FABRIC ENGINE®

It's Time For Real–Time

KL Feature Overview

FABRIC ENGINE

Kernel Language (KL)

- Pronounced ‘kale’
- Very small language scope
- Syntactically based on C/C++, but simpler
- Powerful type system
- Supports multithreading
- Compiles to optimized machine code
- Not interpreted

KL Tool

- Command line tool
- Similar to python executable
- Good for learning and debugging

KL Operators – Hello World

File reference: 01_kl/01_helloworld.kl

- The *operator* statement is used for main entry functions.
- Curly braces, *{* and *}*, are used to define code execution blocks.
- The KL tool executes an operator called *entry*.
- The *report* statement is used to print directly to the host application.

KL Variables

File reference: 01_kl/02_variables.kl

- Variables are defined by their type.
- Basic types for example are *Boolean*, *Integer*, *Scalar*, and *String*.
- Numeric basic types support the standard arithmetic operators (+, −, *, /, etc).

KL Functions

File reference: 01_kl/03_functions.kl

- The *function* statement is used define additional functions or methods on data structures.
- Function parameters can be flagged as *in* or *io*.
- Functions can return values. If no return value is specified the function can be considered *void*.

KL Conditions

File reference: 01_kl/04_conditions.kl

- The conditional language features match most other programming languages (such as JavaScript).
- The *if* statement can be used for conditional code blocks. The *else* statement can be used for the second case of a condition.
- The *switch* statement can be used for a long list of cases to reduce the code.

KL Loops

File reference: 01_kl/05_loops.kl

- Loops can be used to perform iterative tasks.
- KL provides a *for* loop as well as a *while* loop, so counted vs. condition based strategies.

KL Arrays

File reference: 01_kl/06_arrays.kl, 01_kl/07_unguarded.kl

- KL arrays represent a list of values.
- Arrays are defined by the [] suffix.
- Brackets can also be used to access elements.
- Arrays are reference counted.
- Arrays can be concatenated using + and +=.
- Array element access is guarded, but can be unguarded for improved performance.

KL Dictionaries

File reference: 01_kl/08_dictionaries.kl

- KL dictionaries represent a map from one type (key) to another (value).
- Dictionary are defined by the *[key]* suffix.
- Brackets can also be used to access elements.
- Dictionaries are also reference counted, like Arrays.

KL Structs

File reference: 01_kl/09_structs.kl

- KL provides a rich type system.
- Structs represent simple, nestable data structures.
- Structs can provide methods.
- Structs are always copied.
- All (!) types used within any Fabric Engine product are purely implemented in KL, there are no black boxes.

KL Objects

File reference: 01_kl/10_objects.kl

- Objects are similar to structs.
- Objects are reference counted.
- Objects need to be initialized.
- Objects typically are used for heavy data structures which introduce latency for copy operations.

Requiring KL types

File reference: 01_kl/11_require.kl

- KL types can be required into any KL file.
- The *require* statement essentially includes a KL type into the current one.
- Aside from types *require* can also be used to load KL extensions.
- The KL tool can be launched with the *loadexts* flag to load all of the standard extensions.

PEX – Parallel Execution

File reference: 01_kl/12_pex.kl

- KL can deploy multithreading in a variety of ways.
- PEX provides an explicit threading mechanism.
- PEX uses the *<<<* and *>>>* notation, similar to CUDA.
- PEX only works with *operators*, not with *functions*.

MR – Map Reduce

File reference: 01_kl/13_mapreduce.kl

- Can be used to perform recursive–parallel operations
- Compute on very large data sets
- Values inside of MR are created using Producers
- MR can return a single result only, for example.

FABRIC ENGINE