10) Advanced topics

Matt Webster
IMBIM, BMC
matthew.webster@imbim.uu.se

some advanced topics:

1) file tests
2) directories
3) system commands
4) array slices
5) modules

file tests

file tests begin with `-`

`-e` tests if a file exists

```
if (-e "output.txt") {
   die "output file exists!\n";
} else {
   open OUT,">output.txt";
}
```

file tests

symbol	meaning
`-r`	File or directory is readable by this (effective) user or group
`-w`	File or directory is writable by this (effective) user or group
`-x`	File or directory is executable by this (effective) user or group
`-o`	File or directory is owned by this (effective) user
`-R`	File or directory is readable by this real user or group
`-W`	File or directory is writable by this real user or group
`-X`	File or directory is executable by this real user or group
`-O`	File or directory is owned by this real user
`-e`	File or directory name exists
`-z`	File exists and has zero size (always false for directories)
`-s`	File or directory exists and has nonzero size (the value is the size in bytes)
`-f`	Entry is a plain file
`-d`	Entry is a directory
`-l`	Entry is a symbolic link

directories

`chdir` changes working directory

```
chdir '/etc' or die "unable to change directory\n";
```

`glob` expands a list of files

```
@txt_files = glob "*.txt";

foreach $file (@txt_files) {
   open IN,$file;
   while (<IN>) {
      #read each file
   }
   # go to next file
}
```

other file operations

`unlink` *`file`*
to delete a file

```
unlink qw/file1.txt file2.txt file3.txt/
```
to delete a list of files

```
unlink glob '*.txt'
```
to delete a list of files defined by the glob

`unlink` works similar to `rm` in unix

other file operations

`rename file1, file2`
to rename a file

`rename 'file1.txt', 'example/file1.txt'`
to move a file

`rename` works similar to `mv` in unix

`mkdir name, permissions`
to make a new directory

running a Unix command

`system` runs a command

```
system 'plink --ped plink.ped --map plink.map >outfile.txt'

open IN,"outfile.txt";
while (<IN>) {
    ...
```

runs the program plink and opens the output file

array slices

```
@students = qw/Doreen Oskar Elin Sangeet Malin/;

@reordered_students = @students [3, 2, 1, 4, 0];

@selected_students = @students [2, 4, 3];
```

modules

Comprehensive Perl Archive Network
www.cpan.org

thousands of segments of re-usable perl code

some are available with perl
some need to be installed

example LWP::Simple

LWP::Simple can be used to retrieve webpages

```
#!/usr/bin/perl
use LWP::Simple;
$content = get("http://www.ncbi.nlm.nih.gov/pubmed/?
term=sickle cell disease");
while ($content =~ /sickle/g) {
    $i++;
}
print "found blood $i times\n";
```

does a pubmed search for sickle cell disease and counts how many time "blood" is mentioned in the titles

BioPerl

`BioPerl` is a set of modules useful for biology

`Bio::SeqIO`
Process a sequence file to get the sequence or to find the exon locations

`Bio::SearchIO`
Count the number of hits for each query from a BLAST report

`Bio::AlignIO`
Process a Clustalw formatted alignment

`Bio::TreeIO`
Read a phylogenetic tree file

BioPerl

file `getaccs.pl` – reads sequences into a `Bio::SeqIO` object and prints accessions

```
#!/usr/bin/perl –w

use Bio::SeqIO;
my $usage  = "getaccs.pl file format\n";
my $file   = shift or die $usage;
my $format = shift or die $usage;

my $inseq = Bio::SeqIO->new(
                              -file   => "<$file",
                              -format => $format,
                              );
while (my $seq = $inseq->next_seq) {
    print $seq->accession_number,"\n";
}
```

