

A primer on Perl programming

Andrea Telatin

Bioinformatics lab at the CRIBI

VI floor (east)

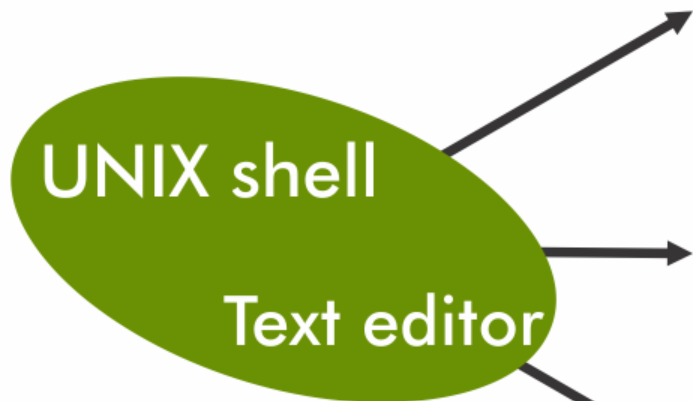
andrea.telatin@cribi.unipd.it



Bioinformatics: tools of the trade

Environment

Tools



Scripting language
(e.g. Perl, Python)

Data elaboration

Relational database
(e.g. MySQL,...)

Data storage and
retrieval

Web interfaces
(HTML)

Simple access to
data



Remember the `ls`, `cd` and `cp` commands from the first-year course?

```
#!/usr/bin/perl

$v = $PERL_VERSION;

print "Perl version = $v\n";

$x = &getnumber;
$y = &getnumber;
$z = 123;
$t = $x + $y + $z;

print "$x + $y + $z = $t\n";

# The program ends there...
# the subroutine is just tacked on at the end:

sub getnumber (
    print "Type in a number: ";
    $number = <>;
    chop($number);
    $number;
)
```

Linux:
gedit or kate

Mac OS X:
Text-wrangler

Windows:
Crimson Editor

What we should learn in the future

Use **Linux** or **Mac**: they have a powerful **terminal** with a standard set of commands (POSIX)

Learn a scripting language, and **Perl** is just perfect for bioinformatics.

Then:

- how to use simple relational databases (MySQL)
- a statistical package (R)
- HTML/CSS/Javascript to make websites (as a job?)



What is BioPerl? Why Perl «saved» the Human Genome Projects?

Genome shotgun assembly

Reference sequence (G=76):

THE-QUICK-BROWN-FOX-JUMPED-OVER-THE-LAZY-DOG-THAT-JUMPED-OVER-THE-OLD-ROCK.



Shotgun (L=5, N=114)

THE-QU K-BROW FOX-JU ED-OVE THE-LA -DOG-T T-JUMP -OVRR- IE-OLD-
HE-QCI -BROWN OX-JUM D-OVER HE-LAZ DOG-TH -JUMPE OVER-T :-OLD-R
E-QUIC BROWN- X-JUMP -OVER- E-LAZY OG-THE JUMPED VER-TH OLD-RC
-QUICK BROWN-F -JUMPE OVER-T -LAZY- G-THAT UMPED- ER-THE)LD-ROC
QUICK- AWN-FO JUMPED VER-TH LAZY-D -THAT- MPED-O R-THE- .D-ROCK
UICK-B WN-FOX UMPED- ER-THE AZY-DO THAT-J PED-OV [-THE-O)-ROCK.
ICK-BR N-FOX- MPED-O R-THE- ZY-DOG HAT-JU AD-OVE THE-OL
CK-BRO -FOX-J PED-OV -THE-L Y-DOG- AT-JUM D-OVER 'EH-OLD
K-BROW FOX-JU ED-OVE THE-LA -DOG-T T-JUMP -OVER- IE-OLD-
-BROWN OX-JUM D-OVER HE-LAZ DOG-TH -JUMPE OVER-T -OLD-R
BROWN- X-JUMP -OVER- E-LAZY OG-THA JUMPED VER-TH OLD-RO



