UNIVERSIDADE ESTADUAL PAULISTA
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO E ESTATÍSTICAS

LORRAINE MARIA PEPE
GUSTAVO FERNANDO BALBINO

ESTUDO SOBRE A LINGUAGEM PERL E SUAS CARACTERÍSTICAS

SÃO JOSÉ DO RIO PRETO – SP
JULHO/2017

RESUMO

Neste trabalho é apresentado um estudo sobre a linguagem Perl, com foco voltado para suas características, além de apresentar uma análise das principais instruções e estruturas que servem de base para o início de seu aprendizado, e também uma visão geral de suas aplicações e como vem se dando seu crescimento até os dias de hoje.

Palavras-chaves: Perl, Larry Wall, Unix, linguagem de *scripting*.

1 INTRODUÇÃO

No meio de diversas linguagens de programação existentes, com suas diversas especialidades e finalidades, a Practical Extraction and Report Language*, (do Inglês, significa extração prática e linguagem de relatório), se destaca principalmente devido a sua versatilidade para a manipulação de cadeia de caracteres, além de sua facilidade na criação de formulários www e realização de operações administrativas no sistema UNIX.

Outra vantagem é o fato de ser open source, isto é, código aberto, e passar por mutações até hoje, sendo portada em mais de 100 diferentes plataformas, expandindo assim o significado de seu slogan "There's more than one way to do it" (Existe mais de uma maneira de fazer isso). A seguir, uma análise mais detalhada dessa linguagem.

* Devido ao extenso nome da linguagem, é amplamente difundido o acrônimo Perl para representá-la, mesmo que este não seja oficial. De agora em diante, utilizaremos apenas o acrônimo.

2 HISTÓRIA

Criada em 1987, Perl foi assim nomeada de acordo com sua utilidade inicial. Foi desenvolvida por Larry Wall quando esse trabalhava no Laboratório de Jatos Propulsores da NASA, e percebeu que havia necessidade de agrupar as diversas ferramentas então usadas para processamento de texto em UNIX em apenas uma, facilitando suas tarefas.

Sendo assim, ele optou por tentar unir as melhores características de todos os recursos que usava: as expressões regulares Stream EDitor (sed), a profundidade e partes da sintaxe de C, o reconhecimento de padrões de AWK, mesclando também com a sintaxe de Shell Script.

Seu lançamento se deu 4 anos antes do sistema operacional LINUX e popularizou-se rapidamente, atingindo sua quarta versão já em 1992 e configurando-se como linguagem padrão do UNIX.

Nem tudo ocorreu como esperado, pois os programadores da época perceberam que não era possível resolver problemas maiores utilizando-a. Então decidiram criar mais uma versão, a quinta, que elevou Perl para o patamar de linguagem completa, sendo finalista, em 1998, do Prêmio de Excelência Técnica no quesito Ferramentas de Desenvolvimento. E graças à essas expansões que hoje em dia, Perl é utilizada inclusive na solução de problemas críticos do setor público e privado.

Dentre outras curiosidades, as versões da linguagem foram disponibilizadas gratuitamente, sem a cobrança de licenças, já que é open source, estando registrada sob a licença artística GNU General Public License (GLP).

Até o momento, já foram lançadas 6 versões de Perl, essa última, em dezembro de 2015.

3 CARACTERÍSTICAS

3.1 Interpretação

Perl é considerada por muitos uma linguagem de *scripting*, ou seja, linguagens usadas para programar tarefas a fim de automatizar processos em tempo de execução, auxiliando programas maiores. Geralmente, essas linguagens são interpretadas, seja parcial ou completamente, ao invés de compiladas, tornando-se mais lenta que muitas linguagens.

Mas, para reverter esse quadro, ela foi espelhada na compilação das linguagens imperativas, compilando seus programas ao menos para uma linguagem intermediária antes da execução de seus códigos. À essa característica dá-se o nome de hibridismo.

Em Perl usa-se como intermediária a linguagem C além de que muitas das estruturas e comandos utilizados na linguagem de Larry se assemelham a ela.