BioPerl

how to parse all of the hits in output of a blast search:

```
use strict;
use Bio::SearchIO;
my $in = new Bio::SearchIO(-format => 'blast',
                           -file   => 'report.bls');
while( my $result = $in->next_result ) {
  ## $result is a Bio::Search::Result::ResultI compliant object
  while( my $hit = $result->next_hit ) {
    ## $hit is a Bio::Search::Hit::HitI compliant object
    while( my $hsp = $hit->next_hsp ) {
      ## $hsp is a Bio::Search::HSP::HSPI compliant object
      if( $hsp->length('total') > 50 ) {
        if ( $hsp->percent_identity >= 75 ) {
          print "Query=",   $result->query_name,
            " Hit=",        $hit->name,
            " Length=",     $hsp->length('total'),
            " Percent_id=", $hsp->percent_identity, "\n";
        }
      }
    }
  }
}
```

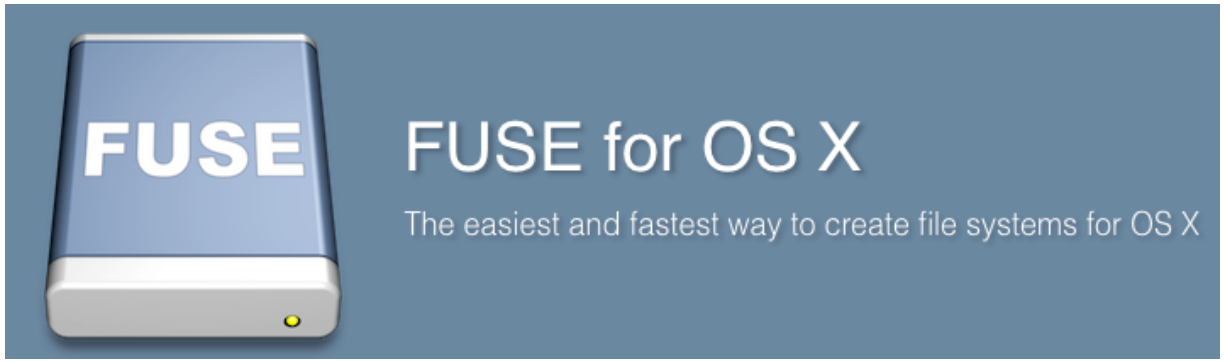
More examples of perl modules are found in the course book (chapter 11) and at www.cpan.org

multidimensional arrays and hashes

```
@array_of_arrays;

$matrix[46][23]=45;



%hash_of_hashes;

$genotype{$individual}{$snp}="AG";
```

Mounting a filesystem with ssh

- http://osxfuse.github.com/



- http://macfusionapp.org/

topic	title	chapters in *Learning Perl (6th ed.)*	key operators / functions	more syntax
1	introduction to unix command line and perl	1	`print`	

2	scalar data	2	`if`
			`chomp`
			`while`
			`<STDIN>`
				`$scalar`
				`+ - * /`
				`\n \t`
				`x`
				`< <= == >= > !=`
				`lt, le, eq, ge, gt, ne`

3	lists / arrays	3	`push, pop`
			`shift, unshift`
			`splice`
			`foreach`
			`reverse`
			`sort`
				`@array`
				`$array[0]`
				`qw//`
				`$_`

4	subroutines	4	`sub`
			`return`
			`my`
			`use strict`	

5	input/output	5	`printf`
			`open, close`
			`die`
				`<>`
				`@ARGV`

6	hashes	6	`keys, values`
			`exists`
			`delete`
				`%hash`
				`$hash{$key}`
				`=>`

7	regular expressions	7,8,9	`m//` `s///` `split` `join`	`. + *` `[] {}` `\w \d` `^ $` `=~` `$1, $2` `$&, $`, $'` `/g ?`

8	control structures	10	`unless`
			`else`
			`elsif`
			`for`
			`last`
			`next`
				`++`
				`--`
				`&& ||`

9	strings + sorting	14	`index`
			`substr`
			`sprintf`	

10	advanced topics: modules, file tests, directories	11,12,13 (+15,16,17)	`use`
			`-e`
			`chdir`
			`glob`
			`unlink`	

the next steps:

- Intermediate Perl
  - introduces modules, references, objects
- Programming Perl
  - a reference to everything
- Perl Cookbook
  - some useful tricks to solve common tasks
- Many Perl for bioinformatics books!