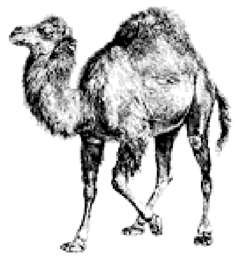


www.perl.org

Learning Perl Through Examples Part 2

L1110@BUMC

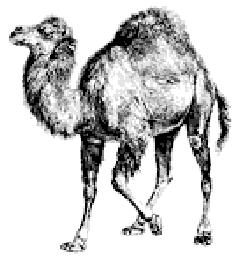
9/22/2017

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Tutorial Resource



www.perl.org

Before we start, please take a note - all the codes and supporting documents are accessible through:

- http://rcs.bu.edu/examples/perl/tutorials/

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

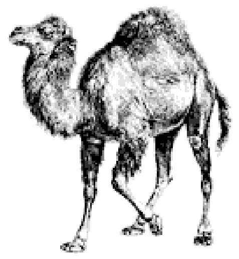
Sign In Sheet

We prepared sign-in sheet for each one to sign
We do this for internal management and quality control
So please SIGN IN if you haven't done so

www.perl.org

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Evaluation



www.perl.org

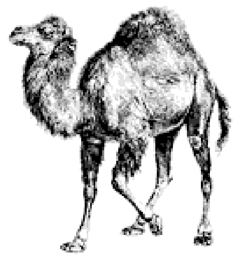
One last piece of information before we start:

- DON'T FORGET TO GO TO:
  - http://rcs.bu.edu/survey/tutorial_evaluation.html

Leave your feedback for this tutorial (both good and bad as long as it is honest are welcome. Thank you)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

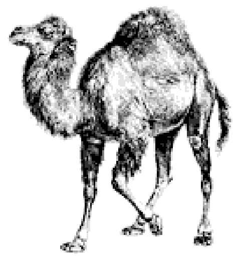
Today's Topic



www.perl.org

- Basics on creating your code
- About Today's Example
- Learn Through Example 1 – fanconi_example_io.pl
- Learn Through Example 2 – fanconi_example_str_process.pl
- Learn Through Example 3 – fanconi_example_gene_anno.pl
- Extra Examples (if time permit)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Basics on creating your code

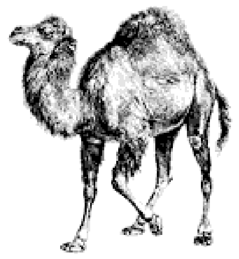
How to combine specs, tools, modules and knowledge.

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

What is needed

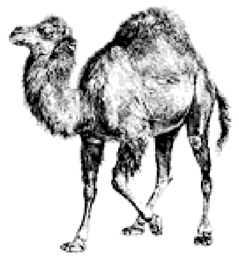


www.perl.org

Consider your code/software a '**product**', what will it take to **produce** **it**?

- **User Requirements (domain knowledge, that's very important)**
- Development Environment (Emacs/gedit/Eclipse/etc)
- Third Party Modules/Toolboxes (CPAN)
- Some workman's craft (You/Programmer)
- Help systems (Help documentation/reference books/stackflow/etc)
- Language specification (Perldoc/reference guide)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

User Requirements

Specify what software is expected to do

Can be formal or casual, but better keep records of.

- Formal – User Requirement Documentation (URD)
- Casual – email conversations, scratch paper memos, etc.

Types of Requirements

- M – Mandatory
- D – Desirable
- O – Optional
- E – Enhanceable

Serve as contract – keep project on track

Pitfall – often ignored

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Development Environment

www.perl.org

It is like your workshop where you go to work and make your product

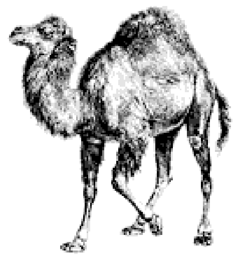
How to pick your development tools (mainly editor or IDE)

- Convenient
- Sufficient enough
- Extensible/adaptive
- Personal preference

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Development Environment



www.perl.org

Some commonly used tools:

1) Editor Only:

emacs
vim
gedit

2) IDE (Integrated Development Environment)

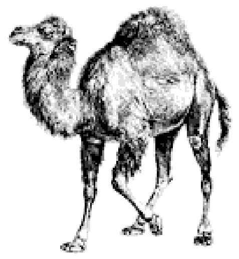
Eclipse
Padre

You may go to http://perlide.org/poll200910/ for the poll result conducted by a Perl guru for Perl Editors