Outrossim, um programa depois de ser interpretado consegue ser facilmente executado em outras máquinas desde que estas forneçam um interpretador à linguagem e sem que haja uma nova interpretação. Atualmente, as máquinas mais utilizadas no mercado já possuem um interpretador de Perl embutido em seu sistema.

3.2 Simplicidade

Em se tratando de estruturas de controle, repetições e algumas palavras reservadas, Perl pode ser considerada bem simples, assemelhando-se a C. Porém, por possuir muitos subtipos de dados tem-se um empecilho. Como por exemplo em **“Ola, mundo”** e **‘Ola, mundo’** que são duas formas de *Strings* idênticas, mas com efeitos diferentes na impressão da mensagem, afetando diretamente na legibilidade e entendimento do programa.

Outro fato é que, apesar de seu lema de haver mais de uma maneira para se realizar uma tarefa, no quesito simplicidade isso pode atrapalhar, levando programadores a cometerem erros por utilizarem recursos de forma inadequada ou até mesmo escrevendo de maneira que outro profissional possa não entender, sendo pontos negativos também para a legibilidade, redigibilidade e capacidade de modificação do código.

Mas, do ponto de vista de aprendizagem, pode ser sim considerada simples. Devido a sua característica de multiplataforma, basta que o programador conheça alguma outra linguagem e então encontrará nela algumas semelhanças, facilitando seu estudo e ganho de experiência.

3.3 Portabilidade

Com a característica de código aberto foi possível diversos ajustes desde a primeira versão, pois diversas pessoas conseguiram fazer suas contribuições para o enriquecimento da linguagem. Isso, além de contribuir para a fama de Perl, trouxe à ela os aspectos de ser altamente portável, reutilizável e modulável, o que significa dizer que Perl é portátil aos mais variados tipos de hardwares e pode ser expandido para uma necessidade específica a partir de seus módulos e da reutilização de recursos.

Essa última intensificou-se ainda mais com o suporte ao paradigma orientado a objetos, acrescentado na quinta versão. Já sobre a modularização, existem mais de 20 mil módulos para serem incrementados à linguagem disponíveis no CPAN *(Comprehensive Perl Archive Network*). E quanto a portabilidade, se deve também à característica da linguagem de aceitar diversas plataformas, sendo versátil. Por exemplo podemos mencionar as seguintes: Unix, Windows, MVS, VMS,

Macintosh, OS/2 e Amiga. Mas a verdade é que já foram listadas mais de 100 dessas ao qual Perl se faz presente.

3.4 Ortogonalidade, expressividade e suporte à abstração

Como a maioria das linguagens hoje em dia, possui suporte para abstração, ou seja, permite realizar construções complicadas e depois utilizá-las sem a necessidade de se compreender bem seu funcionamento interno. Isso se dá por exemplo construindo de sub-rotinas. Mas nem sempre foi assim, como já mencionado em sua história.

Quanto à expressividade, que diz respeito à capacidade de transferir uma ideia para a forma de algoritmos, Perl é considerada bastante expressiva, o que por muitos é considerado vantajoso e para outros não, ainda mais se tratando de programas de grande porte.

Algo que interfere diretamente na expressividade é a ortogonalidade, visto que possui um pequeno conjunto de construções primitivas que combinadas entre si podem formar novas estruturas. Como Perl é altamente ortogonal, seguindo seu lema, diferentes combinações podem originar estruturas com mesmas funcionalidades afetando também a simplicidade, como já discutido.

3.5 Confiabilidade e segurança

Uma grande vantagem encontrada é a verificação de tipos (por exemplo, **int**, **float** e **double**) que ocorre no processo de pré-compilação. Esse processo analisa se os tipos utilizados em dada expressão estão corretos e eram o esperado naquele momento.

Outra, é a possibilidade da manipulação de exceções. Em Perl é feita por meio do eval BLOCK, que é uma análise feita em tempo de compilação, permitindo que erros sejam contornados na execução.

