

BASIC PERL
FOR
BIOLOGISTS

Angelica Lindlöf
2010

Angelica Lindlöf

Basic Perl for Biologists

Contents

Who is this compendium for?	4
Perl as a Programming Language	5
1 Introduction.....	6
1.1 Conventions.....	6
1.2 Installing Perl on your computer	6
2 Problem solving.....	8
2.1 Programming	8
2.2 Procedural Programming.....	8
3 'Hello World' – your first program	10
3.1 Store your program files	10
3.2 'Hello World!'	10
Exercises	12

Who is this compendium for?

This compendium is mainly meant for biologists who wish to gain basic knowledge about programming in Perl. It is not meant to be an exhaustive teaching book about the Perl language, but rather give the reader an entrance to Perl, so that she/he can write small, useful script on her/his own in order to analyze biological data. Additionally, to also make it possible in an extended way communicate with programmers.

Perl as a Programming Language

Perl was developed in the end of the 1980s by Larry Wall, in an attempt to combine the best parts from the tools in UNIX, e.g. awk, sed and sh. The purpose of Perl was to support UNIX users with commonly occurring tasks, which was too complicated to do with languages such as C and awk.

Perl stands for “Practical Extraction and Report Language” or “Pathologically Eclectic Rubbish Lister”, both invented by Larry Wall himself. The language was continuously developed by Larry Wall when users requested more things they wanted to do with it.

Perl is free and can be downloaded from the Internet. It is distributed under the GNU Public License, which basically means if you develop new binaries or modify the binaries in Perl’s library you need to share them with the rest of us.

Perl is both a compiler and an interpreter; it is a compiler because the entire program is read and interpreted before the first statement is executed and it is an interpreter because no object code files (like in C or C++) are created. It is also called a script language.

One of the reasons for Perl being so popular is because it has a very good support for regular expressions, which makes it easy for searching and formatting text files. It is also a very easy-to-learn language and yet very complex, which means that the user often can do advanced tasks in a very short period of learning time.

There are different versions of Perl available; such as Perl4 is version 4 in the development and Perl5 version 5. The latest version is an updated and improved version of the earlier ones. One should always try to install the latest version, since older versions can be unsupported and out-of-date. It is easy to check which version that is installed on your system, simply by stating

```
%>perl -v
```

in your terminal window.

Perl runs on most computers and operating systems such as Macintosh, Windows and UNIX. There is a FAQ site on <http://www.perl.com> that could come in handy. There are also plenty of Usenet groups and useful tutorials available. If you encounter problematic errors or is looking for a special function, a search on the Internet is recommended as a first action.

1 Introduction

This chapter introduces you to the conventions used in the compendium, but also gives an introduction to what a program is, what is procedural programming, how to install the program in Unix and Windows, and ends by giving a small example of a program.

1.1 Conventions

The conventions used in the book are the following:

Courier new	Used for code examples
%><command>	Command that should be typed in the terminal window are preceded with %>
	For example, <enter> means you should push the enter-button on the keyboard.

1.2 Installing Perl on your computer

The first thing to do before trying to install Perl is to check whether it is already installed or not. In a Unix/Linux, MacOS X or Windows environment you can do this by typing the following in a terminal window or using the Command Prompt in Windows:

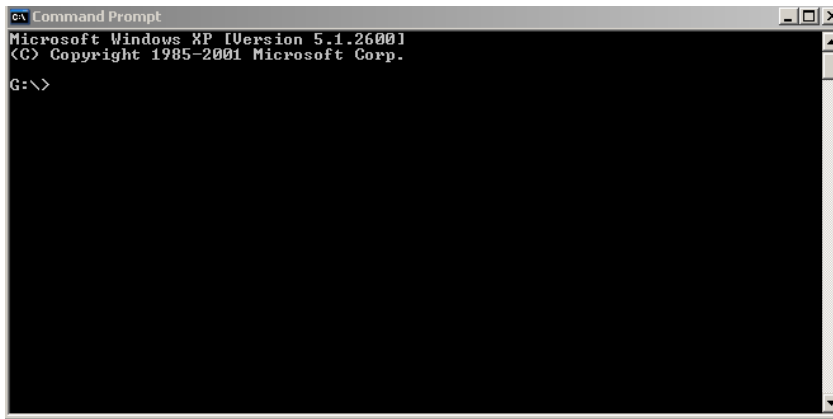
```
%>perl -v
```

The command will either give you information on the version installed on your computer or an error message saying something like 'Command not found', which means it is probably not installed. It could, however, still be installed if you are in a Unix/Linux environment. The path to the program may not have been set properly. In order to find that out, the best thing is to contact the system administrator for help.

To start a terminal window in Windows the simplest way is to go via the Start menu:

```
Start → All programs → Accessories → Command prompt
```

and you should get a program/window looking like this:



```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
Copyright 1985-2001 Microsoft Corp.

G:\>
```

In this window you can now print 'perl -v' and see whether or not the program is installed.

If it is not installed, the next thing is to download the program from the Internet. The web site to visit for getting the program is <http://www.perl.com>. Here you can find download pages that will guide you through the installation of Perl. It is always recommended that you use binary versions of the program, since these are easiest to install. Source code should only be used if you are more familiar with installing programs and want some special options installed for Perl.

For Macintosh user more information about installing can be found on the MacPerl web site, <http://www.macperl.com> and for Windows user ActivePerl is the one to use, which can be found on the web site <http://www.activestate.com/ActivePerl/>.

