



www.perl.org

Learning Perl Through Examples Part I

L1110@BUMC

9/21/2017

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Tutorial Resource

Before we start, please take a note - all the code scripts and supporting documents are accessible through:

- <http://rcs.bu.edu/examples/perl/tutorials/>



www.perl.org

Sign In Sheet

We prepared sign-in sheet for each one to sign
We do this for internal management and quality control
So please SIGN IN if you haven't done so



www.perl.org

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Research Computing Services (RCS)



www.perl.org

- RCS is a group within Information Services & Technology (IS&T) at Boston University provides computing, storage, and visualization resources and services to support research that has specialized or highly intensive computation, storage, bandwidth, or graphics requirements.
- **Three Primary Services:**
 1. Research Computation
 2. Research Visualization
 3. Research Consulting and Training
- More Info: <http://www.bu.edu/tech/about/research/>

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Research Computing Services (RCS) Tutorials



www.perl.org

RCS offers three times a year tutorials

- Spring – in January/February
- Summer – in May/June
- Fall – in September/October

This Perl tutorial is part I of a set (Part II come tomorrow)

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

About Me

- Join RCS March 2016
- long time programmer, dated back in 1987
- Proficient in C/C++/Perl
- Domain knowledge: Network/Communication, Databases, Bioinformatics, System Integration.
- Contact: yshen16@bu.edu, 617-638-5851
- Main Office: 801 Mass Ave. 4th Floor (Crosstown Building)

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Tell Me A bit about You

- Name
- Experience in programming? If so, which specific language?
Self rating?
- Experience in Perl?
- Account on SCC?
- Motivation (Expectation) to attend this tutorial
- Any other questions/fun facts you would like the class to know?



www.perl.org

Evaluation

One last piece of information before we start:

- DON'T FORGET TO GO TO:
 - http://rcs.bu.edu/survey/tutorial_evaluation.html

Leave your feedback for this tutorial (both good and bad as long as it is honest are welcome. Thank you)

Topics for today

Background

Get to know Perl Environment

Using Perl

Code Examples

Packages and Modules

Perl help system

Perl Debugger

Q & A



www.perl.org

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Background

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

What Is Perl

Perl - the most famous backronym rather than an acronym

"Practical Extraction and Reporting Language".

- Developed by Larry Wall in 1987 at System Development Corporation (part of UniSys later on)
- originally as a Unix Scripting Language
- Grown to be a full blown programming language, with many features borrowed from other languages, such as C/sh/Lisp/AWK/sed/CGI
- Perl5 and Perl6 are mostly used now; this tutorial will focus on Perl5
- See official definition on <http://www.perl.org>



www.perl.org

Language Design Philosophy



www.perl.org

- “There's more than one way to do it” design philosophy and **multi-paradigm, dynamically typed** language features leads to great degree of flexibility in program design.
- CPAN and Perl Module (191,032 available modules in CPAN in 35,637 distributions, written by 13,218 authors, mirrored on 250 servers over 60 countries)
- CPAN is honored to be called Perl’s ‘killer app’ (see <https://en.wikipedia.org/wiki/CPAN> for more)



www.perl.org

Perl Classification

Perl 5 and 6 are considered a family of **high-level, general-purpose, interpreted, dynamic** programming languages.

- High-level – syntax/semantics close to natural language
- General purpose – not limited to specific tasks in a particular application domain
- Interpreted – relative to compiled language (prepared/checked vs real-time/interactive)
- Dynamic – not strict in predefined data type constraints, etc.