Em contrapartida, é possível o fenômeno chamado *aliasing,* que é a capacidade para se utilizar nomes distintos para uma mesma região ou célula da memória de um sistema.

Somando como desvantagem temos também a permissão da instrução **goto**, muito mal vista pela comunidade de programadores pois deixa o código confuso e propenso a erros.

Ademais, há em Perl uma grande quantidade de palavras e caracteres reservadas, passando de trezentas, mas que decaiu significativamente com o decorrer das versões. A lista completa pode ser conferida em https://learn.perl.org/docs/keywords.html (Acessado em: julho de 2017).

Além disso, como em muitos pontos a simplicidade fica afetada, então a confiabilidade também, já que dá abertura à más interpretações do código, facilitando que modificações errôneas aconteçam.

Sabe-se que um programa seguro, seja para a internet, para sistemas administrativos ou qualquer outra utilidade, necessita ser confiável. E embora a confiabilidade tenha muitos pontos negativos, Perl oferece suporte para a criação de scripts através de um módulo de da versão cinco, chamado de *pgperl* que permite ao seu servidor usar técnicas públicas de codificação para salvaguardar dados.

3.6 Flexibilidade

A Perl não foi desenvolvida em um plano abstrato. Ela foi criada para resolver um problema particular e evoluiu de modo a servir para um conjunto ainda mais amplo de problemas do dia-a-dia.

Claro que poderia ter sido expandida para lidar com essas tarefas adicionando-se mais e mais palavras-chave e operadores, o que tornaria a linguagem bem maior. Em vez disso, o núcleo da linguagem iniciou pequeno e ficou mais refinado à medida que o tempo passou. Em alguns aspectos, ela na verdade diminuiu já que o número de palavras reservadas da Perl 5 é realmente menos da metade do número de palavras reservadas da Perl 4.

Isso reflete uma preocupação de que Perl resida na sua combinação única de eficiência e flexibilidade. Percebe-se que a linguagem tem crescido lenta e poderosamente, em geral de forma a permitir o acréscimo de melhorias e extensões em vez de receber amarrações.

3.7 Ubiquidade

Essa linguagem já está bastante difundida no mundo, seu uso tem crescido frequentemente. Com o lançamento da sexta versão, Perl fez um grande salto entre o ranking das linguagens mais utilizadas. Atualmente já ultrapassou iOS, PHP e tirou Ruby da lista, segundo o site E-cogni (Disponível em: https://www.e-cogni.com.br/programacao/as-9-linguagens-de-programacao-com-maior-demanda-para-2017/. Último acesso: julho de 2017). Não é à toa que desde suas primeiras versões é conhecida como “a fita adesiva da internet”.

Muitos acham que esse crescimento se deve ao aumento de DevOps, que é a união dos desenvolvedores com a equipe de operações visando a produtividade, visto que Perl se adequa bem ao auxiliar outras linguagens.

A cada dia inúmeras empresas e organizações utilizam programas em Perl em pontos críticos de seus negócios. Bancos como Citigroup, Bank of America, e Deutsche Bank, Hospitais a exemplo o de Lausanne, na Suíça e multinacionais como Amazon, VeriSign e Tickermaster depositam sua confiança neles.

3.8 Projeto de sintaxe

Como já foi dito, Perl possui muitas palavras e caracteres reservados. Essas fazem menção a seus identificadores, que são nomes de instruções, funções ou variáveis. Tais, embora não possam ser usadas pelos programadores, geralmente são intuitivas. E caso se precise de algo com o mesmo significado, embora não aconselhado, pode-se usar sentenças similares, mas diferidas em maiúsculas e minúsculas, visto que é uma linguagem *case sensitive*, ou seja, reconhece-as como palavras diferentes.

4 CUSTOS

Por ser software livre, fica isento de custos com licenças, que no caso de outras linguagens podem ser caríssimas. Além disso, devido a sua portabilidade, não necessita uma nova implementação para mudanças de plataforma. Outra vantagem, é a existência de uma comunidade ativa, fato que mantém a linguagem em uso, documentações em dia, além de recorrentes melhorias e manutenções.