2 Problem solving

2.1 Programming

A program is simply a text file with instructions to the computer, such as where to get the input data from, what to do with the data and where to put the results. For example, we want to read a file of FastA formatted sequences and output all ID lines. The instructions for doing so can be put into a text file, which is then called upon from the terminal window and executed by the compiler, in this case the Perl compiler installed on your computer. The instructions, also known as statements, in the text file are called programming code or simply the code.

A Perl program is a text file with statements and definitions written in the programming language Perl. There are other programming languages such as Java, C and C++, all with their own specific syntax and specialties. Perl supports mainly procedural programming, but has some functionality for doing object oriented programming. However, object oriented programming is not covered in this book.

2.2 Procedural Programming

Procedural programming languages are based on the use of procedures (functions/subroutines/methods), which consist of a series of computational steps that should be executed. The programmer need to specify which procedures that should be executed and in which order. The procedures can be called upon from anywhere in the program and also be called from other functions or by itself (recursion).

Procedural programming languages are based on three different control structures; *sequence*, *selection* and *iteration*. The sequence simply specifies in which order the statements should come, e.g., should the program read in the sequence data first and then search for the ID lines, or search for the ID lines before the data has been read? In this example, the sequence is quite obvious - the data has to be read before the search can be made. However, in larger, more complex programs the sequence may not be so obvious and there are times when the statements come in the wrong order, which will create a mall-functioning program.

The selection is a conditional structure consisting of a sequence of statements. The structure will only execute the group of statements if the conditional test attached to the structure is fulfilled. Otherwise the structure will be skipped and the compiler moves on the next statement in the program. In our example, the selection is whether to print out a line or not. We are only interested in ID lines, i.e. lines starting with '>', all other lines should be skipped. Here, the selection is the choice of two things, print out a line starting with '>' or skip the line. But, the selection can consist of more than two options, it could be three, four, five... or any number of choices. Moreover, in procedural programming only the first true condition is executed, i.e., the first choice for which the condition is fulfilled. For example, if we have the conditions and thereby options: "if line starts with '>' " or "if line do not start with '>' ", whenever we encounter a line starting with '>' this

option will be chosen and the second one ignored. However, when the line does not start with '>' the program will move on to the next option and condition. And if this condition is true, then the option coupled to this condition will be executed.

The iteration, also known as a loop, is a structure that repeats a group of statements until the attached conditional test fails. In the example, it could be to print out all ID lines until end of file (eof), i.e., iterate over all lines in the file until no more lines are available and we have reached the end of the file.

3 'Hello World' – your first program

In this chapter you will learn how to write your first program and be introduced to some Perl code. You will soon see that it is not so difficult to program in Perl, but it is rather intuitive.

3.1 Store your program files

Before you start programming it is always good to keep track of your created programs by making a directory or folder to store them in. For example, you could name the folder 'Perl course' or something like that. Thereafter, start a terminal window or a command prompt and move your way to the new directory by using

```
%cd path/Perl course          or
%cd C:\path\Perl course      if you are using Windows
```

'path' specifies exactly where the folder is located on your computer. From this folder you will start your saved Perl programs.

3.2 'Hello World!'

In order to write a program you need to start a text editor, where you will write the programming code. In Unix/Linux nedit or Emacs are good editors and for Windows the Crimson Editor is a very good text editor that is free downloadable from <http://www.crimsoneditor.com>. In Windows Notepad can also be used as a good start for writing programs in.

In whatever the text editor you choose, type the following:

```
#!/usr/bin/perl -w
print "Hello World!";
```

and save the file with giving it some name and with the extension '.pl', which specifies that it is a Perl program. For example, you could name it 'world.pl'. Make sure you save it in the correct folder (e.g., the Perl course folder you created previously). Thereafter try to run the program by typing the following the terminal window or command prompt:

```
%perl world.pl
```

This is a program with one statement, which simply says the string "Hello World!" should be printed out. After execution of the program the following should therefore appear in the terminal window

```
%>Hello World!
```

In the program the first line indicates that it is a Perl program and not a shell program or from some other programming language. *This line should always be the first line in your programs.* Normally, a line starting with "#" is treated as a

comment by Perl and ignored by the interpreter. However, in combination with the '!', instructions are given to the operative system on how the file should be run. The sentence '/usr/bin/perl' gives information on where the interpreter is located, i.e., where the program that should execute the rest of the lines is located. The '-w' is an option passed on to the interpreter that says additional warning messages should be turned on. In Windows the line is not required; however, it is of good programming style to always include this line. *The ';' marks the end of a statement and blanks in the beginning of a line is ignored.*

The second line in the program says that the sentence between the double-quotation marks should be printed out, i.e., 'Hello world'. The statement before the sentence is a call to the in-built function print(), which gives instructions on how and where to print out the sentence. By default the sentence will be printed out in the terminal window (Command prompt).

Like in all programming languages there is a possibility to add comment lines, which will not be executed by the compiler. In Perl comments are preceded by '#':

```
#!/usr/bin/perl -w
#String that prints out Hello World!
print "Hello World!\n";
```

Everything on line 2, between '#' and the end of the line, is treated as a comment and ignored by Perl.

Exercises

301

Now try change the sentence that should be printed out. For example, change it to 'Today is a good day to start programming!'.

302

Try to add a comment line in the program and see whether it is printed out or not.

303

Try to add a command that will print out a new line after the sentence. This is done by adding the escape sequence '\n' to the sentence:

```
print "Hello World!\n";
```

304

Try to make the programme print out the following:

```
In August I will the start the master's programme in Bioinformatics.  
I look forward to go to Sweden and begin my studies.
```