Borrowed Features



www.perl.org

Perl Borrows many features from other programming languages

- From C: procedural, variables, expression, assignment (=), brace-delimited blocks ({}, ;), control flow (if, while, for, do, etc), subroutine
- From shell: '\$' sign, system command
- From Lisp: lists data structure; implicit return value
- From AWK: hash
- From sed: regular expression

Authentic Features



www.perl.org

Perl's most authentic features of its own:

- auto data-typing
- auto memory management
- It's all handled by Perl interpreter

These are very powerful features and contribute a lot to the wide adoption of Perl language

more details on Perl5 feature summary: <https://www.perl.org/about.html>

Where Perl is used

- System administration
- Configuration management
- Web sites/web application
- Small scripts
- Bioinformatics
- Scientific calculations
- Test automation
- ... (the riches lie in CPAN)



www.perl.org

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Swiss Army Chainsaw or Duct Tape of Internet?



www.perl.org

Perl gained its nickname of ‘Swiss army chainsaw’ for its flexibility and power; its ‘Duct Tape of Internet’ for its ability and often ‘ugly’, quick, easy fixes for solutions to various problems. Commonly referred applications:

- Powerful text processing without data length limitation
- Regular expression and string parsing capability
- CGI (duct tape, glue language for Internet)
- DBI
- BioPerl



www.perl.org

Major versions

- Perl 5 – almost rewrite of Perl interpreter, adding object-oriented (OO) feature, complex data structure, module and CGI support. Among them, module support plays critical role to CPAN's establishment, and nowadays a great resource and strength for Perl community
- Perl 6 – fundamentally different from Perl 5, dedicated to Larry's birthday, goal is to fix all the warts in Perl 5; it's said to be good at **all that Perl 5 is good at, and a lot more.**

Language Scope



www.perl.org

- Perl is highly extensive language
- Open source framework – CPAN model
- CPAN and Perl Module
 - 191,032 available modules
 - 35, 637 distributions
 - written by 13,218 authors
 - mirrored on 250 servers

Language Elements

- Data Types
 - scalar, array, hash, reference
- Control Structures
 - for, while, if, goto (yes, there is a Goto)
- Regular Expressions
- User Defined Extensions (Subroutines and functions)
- Objects/modules/packages



www.perl.org



www.perl.org

Advantage Over C

- Perl runs on all platforms and is far more portable than C.
- Perl and a huge collection of Perl Modules are free software (either GNU General Public License or Artistic License).
- Perl is very efficient in TEXT and STRING manipulation i.e. REGEXP.
- It is a language that combines the best features from many other languages and is very easy to learn.
- Dynamic memory allocation is very easy in PERL, at any point of time we can increase or decrease the size of the array (i.e. splice(), push())



www.perl.org

Disadvantage Over C

- You cannot easily create a binary image ("exe") from a Perl file. It's not a serious problem on Unix, but it might be a problem on Windows.
- Moreover, if you write a script which uses modules from CPAN, and want to run it on another computer, you need to install all the modules on that other computer, which can be a drag.
- Perl is an interpretative language, so its comparatively slower to other compiling language like C. So, it's not feasible to use in Real time environment like in flight simulation system.

Some famous applications

- Web CGI (EBay, Craigslist, BBC, Amazon, ...)
- 1000 Genome Project
- Financial analysis (ease of use, speed for integration, rapid prototyping) - BarclaysCapital
- Summarizing system logs/deal with Windows registry or Unix Passwd or groups file



www.perl.org



www.perl.org

Get To Know Environment

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Connecting to SCC

- Option 1: You are able to keep everything you generate
Use your Shared Computing Cluster account if you have one.
- Option 2: all that you do in the tutorial may be wiped out after tutorial ends unless you move the contents to somewhere belong to you.
 - Tutorial accounts if you need one (will be offered in class).
 - Username: TBD
 - Password: TBD

Download source code



www.perl.org

Follow these steps to download the code:

```
ssh user@sccN.bu.edu ('user' is an account on SCC, 'N' can be 1-4)
```

```
mkdir perlThruEx
```

```
cd perlThruEx
```

```
wget http://scv.bu.edu/examples/perl/tutorials/src/perlThruExamples.zip
```



www.perl.org

Exercise 1 - Where is My Perl

Two commands to use:

‘which perl’

and

‘perl -v’

Do the experiment on next page to help understand the concept and discover more



www.perl.org

Exercise 1a - Where is My Perl

Type 'which perl' in terminal

```
[yshen16@scc4 beginner_perl]$ which perl  
/usr/local/bin/perl
```

