LEARN PARROT

parrot perl virtual machine

tutorialspoint

SIMPLYEASYLEARNING

About the Tutorial

Parrot is a virtual machine designed to efficiently compile and execute bytecode for interpreted languages.

Parrot is going to change the way you see PERL!

Audience

This tutorial has been designed for users who are willing to learn Microsoft PowerPoint in simple steps and they do not have much knowledge about computer usage and Microsoft applications. This tutorial will give you enough understanding on MS PowerPoint from where you can take yourself at higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of computer peripherals like mouse, keyboard, monitor, screen etc. and their basic operations.

Copyright & Disclaimer

Table of Contents

1. Parrot – Overview

When we feed our program into conventional Perl, it is first compiled into an internal representation, or bytecode; this bytecode is then fed into almost separate subsystem inside Perl to be interpreted. So, there are two distinct phases of Perl's operation:

- Compilation to bytecode and
- Interpretation of bytecode.

This is not unique to Perl. Other languages following this design include - Python, Ruby, Tcl and even Java.

We also know that there is a Java Virtual Machine (JVM) which is a platform independent execution environment that converts Java bytecode into machine language and executes it. If you understand this concept then you will understand Parrot.

Parrot is a virtual machine designed to efficiently compile and execute bytecode for interpreted languages. Parrot is the target for the final Perl 6 compiler, and is used as a backend for Pugs, as well as variety of other languages like Tcl, Ruby, Python etc.

Parrot has been written using most popular language "C".

2. Parrot – Installation

Before we start, let's download one latest copy of Parrot and install it on our machine.

Parrot download link is available in Parrot CVS Snapshot. Download the latest version of Parrot and to install it follow the following steps:

- Unzip and untar the downloaded file.
- Make sure you already have Perl 5 installed on your machine.
- Now do the following:

```
% cd parrot

% perl Configure.pl

Parrot Configure

Copyright (C) 2001 Yet Another Society

Since you're running this script, you obviously have

Perl 5 -- I'll be pulling some defaults from its configuration.

...
```

- You'll then be asked a series of questions about your local configuration; you can almost always hit return/enter for each one.
- Finally, you'll be told to type - make *test_prog,* and Parrot will successfully build the test interpreter.
- Now you should run some tests; so type 'make test' and you should see a readout like the following:

```
perl t/harness

t/op/basic.....ok,1/2 skipped:label constants unimplemented in

assembler

t/op/string....ok, 1/4 skipped:  I'm unable to write it!

All tests successful, 2 subtests skipped.

Files=2, Tests=6,......
```

By the time you read this, there could be more tests, and some of those which skipped might not skip, but make sure that none of them should fail!

Once you have a parrot executable installed, you can check out the various types of examples given in Parrot 'Examples' section. Also you can check out the examples directory in the parrot repository.

3. Parrot – Instructions Format

Parrot can currently accept instructions to execute in four forms. PIR (Parrot Intermediate Representation) is designed to be written by people and generated by compilers. It hides away some low-level details, such as the way parameters are passed to functions.

PASM (Parrot Assembly) is a level below PIR - it is still human readable/writable and can be generated by a compiler, but the author has to take care of details such as calling conventions and register allocation. PAST (Parrot Abstract Syntax Tree) enables Parrot to accept an abstract syntax tree style input - useful for those, writing compilers.

All of the above forms of input are automatically converted inside Parrot to PBC (Parrot Bytecode). This is much like machine code, but understood by the Parrot interpreter.

It is not intended to be human-readable or human-writable, but unlike the other forms execution can start immediately without the need for an assembly phase. Parrot bytecode is platform independent.

The Instruction Set

The Parrot instruction set includes arithmetic and logical operators, compare and branch/jump (for implementing loops, if...then constructs, etc.), finding and storing global and lexical variables, working with classes and objects, calling subroutines and methods along with their parameters, I/O, threads and more.

4. Parrot – Garbage Collection

Like Java Virtual Machine, Parrot also keep you free from worrying about memory de-allocation.

- Parrot provides garbage collection.
- Parrot programs do not need to free memory explicitly.
- Allocated memory will be freed when it is no longer in use i.e. no longer referenced.
- Parrot Garbage Collector runs periodically to take care of unwanted memory.