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

CPAN – Where Third Party Modules Resides



www.perl.org

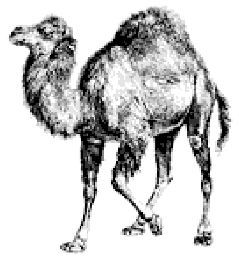
- Perl is a community built software system, enriched by third party contributors. All efforts go to build CPAN open source archive network for Perl.
- Perl's richness and power comes from CPAN and the 3rd party modules and toolkits covering various domains, for example, Finance, BioPerl, Catalyst, DBI, and many others.
- CPAN official site: www.cpan.org
- Two search engine interfaces:

  search.cpan.org (old, traditional)

  metacpan.org (new, modern, provides rich APIs for automation)

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Help systems



www.perl.org

One significant criteria for a good programming language is its documentation and help system – In this sense, Perl is quite good

Its own:

- Language Specification itself well written
- Organized well (divided by categories)
- Presented well (perldoc utility/man, Internet available)

Online Resource:

- Rich online help, tutorials, and e-books (many for free)

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

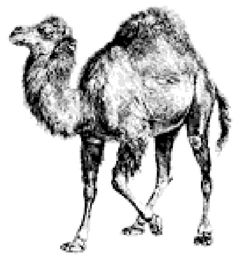
Language specification

Also called ‘Reference Guide’

Perldoc Official Site: http://perldoc.perl.org

Divided to eight subcategories:

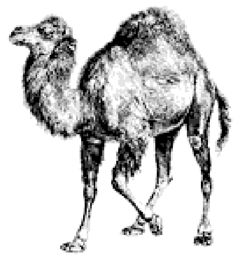
1. Language
2. Functions
3. Operators
4. Special variables
5. Pragmas
6. Utilities
7. Internals
8. Platform Specific



www.perl.org

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Workman’s Crafts



www.perl.org

Hard Part

Takes time to build, but takes no time to start (practice is the best way to learn)

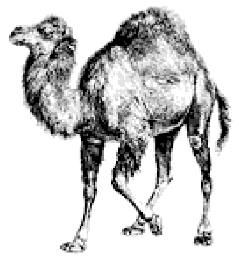
Skills Needed Include:

- Familiarity to language elements
- Software Engineering Methodology
- Algorithm Design
- Code Implementation
- Debugging
- Domain knowledge

Metaphor : How do we acquire skills on natural language

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



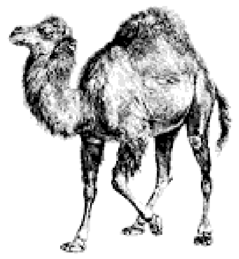
www.perl.org

Before We Start …

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Connecting to SCC



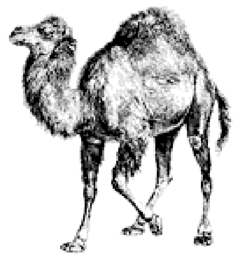
www.perl.org

- Option 1: You are able to keep everything you generate
  Use your Shared Computing Cluster account if you have one.

- Option 2: all that you do in the tutorial may be wiped out after tutorial ends unless you move the contents to somewhere belong to you.
  Tutorial accounts if you need one (will be provided in class)
  Username : TBD
  Password : TBD

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Download source code

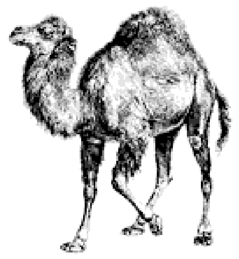


www.perl.org

Follow these steps to download the code:

```
ssh user@sccN.bu.edu ('user' is an account on SCC, 'N' can be 1-4)

mkdir perlThruEx

cd perlThruEx

wget http://scv.bu.edu/examples/perl/tutorials/src/perlThruExamples.zip
```

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



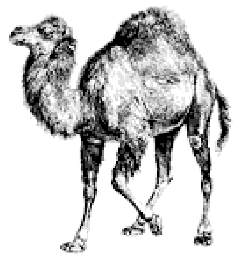
www.perl.org

Today’s Example Overview

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Example Preview – Fanconi Gene Introduction

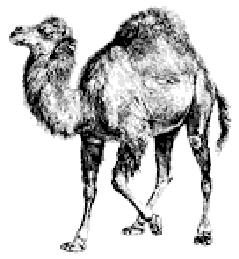


www.perl.org

- Fanconi genes refer to the genes that have been identified as closely related to a genetic disease called 'Fanconi Amaemia'(FA).
- 17 genes are identified so far, and 15 of them named as 'FANC[A-S]', 2 others have totally non-revealing names, 'RAD51C' and 'XPF'.
- For this example, we will only take the 15 genes that start with 'FANC' as the input gene list.