Assembled «contigs»



Scaffolding with «mate paired» reads

Large fragments sequenced from both ends:

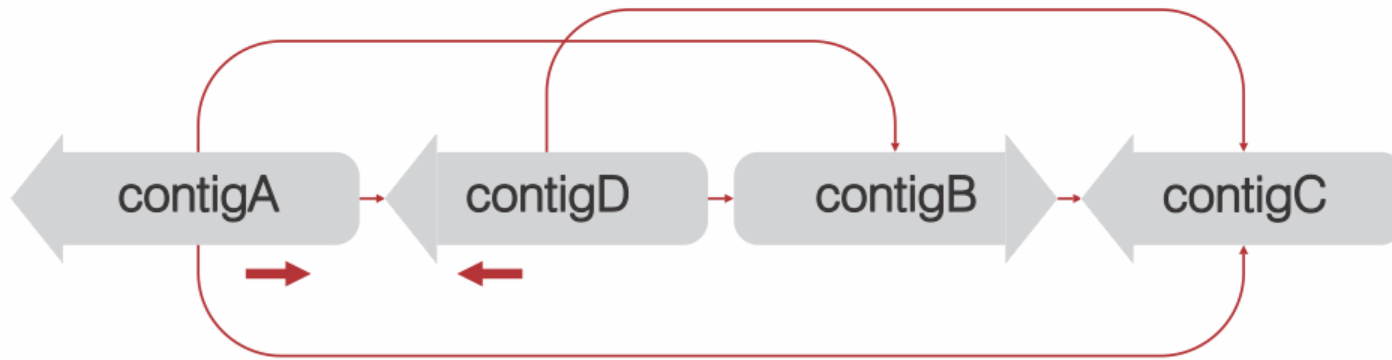


One scaffold:



What about the *strand* of the 3 contigs?

Project overview



Grand goal:

Given a gap design two primers to amplify the region of interest.

How do we do this by hand?

How can we automate this?

Perl is a glue language

External programs (talking with simple text):

Command line primer picker (Primer3)

Command line BLAST

What has our program to do?

And now let's start Perl

1) Get it (with Linux and Mac OS X is already there), for Windows you can download the **ActiveState Perl** package.

2) Open the terminal (cmd for Windows) and type «**perl -v**» to see if it's there.



This is your first Perl homework: having it running!

