range Specifiers**

   There are a lot of times we want to ommit certain places where we have junk in a file. By junk I dont really mean garbage in the sense that its not important but I mean that its not applicable for what we need.

you can achieve this by using Range Specifiers which controls what field the given lines of the input will be read.

A Position Specifier may be used in combitation with a Parser Directive to give more flexibilty to the user; detailing what subset of data of the input that will be read.

given the following syntax.

```
"||0,1"
```

which reads each column skipping the first column entirely

A practical application to get all of the numbers on line 5 would be

```
$ cat file.txt
John Doe
March 21st, 2022
John_The_Doe@hotmail.com (because I feel hotmail is funny)

10 20 30 40 50
hello good bye

$ ltpl file.txt "--5,5[p]"
10 20 30 40 50
```

where it reads only one singular line of text at line 5 and prints it

2.2 Objects

2.2.1 What Are Objects ?

Objects in ltpl really are one of the most complex things in the entire language. but even though they are complex they are also rather simple in structure. Objects are hiearchical and able to be indexed quite simply as well.

[.S a [.NP b c] d] [.S a [b c].NP d]

2.2.2 Object Primitives

Object Primitives are a what you would call literals in any other language but there is a quirk of all objects in ltpl. every object has children represented as numbers. By default you are adressing the children of $0 when refering to lines.

2.2.3 Fields And Implict Objects

1. Fields Fields are the way that LTPL treats columns of text that are seperated by the $FS implicit object.

2. Implicit Objects

Name	Description
$0	The 0th field refering to the whole line of text. That contains an object array o
$n	The $nth field that refers to an object that is broken up into fields by the $fs
$FS	the delimiting character(s) that designantes the seperation of new tokens by a u
$CL	the current line being read
$PL	previous line read
$NL	the next line to be read.
$NLR	the number of lines that have been read.
$NTR	the number of lines that need to be read.
$RED	An object
$GREEN	
$BLUE	
$BLACK	
$WHITE	
$CYAN	
$MAGENTA	
$YELLOW	
$PURPLE	
$PINK	
$ORANGE	

where n is the number of fields

2.2.4 User Defined Objects

Objects In LTPL are able to be created by referancing a nonexistant object by using the $. typing the example nonsensecal case "$bar" is a proper object declaration.

2.2.5 Assignement

Defining new variables along with reassigning existing ones are core parts of any programming language (except for the haskell purists out there) LTPL is no different but has a quirk to do so.

1. Examples LTPL example.txt "==$bar[10]."
   LTPL example.txt "==$foo."
   both of which are valid instances of objects where foo is assigned to an empty string by default and bar is assigned to 10

   $ ls

   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Desktop
   drwxr-xr-x 2 user user 4096 Feb 4 00:36 Documents
   drwxr-xr-x 3 user user 4096 Feb 6 23:16 Downloads
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Music
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Pictures
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Public
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Templates
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Videos

   ls -l | LTPL "==$3[~=$3]$foo[p].

   Output:

   drwxr-xr-x 2 user 4096 Jan 20 19:42 Desktop
   drwxr-xr-x 2 user 4096 Feb 4 00:36 Documents
   drwxr-xr-x 3 user 4096 Feb 6 23:16 Downloads
   drwxr-xr-x 2 user 4096 Jan 20 19:42 Music
   drwxr-xr-x 2 user 4096 Jan 20 19:42 Pictures
   drwxr-xr-x 2 user 4096 Jan 20 19:42 Public
   drwxr-xr-x 2 user 4096 Jan 20 19:42 Templates
   drwxr-xr-x 2 user 4096 Jan 20 19:42 Videos

2.2.6 Object Arrays And Subfields

Subfields and object arrays are more or less equvilent. with the only real difference is in what is being refered. to put it simply:

if it is a field it will be refered to as a subfield. if it is a user defined object it is refered to as an object array. *The encomapssing term between the two is object array*

1. Refering To Object Arrays Below is a dummy file with that we will parse.

Example.txt:
hello world this is a tjest.

if you notice there is a spelling error
you can fix such a minute error like so

LTPL Example.txt "–$6$2[~=$6$2]$6$1[»]$0[p]."
where $6 is the 6th field while refering to the 2nd object field. deleting the object in said field and moving the object on the left next to it over where the previous object resided

2.3 Actions

2.3.1 What Is An Action

An action is the primary enact changes to Objects. they are syntactically represented within [].

1. Example $ ls -l
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Desktop
   drwxr-xr-x 2 user user 4096 Feb 4 00:36 Documents
   drwxr-xr-x 3 user user 4096 Feb 6 23:16 Downloads
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Music
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Pictures
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Public
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Templates
   drwxr-xr-x 2 user user 4096 Jan 20 19:42 Videos

   ls -l | LTPL "==$6[p]"

   Output:
   Jan
   Feb
   Feb
   Jan
   Jan
   Jan
   Jan
   Jan

2.3.2 Actions Predefined

1. Output And More Output Keeping within the bound of the promise that this is not a scripting language There is no way to prompt users for input what so ever. In LTPL There a variety of different mechinisms that the user to write to a file of their choice.

   (a) Writing To Files What Would a text processing language be with out being able to save the manipulated text to files write to files.

2.3.3 Why Can't I Define My Own Actions

well there is a simple answer to that. LTPL is not a scripting language. If you feel you need to define your own actions to make a certain action easier. you should look at some other language. consider using AWK or perl. heck sed can be useful in some circumstances.

2.3.4 Possible Actions

Name	Symbolic Name	Description
print	p	Prints an object to stdout
write	w	writes objects to a file
filter	~	removes if condition is true
cast to type	->	given an object it converts it to the type of a given object
ternary	?	does the next action if true the other if false
italic	i	
bold	b	
underline	_	underlines an object
highlight	#	highlights an object
shift down line	VV	shfits an object down into the line below it.
shift up line	^^	
Swap lines		
Move Right	»	shifts an object right by one field replacing the object that inhabi
Move Left	«	shifts an object left by one field replacing the object that inhabite

3 Implentation Details

4 Examples

5 Benchmarks

ure Ideas

6 Grammar BNF