psilo

a parallel, streaming, iterative list operation language for writing interesting programs. View it on GitHub.

What is psilo?

psilo will be a language for writing software to process large streams of data as efficiently as possible. It is list processing taken to its logical conclusion, augmented with strong static types and compile time optimizations.

It is also nowhere close to being finished; it's merely an educational experiment for myself.

Technical Features (planned):

- No run-time garbage collection necessary owing to uniqueness types
- Static typing for compile-time verification and optimization
- Malleable syntax with macros
- Dead-simple parallelism via pipelines and special list types
- Monadic continuations and iteratee composition made dead simple
- Orthogonal core syntax and semantics for your performance and my sanity

Philosophy:

- All programming is manipulating languages.
- Types define grammars; functions define parsers.
- The earlier a question may be answered, the better.
- If the computer can do it, it should.

Status

Psilo is still being designed. I have written a really simple evaluator for prototyping and experimenting with the language which is actively being developed.

While not production quality, the simple interpreter might be of educational value.

Here is some code the interpreter runs right now:

```
(let
  ((cons (\ (x y)                ; Prepend an item to a list
           (\ (f g) (f x y))))
   (nil  (\ ()                   ; Create an empty list
           (\ (f g) (g))))
   (car  (\ (xs)                 ; Return head of a list
           (xs (\ (x y) x))))
   (cdr  (\ (xs)                 ; Return tail of a list
           (xs (\ (x y) y))))
   (length (\ (xs)               ; Calculate the length of a list
     (let
       ((length-helper (\ (xs n)
          (xs (\ (y ys) (length-helper ys (+ n 1)))
              (\ ()     n)))))
       (length-helper xs 0)))))
  (let
     ; Sample list of 3 numbers
    ((list-1 (cons 1 (cons 2 (cons 3 (nil))))))
    (length list-1)))
```

How to build

You need the Glasgow Haskell Compiler and a number of libraries; I suggest starting off with the Haskell platform.

Clone the repository:

```
git clone https://github.com/gatlin/psilo
```

Set up a cabal sandbox:

```
cabal sandbox init
cabal configure
cabal install --only-dependencies
```

Then make with:

```
make
```

And return to the Edenic, pre-build post-checkout status of the code with

```
make clean
```

Questions / comments / hate mail

Use the Issues feature of GitHub or email me: gatlin@niltag.net.