Já quanto ao custo de treinamento, existem diversos cursos online, tutoriais, e grupos criados pela própria comunidade para debates e esclarecimento de dúvidas. Outro fato que corrobora com este é a facilidade do uso e aprendizado da linguagem, já mencionado anteriormente.

Por fim, em se tratando de custo de manutenção, devido ao seu escopo léxico e presença de caracteres que marcam a qual tipo um dado pertence (@, #, e $), fica fácil visualizar e entender o que foi feito, agilizando esse processo.

5 DOCUMENTAÇÃO

Ao contrário de muitas linguagens, a documentação de Perl ainda não foi traduzida completamente para o inglês, mas essa tarefa está sendo cumprida aos poucos por sua comunidade a fim de que mais pessoas possam usá-la aqui no Brasil e em outros países de mesma língua.

6 VISÃO GERAL DA SINTAXE

Por convenção, para melhor visualização e organização, colocaremos as instruções todas em negrito.

6.1 Início e observações

Todo programa inicia-se com a seguinte instrução: **#!/usr/local/bin/perl**, ela indica ao sistema o local em que se encontra seu compilador, podendo variar conforme a máquina utilizada.

Já para a inserção de comentários havia apenas uma forma, pelo caractere #, onde tudo o que vier após ele em uma linha é considerado comentário, com exceção da primeira linha. Atualmente pode-se adicionar comentários entre as instruções **=pod** e **=cut** para essa finalidade também, não necessitando o #.

Sobre parênteses, só são obrigatórios para esclarecimento de precedências e em na nomenclatura de funções contendo os parâmetros ou argumentos dessas. É importante ressaltar também que cada linha de instrução em Perl com exceção de comentários e início e fim de blocos de comandos são sucedidas por **;** .

Para poder se executar um programa em Perl, deve-se ter o interpretador e um bloco de notas qualquer para a escrita do código. Porém pode-se baixar kits de desenvolvimento fornecidos gratuitamente como o Strawberry Perl ou o ActiveState Perl. Já de IDE, ambiente de desenvolvimento integrado, pode-se usar o Padre, Komodo ou Open Perl.

6.2 Tipo de dados

Em Perl há basicamente três tipos gerais de dados, cada qual com seus subtipos. São eles: *scalar*(**$**), *arrays*(@) e *hashes*(**%**). Quando são usados, não há a necessidade de sua declaração, mas como boa prática e para evitar erros com variáveis globais pode-se usar o comando **my** VARIAVEL**;** em que VARIAVEL é um tipo.

Escalares (Scalar) são variáveis simples, podendo ser um número, uma *string*, ou uma referência, que é o endereço para uma outra variável. Já os vetores (arrays) são listas de escalares, com nome precedido pelo sinal @ e cada elemento contendo um índice indicando sua posição no vetor, com contagem iniciando em 0, além de ser todos de um único tipo (inteiro, float, *string,* entre outros). E por último temos os hashes, chamados também de vetores associativos, que funcionam como listas de valores aleatórios, podendo receber mais de um tipo de dados, cada qual sendo formado por um par de chave mais valor de associação, esses são precedidos por um **%.**

Como **$**, @ e **%** são alguns dos caracteres reservados, caso haja necessidade de imprimi-los ou usa-los para compor frases, deve-se precede-los por uma ****, bem como na escrita da barra, em que se usa **** para exibi-las. Exemplo: **print " \@ é para vetores \\ $ para escalares";**. A saída será: @ é para vetores \ $ para escalares.

```
#Tipos de dados
my $var = "Ola";
my $var2;
my $preco = 3.99;
my @array = ( 5, 4, 3);
my %idade = ('Jose', 45, 'Maria', 30);
```

```
#Jose é a chave e seu valor é 45
```

6.3 Operadores

Há muitos tipos de operadores em Perl. Adiante estão os mais comuns, embora existam muitos outros:

**Aritméticos: + , - , * , / , % , ** (adição, subtração, multiplicação, divisão, resto da divisão, exponenciação, respectivamente.

