SCAWK DOC

Charles-Albert Lehalle

printed the July 6, 2001

Contents

1 Scawk

Scawk is a port of awk to scheme. The implementation of scheme used is siod, but it is planned to extend scawk to other scheme implementations.

The interressant point with siod is that a scheme file can be compiled into an executable one. So I use scawk as an exec file for WinNT (scawk.exe).

I tried to port all the classical functions of awk to scawk. In fact I ported the functions I usually used in awk. Feel free to port other functions. I will try to release new versions of scawk including new features form other people.

You can send me e-mails to lehalle@earthling.net.

Actually you can invoke scawk into a scheme file as : (process '(":f" "scawk-cmd-file.sc" "data-file.txt")) or using scawk.exe as : scawk.exe :f scawk-cmd-file.sc data-file.txt where scawk-cmd-file.sc contains the definition of a function *main* and optionaly *begin* and *end*.

2 Main functions

The function *begin* can be redefined it will be called before the opening of the target file

The fonction *main* can be define it will be called on each line of the target file some global variables will be usable

The function *end* can be redefined it will be called after the target file has been readen

3 Global variables

As for awk, each line is splitted into tokens using SPACE as separator

The global variable *nf* contains the number of fields

• `(\$ n)` returns the n-th element of the line ($ 0) is the complete line, and ($ n) is the nth token of the line

• The function `(form a b ...)`

The default output of display is stdout

4 Usefull classical Awk functions

• The `(split string separator)` function returns a list containing the string tokenized using the specified separator. The separator is a string and can have a length of more than 1.

```
(split "azerdytacvjhagygakj kjah iaug auhkj" "a")
> ("" "zerdyt" "cvjh" "gyg" "kj kj" "h i" "ug " "uhkj")
(split "azerdytacvjhazegygakj kzejah iaug auhkj" "ze")
> ("a" "rdytacvjha" "gygakj k" "jah iaug auhkj")
```

- The `(substr string begin end)` returns the subtring of string begining at begin and ending at end

```
(substr "garzol" 2 3)
> "r"
(substr "garzol" 0 2)
> "ga"
(substr "garzol" 0 20)
ERROR: bad end index (see errobj)
(substr "garzol" -1 2)
ERROR: bad start index (see errobj)
```

- The `(index string key)` returns the index of key in the specified string. If it does not exist, return #false

```
(index "garbure" "rb")
> 2
```

- The `(decompose string key)` function returns a list of two elements : the substring of the specified string before the key, and the substring after the key. If the key is not in the string, the second element is #false.

```
(decompose "garbure" "rb")
> ("ga" "ure")
```

- The `(sub string key1 key2)` function substitutes the first occurence of key1 by key2 into the specified string.

```
(sub "barzol" "zz" "XXX")
> "barXXXl"
```

- The `(char-at string n)` returns the nthe char of the string. The first char has index 1.

```
(char-at "garzol" 2)
> "a"
```

5 Other usefull functions

They are functions I usally used in awk (I have a personal library) with such functions for awk. HEre I decided to include them into scawk. Enjoy using them.

- The function `(between string k-beg k-end)` returns the substring of string that is between k-beg and k-end. If k-beg or k-end is not into string, returns #false.

```
(between "garbure garzol barbure" "bu" "zo")
> "re gar"
```

- The function `(substring+ string key)` returns the part of string that is after key. If key is not in the string, returns #false.

```
(substring+ "garzol" "rz")
> "ol"
```

- The function `(substring- string key)` returns the part of string that is before key. See substring+.

- The function `(without- string key)` remove all the occurences of key at the begining of the string. Usefull to remove useless spaces.

```
(without- "    rtyfygv" " ")
> "rtyfygv"
```

- `(string-revert string)` revert the string.

```
(string-revert "charles")
> "selrahc"
```

- The function `(without- string key)` remove all the occurences of key at the end of the string. See without+.

- The function `(without string key)` remove all the occurences of key at the begining and at the end of the string.

- The function `(without-tags string)` remove XML tags from string. Some implemntation of XML-DOM for scheme exist ; use it for more complex operations on XML files. It is just usefull to convert a HTML file into a TXT one.

```
(without-tags "this is <B>a test</B>..")
> "this is a test.."
```

- the function `(display string [filename|port])` is like the awk print function. The string is written at stdout if there is not a second argument. If the second argument is present it is : a filename (string) then the string is append to a file with this name (which is created if it does not exist) ; or a port name that has been previously opend by the siod scheme (fopen fname mode) function.

- The function `(string-length string)` repace the length function of awk because length applies to lists here. I plan to implement a polymorph version of length but I do not found the time to do it actually.

- The function `(string->list string)` is used to convert a string into a list of string of length 1 (as characters).

Index