分子生物计算
（*Perl* 语言编程）

天津医科大学
生物医学工程与技术学院

2016-2017 学年上学期（秋）
2014 级生信班

第 10..13 章　GenBank、PDB、BLAST、其他

伊现富（Yi Xianfu）

天津医科大学（TIJMU）
生物医学工程与技术学院

2016 年 12 月

```
1  if ( $line =~ /^\/\/\n/ ) {
2    last;
3  }
4
5  if ( $line =~ m!//\n! ) {
6    last;
7  }
```

```perl
#!/usr/bin/perl

use warnings;

"AAC\nGTT" =~ /^.*$/;
print "Without /m:\n", $&, "\n";
#Without /m:
#Use of uninitialized value $& in print at
  XXX.pl line N.

"AAC\nGTT" =~ /^.*$/m;
print "With /m:\n", $&, "\n";
#With /m:
#AAC
```

```perl
1  #!/usr/bin/perl
2
3  use warnings;
4
5  "AAC\nGTT" =~ /^.*$/;
6  print "Without /s:\n", $&, "\n";
7  #Without /s:
8  #Use of uninitialized value $& in print at
     XXX.pl line N.
9
10 "AAC\nGTT" =~ /^.*$/s;
11 print "With /s:\n", $&, "\n";
12 #With /s:
13 #AAC
14 #GTT
```

```perl
#!/usr/bin/perl

use strict; use warnings;

my $alphabet = join "", 'a' .. 'z';
$alphabet =~ /k(lmnop)q/;
print $1, "\n\n";
#lmnop

$alphabet =~ /(((a)b)c)/;
print "First pattern = ",  $1, "\n";
print "Second pattern = ", $2, "\n";
print "Third pattern = ",  $3, "\n";
#First pattern = abc
#Second pattern = ab
#Third pattern = a
```

```perl
1  my $save_input_separator = $/;
2
3  $/ = "//\n";
4  $record = <GBFILE>;
5
6  $/ = $save_input_separator;
```

```
1  for (;;) {
2      for ($curpos = tell(FILE); $_ = <FILE>;
   $curpos = tell(FILE)) {
3          # search for some stuff and put it
   into files
4      }
5      sleep($for_a_while);
6      seek(FILE, $curpos, 0);
7  }
```

```perl
#!/usr/bin/perl
use strict; use warnings;
list_recursively('pdb');

sub list_recursively {
    my ($directory) = @_;
    my @files = ();
    unless ( opendir( DIRECTORY, $directory ) ) {
        print "Cannot open directory $directory!\n";
        exit;
    }
    @files = grep ( !/^\.\.?$/, readdir(DIRECTORY) );
    closedir(DIRECTORY);
    foreach my $file (@files) {
        if ( -f "$directory/$file" ) {
            print "$directory/$file\n";
        }
        elsif ( -d "$directory/$file" ) {
            list_recursively("$directory/$file");
        }
    }
}
```

```perl
#!/usr/bin/perl

use strict;
use warnings;
use File::Find;
#perldoc File::Find

find( \&my_sub, ('pdb') );

sub my_sub {
    -f and ( print $File::Find::name, "\n" );
}
```