Now type 'perl -v'

```
[yshen16@scc4 beginner_perl]$ perl -v  
This is perl, v5.10.1 (*) built for x86_64-linux-thread-multi  
Copyright 1987-2009, Larry Wall  
Perl may be copied only under the terms of either the Artistic License or the  
GNU General Public License, which may be found in the Perl 5 source kit.  
Complete documentation for Perl, including FAQ lists, should be found on  
this system using "man perl" or "perldoc perl". If you have access to the  
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```



www.perl.org

Exercise 1b - Where is My Perl

Type 'module load perl', then type 'which perl' in terminal

```
[yshen16@scc4 beginner_perl]$ module load perl  
[yshen16@scc4 beginner_perl]$ which perl  
/share/pkg/perl/5.24.0/install/bin/perl
```

Now type 'perl -v'

```
[yshen16@scc4 beginner_perl]$ perl -v  
  
This is perl 5, version 24, subversion 0 (v5.24.0) built for x86_64-linux  
Copyright 1987-2016, Larry Wall  
  
Perl may be copied only under the terms of either the Artistic License or the  
GNU General Public License, which may be found in the Perl 5 source kit.  
  
Complete documentation for Perl, including FAQ lists, should be found on  
this system using "man perl" or "perldoc perl". If you have access to the  
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```



www.perl.org

Exercise 1 - Observation

What's the difference between Exercise 1a and 1b?

BOSTON
UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

What do we learn from Exercise 1

- Perl is an environment – means it can be changed by pointing to different installations.



www.perl.org



www.perl.org

Exercise 2 – Perl Program Structure

Open code examples in gedit and browse the content:

`codeEx_simplest.pl` and `codeEx_simplest.pl.nofirst`

Try to run the following commands:

```
./codeEx_simplest.pl  
./codeEx_simplest.pl.nofirst
```

What happened?



www.perl.org

Exercise 2 – Perl Program Structure (2)

Here is what would be:

```
[yshen16@scc4 code]$ ./codeEx_simplest.pl
Hello World!
[yshen16@scc4 code]$ ./codeEx_simplest.pl.nofirst
./codeEx_simplest.pl.nofirst: line 3: print: command not found
[yshen16@scc4 code]$
```

Now try to run the following command:

```
perl ./codeEx_simplest.pl.nofirst
```

What happened?



www.perl.org

Exercise 2 – Perl Program Structure (3)

Here is what would be this time:

```
[yshen16@scc4 code]$ ./codeEx_simplest.pl.nofirst
./codeEx_simplest.pl.nofirst: line 3: print: command not found
[yshen16@scc4 code]$ perl ./codeEx_simplest.pl.nofirst
Hello World!
[yshen16@scc4 code]$
```

So why? Why is 'perl' in the command so critical to the 2nd code example?

Topic: Perl program and OS

Exercise 2 – Check Source Code



www.perl.org

```
emacs@scc4.bu.edu
File Edit Options Buffers Tools Help
#!/usr/local/bin/perl
# Hello World Program in Perl
print "Hello World!\n";
codeEx_simplest.pl All L1 (Per

emacs@scc4.bu.edu
File Edit Options Buffers Tools Help
# Hello World Program in Perl
print "Hello World!\n";
codeEx_simplest.pl.nofirst All
```



Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Comments on Exercise 2

Comment#1: file name doesn't matter (.pl is just a convention)

Comment#2: file permission doesn't matter (the file can be in plain readable text permission)

Reason: in the first command, `./codeEx_simplest.pl`, the file functions as an executable (in this case, the executable permission is a must), and inside the script, it must contains the location for the perl interpreter (which is what the first line of the code does)

But in the second form with perl leading the command: the file functions as mere an input parameter to feed 'perl' command. The true executable from OS point is 'perl' program itself.



www.perl.org

What do we learn from Exercise 2

- Importance of the first line of almost every Perl script (Perl Interpreter is mandatory to be present)
- This is why the path has to be specified in each Perl script to let the system know where to start (this is called 'Entry Point')



www.perl.org

Using Perl

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Command line Option Explained