Lógicos: && , || , ! (e, ou, não) **, & , | , ~ , ^** (e bit-a-bit, ou bit-a-bit, não bit-a-bit, xor bit-a-bit), **and , or , not** (tem menor precedência que os 3 primeiros) **, xor** (ou exclusivo).

String: . (concatenação).

Relacionais: ==, > , < , >= , <= , != **(contexto numérico) **, eq , gt , lt , ge , le , ne (contexto de string).

De teste (para expressões regulares): =~ (para tentar "casar" informações) e **!=** (para negar estas).

6.4 Sub-rotinas

Uma sub-rotina ou função é um agrupamento de instruções bem definidas com a utilidade de organizar o código, além de evitar repetições de instruções nesses. Uma sub-rotina é definida da seguinte forma:

```
sub nome_da_sub{
      instruçoes
}
```

E chamada para uso assim:

```
nome_da_sub(argumentos);  #nas versões 5 e 6
ou
&subroutine_name( list of arguments ); #nas versões anteriores
```

Elas podem retornar valores de seu corpo para o lugar onde foi chamada através da instrução **return**. Por exemplo **return 0;** ou ainda **return $var**. Podendo retornar qualquer tipo de dado.

6.5 Estruturas de controle

Há dois tipos de desvios em Perl, os condicionais e os incondicionais. O primeiro depende de um fato para ocorre, já o segundo não. São eles: (Utilizamos aqui colchetes para indicar opcionalidade).

Condicionais:

if (EXPRESSÃO) { BLOCO } [[**elseif** (EXPRESSÃO) { BLOCO }]
else { BLOCO }]
unless (EXPRESSÃO) { BLOCO } [**else** { BLOCO }]
do { BLOCO } **unless** (EXPRESSÃO);

switch{} – incorporado ás últimas versões, funciona como menu de condições.

(EXPRESSÃO)**?** INSTRUÇÃO1 **:** INSTRUÇÃO2 – **?:** é o operador ternário Se a expressão for verdadeira realiza a instrução1, senão a 2.

if e unless – se a expressão acontecer, realizará o procedimento

do{}unless() – faça o bloco entre chaves se unless acontecer.

elsif – senão se acontecer essa outra.

else – senão, faça isso.

Incondicionais:

return EXPRESSÃO - sai da sub-rotina retornando o valor da EXPRESSÃO.
goto RÓTULO - desvia para a sentença rotulada com RÓTULO.
last [RÓTULO] - desvia para a primeira sentença após o laço rotulado com RÓTULO ou para a primeira sentença após o laço corrente, se RÓTULO for omitido.
next [RÓTULO] - inicia uma nova iteração do laço rotulado com RÓTULO ou do laço corrente, se RÓTULO for omitido.
redo [RÓTULO] - reinicia a iteração corrente do laço rotulado com RÓTULO ou do laço corrente, se RÓTULO for omitido.

Além do mais, há laços condicionais e laços contados, que servem para repetir instruções ou caso a expressão seja satisfeita, no caso dos condicionais, ou pela quantia de vezes determinada. Observe:

Laços condicionais:

while (EXPRESSÃO) { BLOCO } – enquanto tal expressão ocorrer, execute o bloco. Necessita uma condição de parada no bloco.

do { BLOCO } **while** (EXPRESSÃO); - Faça o bloco, até que a expressão seja falsa. Semelhante ao anterior, porém é executado uma vez antes de analisar a expressão.
until (EXPRESSÃO) { BLOCO } – o oposto do **while**(){}, executa o bloco até que a condição seja verdade. Analisa a expressão primeiro.
do { BLOCO } **until** (EXPRESSÃO); - o oposto do **do{}while()**, até que a expressão seja verdadeira.

