An introduction to Perl

W. H. Bell
W.Bell@cern.ch

Abstract

A basic overview of the `perl` scripting language is given. The language is introduced using examples covering a few aspects at a time. Reference tables are provided for commonly used `perl` functionality.

Perl - Introduction

There are lots of manual pages:

```
[wbell@ppepc54 ~]$ man perl
```

For ease of access, the Perl manual has been split up into several sections. (The way the manual page sections is divided depends on the version of Perl)

For the less man page adept there is also a web site: http://www.perl.com/pub/q/documentation which presents the same information.

Perl - Overview

- A basic program
- Syntax
  - Loops
  - Conditional Statements
  - Functions
- Variable Types
  - Scalars
  - Arrays
- Files Handles and their Usage
  - Reading and Writing
  - Using Commands
- Command Line Arguments
- String Manipulation
  - Pattern Matching
  - Subsitution
  - Splitting and Joining

Perl - A Basic Program

```
#!/usr/bin/perl

print STDOUT "In the beginning\n";
```

example_01.pl: A basic perl program

- The first line: `PATH` to executable.
- The second line: print via the standard out.

For Loops

```
# for loop
for ($i=1; $i<=2; $i++) {
  print STDERR "for loop $i\n";
}

# for loop 2
@fruits = ("apple", "pear", "plumb");

for my $fruit (@fruits) {
  print STDOUT "$fruit ";
}
print STDOUT "\n";

# for loop 3
for (1..2) {
  print STDOUT "Loop $_\n";
}
```

example_02.pl: A perl program demonstrating different types of loops.

- The C style is slower than the foreach style of for loop.
- Try to code in the Perl paradigm

More Loops

```
while (condition) {
statement;
}

do {
statement;
} until condition;

do {
statement;
} while (condition);
```

- The while loop is implemented within **example_02.pl**.

Conditional Statements: if

```
#!/usr/bin/perl

for (1..3) {
  if($_==1) {
    print STDOUT "1, ";
  }
  elsif($_==2) {
    print STDOUT "2, ";
  }
  else {
   print STDOUT "3!\n";
  }
}
```

example_03.pl: A program demonstrating an `if`, `elsif`, `else` statement.

- Note the usage of the $_ variable.
- If the output of an argument is not assigned to a variable it is implicitly assigned to $_.

Conditional Statements: die, unless

```
#!/usr/bin/perl

for (1..5) {
  die "It’s time to die." unless ($_<4);
  print STDOUT "$_\n";
}
```

example_04.pl: A simple program to illustrate the use of die, unless conditional statements.

- The `die` message is printed together with a line number.
- Useful for critical checks, e.g. opening files etc.

Functions

```
#!/usr/bin/perl

printStrings(1,2,3);

sub printStrings {
  my ($a, $b, $c) = @_;

  print STDOUT "a=$a, b=$b, c=$c\n";
}
```

example_05.pl: A simple script to demonstrate basic function usage.

- The usage of `@_` is similar to that of `$_`. The values are assigned implicitly.
- Warning: passing arrays in this way can be tricky.

File I/O - Writing

```
$outputfile = "perl_output";

die ("Can't open $outputfile for writing\n")
  unless open(OUTPUT, "> $outputfile");

print OUTPUT "Pythia 6.2\n";
print OUTPUT "EvtGen On\n";
print OUTPUT "Geant v4\n";

close(OUTPUT);
```

example_06.pl: A program that illustrates usage of file i/o. [The writing section.]

- The file handle works in the same way as the STDOUT handle.

File I/O - Reading

```
die ("Can't open $outputfile for reading\n")
  unless open(INPUT, "< $outputfile");

while (<INPUT>) {
  chomp;
  if(defined $_) {
    print STDOUT "$_\n";
  }
}
close(INPUT);

unlink($outputfile);
```

example_06.pl: A program that illustrates usage of file i/o. [The reading section.]

- The value following the file handle can be many different things. It behaves like a commnand line, i.e. pipe and redirection can be used.

Array Manipulation

```
...
my @run_numbers;
print STDOUT "\@run_numbers last index=";
while ($run = <RUNFILE>) {
  chomp($run);
  push(@run_numbers,$run);
  print STDOUT "$#run_numbers... ";
}
print STDOUT "\n\n";

print STDOUT "Run Numbers\n";
while ($#run_numbers >= 0) {
  $run=shift(@run_numbers);
  print STDOUT "$run\n";
}
print STDOUT "\n";
```

example_07.pl: A program to demonstrate dynamic array management

- push and pop, add and remove from the end of the array.

- `unshift` and `shift`, add and remove from the beginning of the array.

Using Commands

```
# find files which have been modified in the last 60 minut
# the output as a file input stream.
open(FINDFILE,  "find    ./ -cmin -60 |");

# While there are files returned from the find command
while (<FINDFILE>) {
  chomp;
  if(defined $_) {
     print STDOUT " $_ was in the last 60 minutes.\n";
  }
}
close(FINDFILE);
```

example_08.pl: A program to demonstrate the usage commands piped into a file input stream

- Any executable can be run in a similar manner.

Command Line Arguments

```
$numArgs = $#ARGV + 1;
print STDOUT "\n $0 with $numArgs command-line arguments\n

foreach $argnum (0 .. $#ARGV) {
   print STDOUT "\$ARGV[$argnum] = $ARGV[$argnum], ";
}

print STDOUT "\n\n";
```

example_09.pl: A program demonstrating usage of command line arguments

- Arguments are accessed via `@ARGV`.
- `$#ARGV` index of last argument in array.
- Executable name via $0

Pattern Matching

```
open(FILELIST,  "ls |");

while ($file = <FILELIST>) {
  chomp($file);
  if(defined $file) {
    if($file =~ /ex[1-4]\.pl/) {
      print STDOUT " $file \n";
    }
  }
}
close(FILELIST);
```

example_10.pl: A script to demonstrate pattern matching with regular expressions.

- Matches file containing ex followed by one of 1,2,3,4 and `.pl`. (If the file was called fedex1.pl it would match too.)

Pattern Matching

```
open(FILELIST,  "ls |");

while ($file = <FILELIST>) {
  chomp($file);
  if(defined $file) {
    if($file =~ /^ex\d.[\w]/) {
      print STDOUT " $file \n";
    }
  }
}
close(FILELIST);
```

example_11.pl: A program to demonstrate using an anchor and decimal and word

- The program only selects files beginning with `ex` followed by a single number character and then any word extension e.g. `pl` or `sh`.

Substitution

```
@al = ('a','b','c','d','e','f','g','h','i','j','k','l','m'
       'o','p','q','r','s','t','u','v','w','x','y','z');

@AL = ('A','B','C','D','E','F','G','H','I','J','K','L','M'
       'O','P','Q','R','S','T','U','V','W','X','Y','Z');

open(LISTOF,"ls |");

while (<LISTOF>) {
  chomp;
  if(defined $_) {
    $i = 0;
    while ($i<26) {
      s/$al[$i]/$AL[$i]/g;
      $i = $i + 1;
    }
    print STDOUT "$_\n";
  }
}
close(LISTOF);
```

example_12.pl: A program to demonstrate substitution.

Splitting and Joining

```
while (<TABFILE>) {
  chomp;
  if(defined $_) {
    @data = split(/\t/,$_);
    $data_line = join(',',@data);
    print CVSFILE "$data_line\n";
  }
}

close(TABFILE)
```

example_13.pl: A program to demonstrate the split and join.