- Command format:
perl `-[v|p|e|i]` “perl statement/expression” input
- Options: (type “perl -h” for more options)
 - e # tell perl to execute some statements in what is quoted following
 - v # check current perl version
 - i[extension] # edit input files in place (makes backup if extension supplied)
 - n # assume "while (<>) { ... }" loop around program
 - p # assume loop like -n but print line also



www.perl.org

Command line Examples

- `perl -e 'print "Hello World\n"'`
 - same result as run `codeEx_simplest.pl`
- `perl -n -e 'print "$. - $_" codeEx_simplest.pl`
 - implicit loop, print code with line number
- `perl -p -n -e '$_="$. - $_" codeEx_simplest.pl`
 - implicit loop, implicit print, , using `$_` new assignment
- `perl -ne 'print "$. - $_" unless /^#/' codeEx_simplest.pl`
 - implicit loop, print code with line number
- `perl -ne 'print "$. - $_" if /^#/' codeEx_simplest.pl`
 - print all lines that are starting with `#`



www.perl.org

Good Programming Practices

- Always starts with hash-bang line
`#!/usr/local/bin/perl`
- Using template/framework to standardize and simplify code tasks (see MyFramework.pl for explanation)
- Learn to using Perl debugger tool rather than use 'print'
- Start with minimum code required (isolate code)
- Reduce interference by defining good interfaces through subroutines
- Pay attention to format (especially with statement across multiple lines)
- Many more ... (refer to 'Perl Best Practice')

Good Programming Practices Code Example



www.perl.org

```
emacs@scc4.bu.edu
File Edit Options Buffers Tools Help

#
# : Amanda Yun Shen
# date: 06/24/2016
# purpose: serve as starting point for each script to set up all common elements (framework)
# usage:
#
#

##!/usr/bin/perl

# use 5.010;
# use strict;
use warnings;

use DBI;
use strict;
#use POSIX qw(strftime);
use POSIX qw(ceil);
use Getopt::Long;
use Get;
#use Lib 'C:/myLib/perl/'; # on WINDOWS
#use Lib '/usr2/calLab/yshen67/Lib/perl/'; # on SCC
use Tie::IxHash;
use My::DBUtil;
use My::Authen::DBAuth;
use enum qw( DB_0_DEV );

# my $debug = 1; #set debug flag

## define db connection param

my $host = 'scc-data01'; # bu.edu';
my $user = '';
my $passwd = '';

# get db authentication
MyAuthen::DBAuth($host);

#
# my @db = (
# { DBHOST=>$host, 'DBNAME'=>'scc_util_dev', 'DBUSER'=>$user, 'DBPWD'=>$passwd},
# { DBHOST=>$host, 'DBNAME'=>'yun_dev', 'DBUSER'=>$user, 'DBPWD'=>$passwd},
# );

### create db connection

my @db=();
for my $i (0..$#db) {
    $db[$i] = DBI->connect("dbi:mysql:$db[$i]->[DBNAME];host=$db[$i]->[DBHOST]", $db[$i]->[DBUSER], $db[$i]->[DBPWD], {RaiseError => 1});
}

# define global variables:
my ($ret, $query, $rm);
my %util = ();

### call main functions

### release db connections

for my $i (0..$#db) {
    $db[$i]->disconnect;
}

#####
# BEGIN of utility function
#####
sub ordered_hash_ref {
    tie my %hash, Tie::IxHash, @_;
    return \%hash;
}

--:--:-- MyFramework.pl Top L42 (Perl -3)-----
```



Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Variable Scope

- What is scope? The space that something is seen/valid
- Two types of scope: Global vs. Lexical
 - Global variable – visible in the entire package, ‘our’ keyword
 - lexical variable – only visible in the context, with ‘my’ keyword
- Override: Inside variable overrides(hides) the outside variable
- Package independence - same variable name can be used in different packages, they are totally independent and won’t affect each other
- Use namespace to provide specificity – use “package::variable”
qualifier