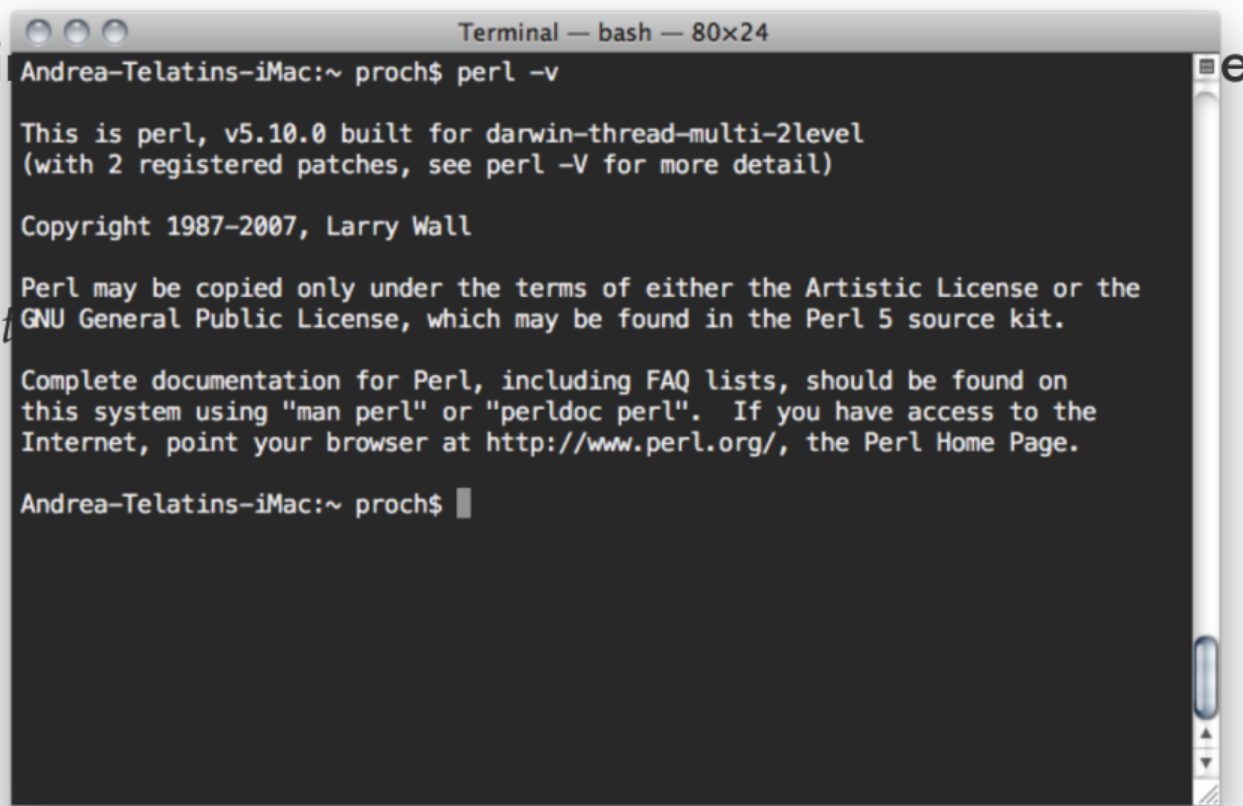
And now let's start Perl

1) Get it (with Linux and Mac OS X is already there), for Windows you can download the **ActiveState Perl** package.

2) Open the terminal if it's there.



This is your first



```
Terminal — bash — 80x24
Andrea-Telatin-iMac:~ proch$ perl -v

This is perl, v5.10.0 built for darwin-thread-multi-2level
(with 2 registered patches, see perl -V for more detail)

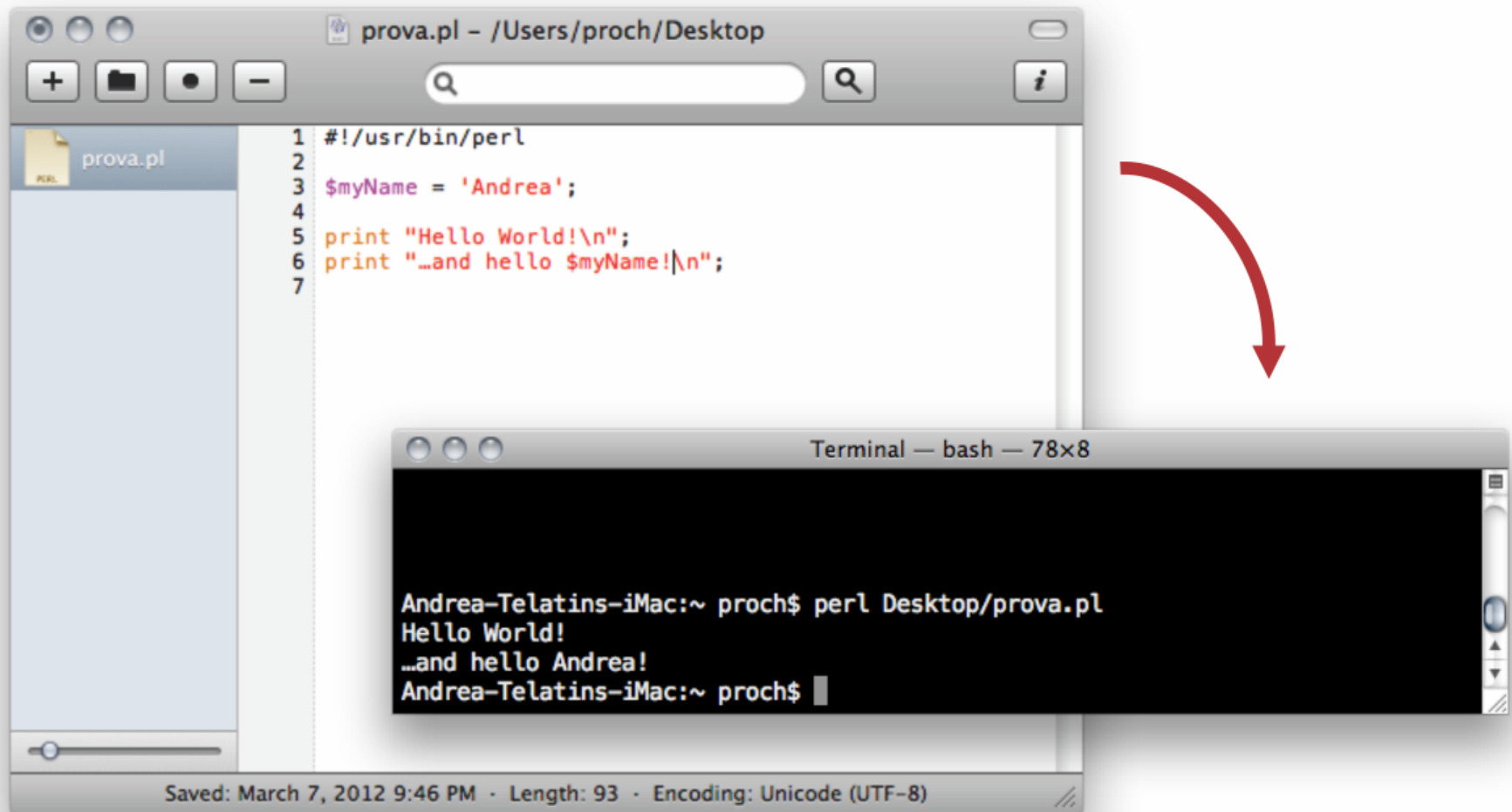
Copyright 1987-2007, Larry Wall

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.

Andrea-Telatin-iMac:~ proch$
```

