About Files

Input, Output and Other Things

I/O: Input and Output

-
- The standard streams: STDIN, STDOUT and STDERR

STDIN, STDOUT and STDERR

```
my $data = <STDIN>;

my $data = <>;


print STDOUT $data;

print $data;


print STDERR "Something terrible has happened ... aborting.\n";

warn "Something terrible has happened ... aborting.\n";
```

Reading Files

```
This is the first disk-file, line 1.
This is the first disk-file, line 2.
This is the first disk-file, line 3.
This is the first disk-file, line 4.
This is the first disk-file, line 5.

This is the second disk-file, line 1.
This is the second disk-file, line 2.
This is the second disk-file, line 3.
```

Merging Files

```
This is the first disk-file, line 1.
This is the second disk-file, line 1.
This is the first disk-file, line 2.
This is the second disk-file, line 2.
This is the first disk-file, line 3.
This is the second disk-file, line 3.
This is the first disk-file, line 4.
This is the first disk-file, line 5.
```

Determining the disk-file names

```
#! /usr/bin/perl -w

# determine_args - print out the names of the disk-files named
# on the command-line.

if ( $#ARGV != 1 )
{
    warn "Please supply the names of two disk-files on the
              command-line.\n";
    exit;
}

my ( $first_filename, $second_filename ) = @ARGV;

print "first disk-file name is: $first_filename\n";
print "second disk-file name is: $second_filename\n";
```

The Default Argument Array

```
my $first_filename = $ARGV[0];
my $second_filename = $ARGV[1];

my $first_filename = shift;
my $second_filename = shift;
```

```
$ perl determine_args first_file.txt second_file.txt

first disk-file name is: first_file.txt
second disk-file name is: second_file.txt
```

Opening named disk-files

```
#! /usr/bin/perl -w

# check_args - check that the disk-files named
# on the command-line exist.

if ( $#ARGV != 1 )
{
    die "Please supply the names of two disk-files on
             the command-line.\n";
}

my ( $first_filename, $second_filename ) = @ARGV;

unless ( -e $first_filename && -f $first_filename && -r $first_filename )
{
    die "$first_filename cannot be accessed. Does it exist?\n";
}
unless ( -e $second_filename && -f $second_filename && -r $second_filename )
{
    die "$second_filename cannot be accessed. Does it exist?\n";
}
```

The check_args program, cont.

```
open FIRSTFILE, "$first_filename";
open SECONDFILE, "$second_filename";

open FIRSTFILE, "<$first_filename";
open SECONDFILE, "<$second_filename";

open FIRSTFILE, "$first_filename"
    or die "Could not open $first_filename. Aborting.\n";
open SECONDFILE, "$second_filename"
    or die "Could not open $second_filename. Aborting.\n";

close FIRSTFILE;
close SECONDFILE;
```

Maxim 6.1

Open a disk-file for as long as needed, but no longer

Maxim 6.2

If you open a disk-file, be sure to close it later.

Reading a line from each of the disk-files

```
my ( $linefromfirst, $linefromsecond );

$linefromfirst = <FIRSTFILE>;
$linefromsecond = <SECONDFILE>;
```

Putting it all together

```
#! /usr/bin/perl -w

# merge2files - merge the two disk-files named on the command-line.

if ( $#ARGV != 1 )
{
    die "Please supply the names of two disk-files on
             the command-line.\n";
}

my ( $first_filename, $second_filename ) = @ARGV;

unless ( -e $first_filename && -f $first_filename && -r $first_filename )
{
    die "$first_filename cannot be accessed. Does it exist?\n";
}
```

The merge2files program, cont.

```
unless ( -e $second_filename && -f $second_filename && -r $second_filename )
{
    die "$second_filename cannot be accessed. Does it exist?\n";
}

open FIRSTFILE, "$first_filename"
    or die "Could not open $first_filename. Aborting.\n";
open SECONDFILE, "$second_filename"
    or die "Could not open $second_filename. Aborting.\n";

my ( $linefromfirst, $linefromsecond );

while ( $linefromfirst = <FIRSTFILE> )
{
    $linefromsecond = <SECONDFILE>;

    print $linefromfirst;
    print $linefromsecond;
}

close FIRSTFILE;
close SECONDFILE;
```

Running merge2files ...

```
$ chmod +x merge2files

$ ./merge2files first_file.txt second_file.txt


This is the first disk-file, line 1.
This is the second disk-file, line 1.
This is the first disk-file, line 2.
This is the second disk-file, line 2.
This is the first disk-file, line 3.
This is the second disk-file, line 3.
This is the first disk-file, line 4.
Use of uninitialized value in print at merge2files line 35, <SECONDFILE> line 3.
Use of uninitialized value in print at merge2files line 35, <SECONDFILE> line 3.
This is the first disk-file, line 5.
Use of uninitialized value in print at merge2files line 35, <SECONDFILE> line 3.
Use of uninitialized value in print at merge2files line 35, <SECONDFILE> line 3.
```

Running merge2files again ...

```
$ ./merge2files second_file.txt first_file.txt


This is the second disk-file, line 1.
This is the first disk-file, line 1.
This is the second disk-file, line 2.
This is the first disk-file, line 2.
This is the second disk-file, line 3.
This is the first disk-file, line 3.
```