Variable Scope Example 1



www.perl.org

Variable scope: enclosing block

```
1. #!/usr/bin/perl
2. use strict;
3. use warnings;
4.
5. {
6.     my $email = 'foo@bar.com';
7.     print "$email\n";      # foo@bar.com
8. }
9. # print $email;
10. # $email does not exists
11. # Global symbol "$email" requires explicit package name at ...
```



www.perl.org

Variable Scope Example 2

Variable hidden by other declaration

```
1. #!/usr/bin/perl
2. use strict;
3. use warnings;
4.
5. my $fname = "Foo";
6. print "$fname\n";      # Foo
7.
8. {
9.     print "$fname\n";  # Foo
10.
11.     my $fname = "Other";
12.     print "$fname\n";  # Other
13. }
14. print "$fname\n";    # Foo
```

Variable Scope Example 3



www.perl.org

```
[yshen16@scc4 session1]$ more codeEx_varScope_namespace.pl
#!/usr/local/bin/perl
use strict;
use warnings;

package Calc;
use strict;
use warnings;

our $total = 100;

sub add {
    my $total=0;
    $total += $_ for (@_);
    return $total;
}

package main;

my $total = Calc::add(3, 4);
print "\$total in Main: $total\n";
print "\$total in Calc: ${Calc::total}\n";
[yshen16@scc4 session1]$
[yshen16@scc4 session1]$ perl codeEx_varScope_namespace.pl
"my" variable $total masks earlier declaration in same scope at codeEx_varScope_namespace.pl line 20.
$total in Main: 7
$total in Calc: 100
[yshen16@scc4 session1]$
```

Variable Scope Good Practice



www.perl.org

To avoid ambiguity –

- avoid using same name for different variables unless you are sure they are meant to be same thing ;
- use meaningful names for each variable



www.perl.org

Special Symbols

- Also called ‘pre-defined variables’ in perldoc
- Can be divided into five categories:
 - General Variables
 - Regular Expression Variables
 - Filehandle Variables
 - Error Variables
 - State Variables
- Perl programming depends highly on using these special symbols (variables, more officially). So it is good to know about them.
- Use ‘perldoc perlvar’ to read the help documentation



www.perl.org

Special Symbols - General

`$ARG/$_` – default input space

`@ARG/@_` – parameter array for subroutine

```
1. sub max {
2.     my $max = shift(@_);
3.     foreach $foo (@_) {
4.         $max = $foo if $max < $foo;
5.     }
6.     return $max;
7. }
8. $bestday = max($mon,$tue,$wed,$thu,$fri);
```

`$a` – small number in `sort()`; `$b` – large number in `sort()`

```
@all = sort { $b <=> $a } 4, 19, 8, 3;
@ordered = sort { $a->name cmp $b->name } @employees;
```

`%ENV` – environment variables

`%INC` – the paths to be searched

...



www.perl.org

Special Symbols – Regular Expression

\$1, \$2, ... - matching groups in the parentheses in pattern

```
1. my $outer = 'Wallace and Grommit';
2. my $inner = 'Mutt and Jeff';
3.
4. my $pattern = qr/(\S+) and (\S+)/;
5.
6. sub show_n { print "\$1 is $1; \$2 is $2\n" }
7.
8. {
9. OUTER:
10.     show_n() if $outer =~ m/$pattern/;
11.
12.     INNER: {
13.         show_n() if $inner =~ m/$pattern/;
14.     }
15.
16.     show_n();
17. }
```

Output:

```
1. $1 is Wallace; $2 is Grommit
2. $1 is Mutt; $2 is Jeff
3. $1 is Wallace; $2 is Grommit
```



Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Special Symbols – Regular Expression (2)