教学提纲

```perl
1  while(<>)  {
2    /^ATOM/ or next;
3
4    my($n, $x, $y, $z, $element)
5      = ($_ =~ /^.{6}(.{5}).{19}(.{8})(.{8})
     (.{8}).{22}(..)/);
6
7    $n       =~ s/^\s*//;
8    $element =~ s/^\s*//;
9
10   if (($n == 1) or ($n == 1078)) {
11     printf "%8.3f%8.3f%8.3f %2s\n", $x, $y,
     $z, $element;
12   }
13 }
```

```
1  my $first  = '3.14159265';
2  my $second  = 76;
3  my $third = "Hello world!";
4
5  printf STDOUT "A float: %6.4f An integer: %-5
     d and a string: %s\n", $first, $second,
     $third;
6  #A float:  3.1416 An integer: 76    and a
     string: Hello world!
```

```perl
#!/usr/bin/perl
use strict; use warnings;
my $DNA = 'AAACCCCCGGGGGGGGTTTTTT';
for ( my $i = 0 ; $i < 2 ; ++$i ) {
    print <<HEREDOC;
     On iteration $i of the loop!
    $DNA

HEREDOC
}
#     On iteration 0 of the loop!
#    AAACCCCCGGGGGGGGTTTTTT
#
#     On iteration 1 of the loop!
#    AAACCCCCGGGGGGGGTTTTTT
#
```

```perl
1  #!/usr/bin/perl
2  use strict; use warnings;
3  my $id          = 'A0000';
4  my $description = 'Highly weird DNA.  This DNA is so
     unlikely!';
5  my $DNA = '
     AAAAAACCCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGTTTTTTTTTTTTTTTTTTT
     ';
6  # Define the format
7  format STDOUT =
8  # The header line
9  >@<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<...
10 $id,        $description
11 # The DNA lines
12 ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<~~
13 $DNA
14 .
15 # Print the fasta-formatted DNA output
16 write;
```

```
1 >A0000      Highly weird DNA.  This DNA is so un...
2 AAAAAACCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGGGGTTTTTTTT
3 TTTTTTTTTTTTT
```

```perl
1  my $filename = $ARGV[0];
2  my $stride = '/usr/local/bin/stride';
3  my $options = '';
4  # 捕获输出
5  my @results = `$stride $options $filename`;
6  my $now = `date`;
7  my @functions = qw{ int rand length };
8  my %about;
9  foreach (@functions) {
10   #$about{$_} = `perldoc -t -f $_`;
11   $about{$_} = qx(perldoc -t -f $_);
12 }
13
14 # 不捕获输出，返回值为程序退出状态
15 system "$stride $options $filename";
16 system 'date';
17 system 'tar', 'cvf', $tarfile, @dirs;
```

```perl
#!/usr/bin/perl

if ( 10 / 3 == ( ( 1 / 3 ) * 10 ) ) {
    print "Success!\n";
}
else { print "Failure!\n"; }
#Failure!

if ( abs( 10/3 - ( ( 1/3 ) * 10 ) ) < 1e-10 ) {
    print "Right!\n";
    print "E=", abs(10/3 - ( (1/3) * 10 )), "\n";
}
else { print "Wrong!\n"; }
#Right!
#E=4.44089209850063e-16
```

```
1  use Bio::SeqIO;
2
3  my $string = ">SEQ1\nacgt\n>revseq1\ntgca ";
4  my $format = "fasta";
5
6  my $stringfh = IO::String->new($string);
7  open($stringfh, "<", $string) or die "Could not open
     string for reading: $!";
8
9  my $seqio = Bio::SeqIO-> new(-fh     => $stringfh,
10                              -format => $format,
11                              );
12
13 while( my $seq = $seqio->next_seq ) {
14     print $seq->id . ' = ' . $seq->seq() . "\n";
15 }
```

```perl
use strict; use Bio::SeqIO;
my $input_file = shift;
my $seq_in = Bio::SeqIO->new( -format => 'embl',
                              -file   => $input_file,
                            );
my @seq_array;
while ( my $seq = $seq_in->next_seq() ) {
    push(@seq_array,$seq);
}
@seq_array = sort { $a->length <=> $b->length }
  @seq_array;
my $total = 0;
my $count = 0;
for my $seq ( @seq_array ) {
    $total += $seq->length;
    $count++;
}
print "Mean length ", $total/$count, " Median ",
      $seq_array[$count/2]->length, "\n";
```

```perl
1  use Bio::SeqIO;
2
3  my $usage = "x2y.pl infile informat outfile outformat ";
4  my $infile = shift or die $usage;
5  my $informat = shift or die $usage;
6  my $outfile = shift or die $usage;
7  my $outformat = shift or die $usage;
8
9  my $seq_in = Bio::SeqIO->new( -file   => "$infile",
10                               -format => $informat,
11                             );
12 my $seq_out = Bio::SeqIO->new( -file   => ">$outfile",
13                                -format => $outformat,
14                             );
15 while (my $inseq = $seq_in->next_seq) {
16     $seq_out->write_seq($inseq);
17 }
```

BioPerl

```perl
use Bio::SeqIO;
my $usage = "splitgb.pl infile "; my $infile = shift or die $usage;
my $inseq = Bio::SeqIO->new( -file  => "<$infile", -format => 'Genbank',);
my %outfiles = ( human => {
                        Genbank => Bio::SeqIO->new(
                                                  -file   => '>human.gb',
                                                  -format => 'Genbank',
                                                  ),
                        Fasta   => Bio::SeqIO->new(
                                                  -file   => '>human.fa',
                                                  -format => 'Fasta',
                                                  ),
                        },
                other => {
                        Genbank => Bio::SeqIO->new(
                                                  -file   => '>other.gb',
                                                  -format => 'Genbank',),
                        Fasta   => Bio::SeqIO->new(
                                                  -file => '>other.fa',
                                                  -format => 'Fasta',),
                        }
                );
while (my $seqin = $inseq->next_seq) {
    if ($seqin->species->binomial =~ m/Homo sapiens/) {
        $outfiles{'human'}->{'Genbank'}->write_seq($seqin);
        $outfiles{'human'}->{'Fasta'}->write_seq($seqin);
    } else {
        $outfiles{'other'}->{'Genbank'}->write_seq($seqin);
        $outfiles{'other'}->{'Fasta'}->write_seq($seqin);
    }
}
```

BioPerl

```perl
1  use Bio::DB::GenBank;
2  use Bio::DB::Query::GenBank;
3
4  $query = "Arabidopsis[ORGN] AND topoisomerase[TITL] and
     0:3000[SLEN]";
5  $query_obj = Bio::DB::Query::GenBank->new(-db => '
     nucleotide',
6                                              -query =>
     $query );
7
8  $gb_obj = Bio::DB::GenBank->new;
9
10 $stream_obj = $gb_obj->get_Stream_by_query($query_obj);
11
12 while ($seq_obj = $stream_obj->next_seq) {
13     # do something with the sequence object
14     print $seq_obj->display_id, "\t", $seq_obj->length,
     "\n";
15 }
```