Creating merge2files_v2

```
if ( !eof( SECONDFILE ) )
{
    $linefromsecond = <SECONDFILE>;
    print $linefromsecond;
}
```

Running merge2files_v2 ...

```
$ ./merge2files_v2 first_file.txt second_file.txt


This is the first disk-file, line 1.
This is the second disk-file, line 1.
This is the first disk-file, line 2.
This is the second disk-file, line 2.
This is the first disk-file, line 3.
This is the second disk-file, line 3.
This is the first disk-file, line 4.
This is the first disk-file, line 5.
```

Running merge2files_v2 again ...

```
This is the second disk-file, line 1.
This is the first disk-file, line 1.
This is the second disk-file, line 2.
This is the first disk-file, line 2.
This is the second disk-file, line 3.
This is the first disk-file, line 3.
```

Creating merge2files_v3

```
#! /usr/bin/perl -w

# merge2files_v3 - third version of merge2files: merge the disk-files
# named on the command-line (with some help from eof and defined).
# Make sure all lines are read from both disk-files.

if ( $#ARGV != 1 )
{
    die "Please supply the names of two disk-files on
             the command-line.\n";
}

my ( $first_filename, $second_filename ) = @ARGV;

unless ( -e $first_filename && -f $first_filename && -r $first_filename )
{
    die "$first_filename cannot be accessed. Does it exist?\n";
}
```

The merge2files_v3 program, cont.

```
unless ( -e $second_filename && -f $second_filename && -r $second_filename )
{
    die "$second_filename cannot be accessed. Does it exist?\n";
}

open FIRSTFILE, "$first_filename"
    or die "Could not open $first_filename. Aborting.\n";

open SECONDFILE, "$second_filename"
    or die "Could not open $second_filename. Aborting.\n";

my ( $linefromfirst, $linefromsecond );
```

The merge2files_v3 program, cont.

```
while ( $linefromfirst = <FIRSTFILE> )
{
    print $linefromfirst;
    if ( !eof( SECONDFILE ) )
    {
        $linefromsecond = <SECONDFILE>;
        print $linefromsecond;
    }
}

while ( !eof( SECONDFILE ) )
{
    $linefromsecond = <SECONDFILE>;
    print $linefromsecond;
}

close FIRSTFILE;
close SECONDFILE;
```

Slurping

```
@entire_file = <>;
```

Slurping example

```
#! /usr/bin/perl -w

# slurper - a program which demonstrates disk-file "slurping".

use lib "$ENV{'HOME'}/bbp/";
use UsefulUtils qw( drawline );

open FIRSTSLURPFILE, "first_file.txt"
    or die "Could not open first slurp disk-file. Aborting.\n";
open SECONDSLURPFILE, "second_file.txt"
    or die "Could not open second slurp disk-file. Aborting.\n";

my @linesfromfirst = <FIRSTSLURPFILE>;
my @linesfromsecond = <SECONDSLURPFILE>;

print drawline( Count => 40 ), "\n";
print @linesfromfirst;
print drawline( Count => 40 ), "\n";
print @linesfromsecond;
print drawline( Count => 40 ), "\n";

close FIRSTSLURPFILE;
close SECONDSLURPFILE;
```

Results from slurper ...

```
----------------------------------------
This is the first disk-file, line 1.
This is the first disk-file, line 2.
This is the first disk-file, line 3.
This is the first disk-file, line 4.
This is the first disk-file, line 5.
----------------------------------------
This is the second disk-file, line 1.
This is the second disk-file, line 2.
This is the second disk-file, line 3.
----------------------------------------
```

Writing Files

```
my $file_to_open = "errors.log";

...

open( LOGFILE, ">$file_to_open" )
    or die "Could not write to/create errors log disk-file.\n";

open( LOGFILE, ">>$file_to_open" )
    or die "Could not append to/create errors log disk-file.\n";

print LOGFILE "Error: something terrible has happened.\n";
```

Redirecting output

```
$ ./merge2files first_file.txt second_file.txt > merge.out

$ ./merge2files first_file.txt second_file.txt >> merge.out

$ ./merge2files first_file.txt second_file.txt > merge.out 2> merge.err

$ ./merge2files first_file.txt second_file.txt >> merge.out 2>> merge.err
```

Variable interpolation

```
my $sequence = "TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA";

print "The sequence is: $sequence\n";
print 'The sequence is: $sequence\n';

The sequence is: TTATTATGTT GCTCAGTTCT GACCTCTTAA CTATGCGGTA
The sequence is: $sequence\n
```

Chopping and chomping

```
my $dna = "ATGTGCGGTATTGCTGACCTCTTA\n";

my $last = chop $dna;

# $dna is now "ATGTGCGGTATTGCTGACCTCTTA";

my $next = chop $dna;

# $dna is now "ATGTGCGGTATTGCTGACCTCTT";

my $dna = "ATGTGCGGTATTGCTGACCTCTTA\n";

my $last = chomp $dna;

# $dna is now "ATGTGCGGTATTGCTGACCTCTTA";

my $next = chomp $dna;

# $dna is still "ATGTGCGGTATTGCTGACCTCTTA";
```

Where To From Here