- `$&/${^MATCH}` – last successful matching string
- `$`/${^PREMATCH}` – the string preceding the last matching string
- `$'/${^POSTMATCH}` – the string following the last matching string

```
1.  local $_ = 'abcdefghi';
2.  /def/;
3.  print "$`:$&:$'\n";           # prints abc:def:ghi
```



www.perl.org

Special Symbols – File handlers

- \$ARGV – name of current file
- @ARGV – command line arguments
- ARGV – special file handle for command line filenames
- \$. – current line number
- \$/ - input line delimiter
- \$\ - output line delimiter
- \$% - current page number



www.perl.org

Special Symbols – File handlers

- `$@` Perl error string
- `#!` Error number from C, 'errno'
- `$$E` Extended OS error info, such as 'CDROM tray not closed'
- `$?` Exit status from last process

```
1.  eval q{
2.      open my $pipe, "/cdrom/install |" or die $!;
3.      my @res = <$pipe>;
4.      close $pipe or die "bad pipe: $?, $!";
5.  }
```



www.perl.org

Code Examples

BOSTON
UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Walk Through Code Examples

Examples To walk through: (code examples are in ./code/session1/)

1. bio_nts_trans.pl - example in real world to show regular expression in use
2. bio_prot_trans.pl – example in real world to show hash structure in use

Let's go to the terminal to go through these examples now.



www.perl.org



www.perl.org

Packages and Modules

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Purpose of Packages/Modules

- To address the complicity of software functionality, when single script is not sufficient and clear to provide the service.
- It's a way to organize code



www.perl.org



www.perl.org

What is Package

- ‘package’ – the term used for functionality, means a division of global namespace; can be spread across several files (modules);
- It’s a **logical** unit for code functionality;
- Declares the BLOCK or the rest of the compilation unit as being in the given namespace (Perldoc definition)
- **Package = Namespace** (simplified)
- Way Perl uses to implement ‘class’ (object-oriented)



www.perl.org

What is Module

- 'module' – a library file consists of a set of related methods;
- It can be used as 'class' definition or class implementation , or both (for example: Bio::SeqIO)
- modules are actual physical libraries stored in file system to implement desired functioning system
- the common practice is to organize them by their logical namespaces (package)



www.perl.org

Package vs Module - relationship

- Modern design of perl modules – one module one package
- object-oriented
 - hierarchically organized, so outer namespace could cover the inner namespace, to provide modularity
 - Module file directory reflects namespace hierarchy
 - well defined interfaces between modules (namespaces);
 - Two Examples, Bio::DB and Bio::SeqIO
 - Bio::DB – no common interface; every sub namespace is self-referenced
 - Bio::SeqIO – has common abstract interface defined (implemented), while inside every sub namespace related to certain SeqIO may refer to this common interface

BioPerl on SCC



www.perl.org

This is the first level file structure of BioPerl installed on SCC:

```
[yshen16@scc4 Bio]$ ls
Align                CodonUsage           LiveSeq              Perl.pm             SearchIO.pm         Symbol
AlignIO             Coordinate           LocatableSeq.pm    Phenotype           Seq                 Taxon.pm
AlignIO.pm          DB                  Location            PhyloNetwork       Seq.pm             Taxonomy
AnalysisI.pm        DBLinkContainerI.pm LocationI.pm        PhyloNetwork.pm    SeqAnalysisParserI.pm Taxonomy.pm
AnalysisParserI.pm  Das                 Map                PopGen              SeqEvolution        Tools
AnalysisResultI.pm DasI.pm             MapIO              PrimarySeq.pm      SeqFeature          Tree
AnnotatableI.pm    DescribableI.pm    MapIO.pm           PrimarySeqI.pm     SeqFeatureI.pm     TreeIO
Annotation          Draw               Matrix             PullParserI.pm     SeqI.pm            TreeIO.pm
AnnotationCollectionI.pm Event              MolEvol            Range.pm            SeqIO               UpdateableSeqI.pm
AnnotationI.pm      Factory            Nexml              RangeI.pm           SeqIO.pm            Variation
Assembly           FeatureHolderI.pm  NexmlIO.pm        Restriction         SeqUtils.pm        WebAgent.pm
Cluster            HandlerBaseI.pm    Ontology           Root                SimpleAlign.pm
ClusterI.pm        IdCollectionI.pm   OntologyIO         Search              SimpleAnalysisI.pm
ClusterIO          IdentifiableI.pm  OntologyIO.pm     SearchDist.pm      Species.pm
ClusterIO.pm       Index              ParameterBaseI.pm SearchIO             Structure
```

