The LéPiX Manual

Fatima Koly (fak2116)
Manager
`fak2116@barnard.edu`

Gabrielle Taylor (gat2118)
Language Guru
`gat2118@columbia.edu`

Jackie Lin (jl4162)
Tester
`jl4162@columbia.edu`

Akshaan Kakar (ak3808)
Codegen
`ak3808@columbia.edu`

ThePhD (jm3689)
System Architect
`jm3689@columbia.edu`

`https://github.com/ThePhD/lepix`

October 8, 2016

Contents

1 General **2**
1.1 Introduction 2

2 Lexical Conventions **3**
2.1 Character Sets 3

3 Expressions **4**
3.1 Primary Expressions 4
3.1.1 Literal Sequences 4
3.2 Postfix Expressions 4
3.2.1 Subscription 4
3.2.2 Function Call 5
3.2.3 Access 5
3.3 Postfix Expressions 5
3.3.1 Subscription 5
3.3.2 Function Call 5
3.3.3 Access 5

Referenced Work **6**

Chapter 1

General

1.1 Introduction

This is the manual for LéPiX, which describes an abstract virtual machine for conducting computation. This manual also specifies the requirements for an implementation of LéPiX, and so also defines the LéPiX language.

Chapter 2

Lexical Conventions

2.1 Character Sets

1. The basic source character set consists of the ASCII letters and symbols.
2. The basic source character set will be used throughout this document to define the language and is the normative set of characters and control characters for this language.
3. The universal character name scheme provides a way to specify names outside of the basic source character set.
4. An implementation may provide support for a superset of the basic source character set or a compatible source character set, providing the following: every character in the basic source character set has a distinct and unambiguous mapping to a character in the implementation-defined source character set.

Chapter 3

Expressions

An expression can be one of several different groupings of lexical tokens that can be sequenced with other expressions.

3.1 Primary Expressions

A primary expression is one that does not need to be preceded by an existing expression to work. In general, this means identifiers (of functions or variables), anonymous closures (lambdas) and literal expressions. An expression takes the following form:

$\langle \textit{expr} \rangle$ ::= $\langle \textit{literal} \rangle$ | $\langle \textit{name-id} \rangle$ | $\langle \textit{lambda-expr} \rangle$

3.1.1 Literal Sequences

A literal is one of a numeric literal, which can be an integer literal or a float literal, a string literal, or an array literal. These are the same literals defined in the

$\langle \textit{literal} \rangle$ ::= $\langle \textit{string-literal} \rangle$ | $\langle \textit{integer-literal} \rangle$ | $\langle \textit{float-literal} \rangle$

3.2 Postfix Expressions

3.2.1 Subscription

1. Array Access is done with square brackets and refers to the expression that immediately precedes it.

3.2.2 Function Call

1. Function calls are done with the parentheses and is applied to the expression that immediately precedes it.

3.2.3 Access

1. A period can be applied to the preceding expression to access into the result of that expression.

3.3 Postfix Expressions

3.3.1 Subscription

1. Array Access is done with square brackets and refers to the expression that immediately precedes it.

3.3.2 Function Call

1. Function calls are done with the parentheses and is applied to the expression that immediately precedes it.

3.3.3 Access

1. A period can be applied to the preceding expression to access into the result of that expression.

Bibliography

[1] Khronos Group, *Khronos Registry: SPIR-V 1.1*, April 18, 2016. `https://www.khronos.org/registry/spir-v/`

[2] Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Pat Hanrahan, Mike Houston, Kayvon Fatahalian, *Brook for GPUs*, Stanford University, 2016. `https://graphics.stanford.edu/projects/brookgpu/arch.html`

[3] Khronos Group, *OpenCL 2.2*, April 12, 2016. `https://www.khronos.org/opencl/`

[4] Chuck Walbourn, Microsoft, *Compute Shaders*, July 14, 2010. `https://blogs.msdn.microsoft.com/chuckw/2010/07/14/directcompute/`

[5] Dillion Sharlet, Aaron Kunze, Stephen Junkins, Deepti Joshi, *Shevlin Park: Implementing C++ AMP with Clang/LLVM and OpenCL*, November 2012. `http://llvm.org/devmtg/2012-11/Sharlet-ShevlinPark.pdf`