Laços contados:

for (INICIALIZAÇÃO; CONDICÕES-LIMITE; INCREMENTO) { BLOCO }- inicializa um contador, verifica a condição sobre o contador, se for verdadeira, realiza o bloco, incrementando ou decrementando o contador e novamente verificando. A inicialização ocorre somente uma vez.
foreach VARIÁVEL (LISTA) { BLOCO } – realiza o bloco para cada elemento corrente da lista.

6.6 Expressões regulares

Uma das ferramentas mais poderosas da linguagem é o uso de expressões regulares. Tais expressões possuem a funcionalidade de definir um padrão de formação ou formatação de alguma informação e, de posse desse padrão, tentar assimilar informações com ela. Caso o padrão seja encontrado dentro desse conteúdo, alguma ação é tomada e o processamento segue adiante.

Essas expressões facilitam na busca e reconhecimento sobretudo de *strings*, sendo grande aliada na programação de editores de textos, na criação de relatórios e análises de bioinformática.

Não aprofundaremos nesse assunto devido a sua extensão e por representar um conteúdo para níveis mais avançados, o que não é alvo dessa análise.

7 APLICAÇÕES

Além de todas as sugestões e ideias de onde Perl pode ser aplicada, pode-se analisar outras aplicações para transparecer o sucesso que ela vem obtendo até hoje. Sendo assim, tem-se por exemplo seu uso em lugares como: Human Genome Project auxiliando no mapeamento dos genes humanos, Disney, DreamWorks, Nintendo, Oracle, Embratel e Claro.

8 CONCLUSÕES FINAIS

Com toda essa análise, pode-se concluir que, de fato, Perl é uma “fita adesiva” que vem mostrando cada dia mais possibilidades de aplicações e ganho de espaço no mercado. Embora aqui no Brasil ainda não seja muito mencionada, pode-se afirmar que se suas evoluções continuarem conforme se deram até agora, isto é, de maneira pensada e bem estruturada, em breve ainda se ouvirá muito sobre ela.

Através desse estudo também, foi possível perceber como a falta de uma documentação bem esclarecida de cada versão, o que ocorre com as demais linguagens, faz falta, já que é necessário buscar informações na comunidade que muitas vezes não especificam à qual versão pertencem.

Concluindo, Perl é exatamente como descreveu Larry Wall, seu criador: “Perl foi projetado para nunca ser perfeito. Foi projetado para evoluir, para tornar-se mais adaptável”, mantendo assim seu ciclo de vida e dando cada vez mais suporte a diversas necessidades humanas, bem como melhorar ferramentas e recursos já existentes, de forma exclusiva e que só Perl tem a oferecer.

9 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sebesta, R.W; **Conceitos de linguagens de programação**. 9º edição. Bookman, 2010.

[2] Deitel, H.M.; Deitel, P.J.; **Perl, como programar**. Bookman, 2002.

[3] **Instituto de Matemática e Estatística da Universidade de São Paulo.** Disponível em: https://www.ime.usp.br/~glauber/perl/perl.htm. Último acesso: 03 de julho de 2017.

[4] **W3ii.com**. Disponível em: http://www.w3ii.com/pt/perl/perl_data_types.html. Último acesso: 05 de julho de 2017.

[5] **São Paulo Perl Mongers.** Disponível em: http://sao-paulo.pm.org/pub/por-que-perl. Último acesso: 06 de julho de 2017.

[6] **Perl Maven.** Disponível em: https://br.perlmaven.com/. Último acesso: 06 de julho de 2017.

[7] **Verdade @bsoluta**. Disponível em: http://www.absoluta.org/cgi/cgi_perl.htm. Último acesso: 06 de julho de 2017.

[8] **Utilização à lógica de programação utilizando Perl**. Curso de Verão de Bioinformática, 2008; Laboratório de Genética Molecular e Bioinformática; Departamente de Genética – FMRP/USP. Disponível em: http://lgmb.fmrp.usp.br/cvbioinfo2008/extras/apresentacoes/thiago_logica.pdf. Último acesso: 06 de julho de 2017.