for full library structure, refer to : [doc/bioperl_structure.txt](#)



www.perl.org

Perl help system

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Perl Language Reference



www.perl.org

- This is the ultimate resource of authority – BLUEPRINT of a language;
- Access entrance:
 - <http://perldoc.perl.org/index-language.html>
- May be found too difficult to be understood for beginners



www.perl.org

'perldoc' utility

- Embedded Perl documentation system in 'POD' (Plain Old Documentation) format
- Mostly written for Perl library modules:

```
perldoc perldoc # how to use perldoc
perldoc perlintro # perl introduction for beginners
perldoc perltoc # Perl table of contents
perldoc perl # overview of Perl
perldoc perlfunc # Full list of Perl functions
perldoc -f print # help on built-in function called 'print'
perldoc perlop # full list of perl operators
```

many more ... (<http://perldoc.perl.org/perl.html>)

<http://perldoc.perl.org/index-language.html>



www.perl.org

A screenshot of the perldoc.perl.org website. The browser address bar shows 'perldoc.perl.org/index-language.html'. The page features a dark blue header with the Perl logo and 'perldoc.perl.org Perl Programming Documentation'. A sidebar on the left contains navigation menus for 'Perl version', 'Manual', 'Reference', and 'Modules'. The main content area is titled 'Language reference' and lists various Perl documentation topics such as 'perlsyn - syntax', 'perldata - Perl data types', 'perlsub - subroutine function', and 'perlref - reference pointer data structure structure struct'. A search bar is visible in the top right of the content area.



Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

'man' command

- Linux 'man' command can be used to access perl module help, for example:

man perl

man perldoc

man perltoc

man perlre

...

- 'perldoc' is recommended over 'man' – 'man' depends on if the man pages are installed for certain Perl Modules or not

Get Help – online resources

Websites:

<https://learn.perl.org/tutorials/>

<https://perlmaven.com/>

<http://perlmonks.org/>

<https://www.tutorialspoint.com/perl/>

<http://stackoverflow.com/>

Books: (for more refer to [perlbook_list.txt](#))

<https://www.perl.org/books/beginning-perl/>

<http://docstore.mik.ua/oreilly/perl/cookbook/>



www.perl.org



www.perl.org

Perl debugger

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

perl -d

- Use 'perl -d scriptname' to start debugger
- Perl debugger is a fully integrated part to Perl interpreter, that means code must first pass the compiling process to be able to use debugger
- Frequently used debugger commands:

h: type the help information

n: execute next statement

s: single step execution

r: start/restart/continue run the code

b: set breakpoints

v: view source code in the context



www.perl.org

Data::Dumper

- Perl module commonly used to print out the variable structure and value; but more convenient
- Usage:

```
use Data::Dumper qw(Dumper);
```

```
print Dumper \@an_array;  
print Dumper \%a_hash;  
print Dumper $a_reference;
```

Data::Dumper Code Example



www.perl.org

```
[yshen16@scc4 session1]$ more codeEx_useDumper.pl
#!/usr/local/bin/perl
use 5.010;
use strict;
use warnings;
use Data::Dumper qw(Dumper); # this is the custom module added for this particular purpose

# in Perl
#
my $a = 1;
my %b_hash = (
    1 => 'apple',
    2 => 'pearl',
    3 => 'orange',
);
my @c = sort keys %b_hash;

print "Here is the data in these variables:\n";
print Dumper $a;
print Dumper \%b_hash;
print Dumper \@c;
[yshen16@scc4 session1]$ perl codeEx_useDumper.pl
Here is the data in these variables:
$VAR1 = 1;
$VAR1 = {
    '1' => 'apple',
    '3' => 'orange',
    '2' => 'pearl'
};
$VAR1 = [
    '1',
    '2',
    '3'
];
[yshen16@scc4 session1]$
```



www.perl.org

Q & A

**BOSTON
UNIVERSITY**

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017