And now let's start Perl



```
1 #!/usr/bin/perl
2
3 $myName = 'Andrea';
4
5 print "Hello World!\n";
6 print "...and hello $myName!\n";
7
```

```
Andrea-Telatin-iMac:~ proch$ perl Desktop/prova.pl
Hello World!
...and hello Andrea!
Andrea-Telatin-iMac:~ proch$
```

Saved: March 7, 2012 9:46 PM - Length: 93 - Encoding: Unicode (UTF-8)

Variables are post its with a name

Perl has three kind of variables: scalar, arrays and hashes.

```
$variable = 10;  
$name = 'MyName';
```

```
@array = ('list', 'of', 'variables');
```

```
%hash = ('jack' => 15, 'rudy' => 23, 'momi' => 9);
```

The «=**=**» sign **assigns** a value to a variable.

```
$number = 15;  
$string = 'People:';  
$newstring = '$string: $number';  
$newstring = $string.$number;  
$newnumber = $number^10;
```

Variables are post its with a name

A script using an array and accessing its values:

```
my @fruits = ("apples", "bananas", "cherries");

print "Fruit flies like $fruits[1].\n";
print "Life is like a bowl of $fruits[$#fruits].\n";
print "We need more $fruits[-3] to make the pie.\n";

$fruits[0] = "oranges"; # Replace apples with oranges
```

Output:

```
Fruit flies like bananas.
Life is like a bowl of cherries.
We need more apples to make the pie.
```

Operators and IF statement

Numerical operators as +, -, /, %, ^
and string operators as the . we saw before.

```
$number = 10*30-3;  
$power  = $number^4;
```

The IF statement perform a set of actions only if a condition is TRUE

```
if ($number == 10) {  
    print 'Bingo!';  
}
```

```
if ($number < 10) {  
    print 'Small!';  
} else {  
    print 'Ok!';  
}
```