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Example Preview – Content Coverage



www.perl.org

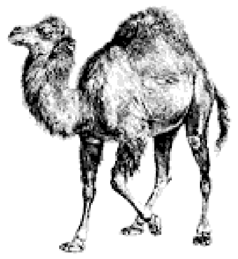
This tutorial will use fanconi genes example to go through three main functional strengths of Perl – File IOs, string match and process, and last, power in using 3rd party modules, in this case, we use BioPerl's gene annotation module, GenBank.

- Example 1: File IOs
- Example 2: String Processing
- Example 3: Gene Annotation

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Example Preview – Code Organization



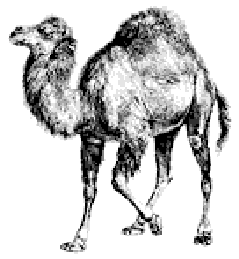
www.perl.org

General Setting:

- Input directory – the place to put all input files
  - ./code/session2/data_in
- Output directory – the place where end result is put
  - ./code/session2/data_out
- Script directory – the place where Perl scripts reside
  - ./code/session2/scripts

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Example 1 Preview



www.perl.org

Script: fanconi_example_io.pl

Purpose: build up the standard File IO concepts.

- contains 6 subroutines, each demonstrates a slightly different way Perl handles input and output

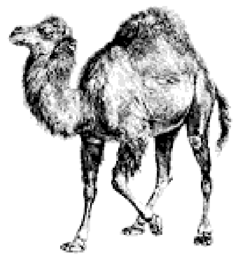
Command:

```
perl fanconi_example_io.pl --example n
```

Note: fanconi_example_io_fancy.pl is a bit fancier version, which adds support of command line arguments for the flexibility

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Example 2 Preview



www.perl.org

Script: fanconi_example_str_process.pl

Purpose: Demonstrate regular expressions in Perl.

Contains 2 subroutines:

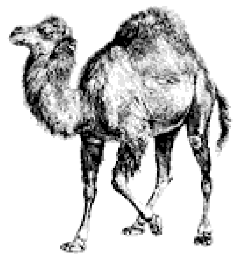
- input file format conversion, from csv to tab format
- gene selection using specified criteria, from all 15 genes, only pick first 5 with FANC[A-D].

Command: `perl fanconi_example_str_process.pl --example` *n*

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Example 3 Preview



www.perl.org

Script: fanconi_example_gene_anno.pl

Purpose: Demonstrate the comprehensive coding and debugging skills.

- Use BioPerl module (Bio::DB::EntrezGene) for gene annotation from entrez gene id.
- Go through code in detail with debugger

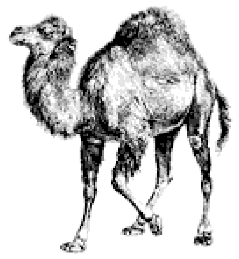
Command:

```
fanconi_example_gene_anno.pl
```

This will consume the most of the tutorial time.

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Learn Through Example - File IO

fanconi_example_io.pl

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

File IO basics - Filehandle

www.perl.org

A structure used to associate a physical file with name

Three standard file handles:

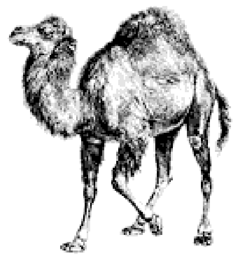
- STDIN – Standard input, usually set to be keyboard
- STDOUT – Standard output, usually set to be screen, using device id ‘1’
- STDERR – Standard error, to display error info (usually set to be same screen), using device id ‘2’

One special file handle - /dev/null, logical file handle to absorb all unwanted output, like black hole, no return once get in. For example:

```
>/dev/null 2>&1
```

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO basics – File IO functions



www.perl.org

- These are the actual functions one may use to manipulate files
- Basic File IO functions:
  - open
  - close
  - opendir
  - closedir
  - read
  - print
  - rename
  - unlink
  - tell
  - seek

Form full list:

http://perldoc.perl.org/index-functions-by-cat.html#Input-and-output-functions

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO basics - Operators

www.perl.org

Operators are actually functions of special type, which are predefined by language to accomplish specific operations upon operand, usually independent of operand(s) it involves.

<> : File handle operator, used to read file using handle; can be single line or multiple lines, depending on context, for example:

```
$single_line = <STDIN>; # read single line from screen
@multi_lines = <DATA>; # read whole data file
```

- : File test operator, used to test various attributes of the file

-e	check file existence
-s	check file size
-d	check if file is actually a directory
-z	check if file size is zero

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO basics – Special variables

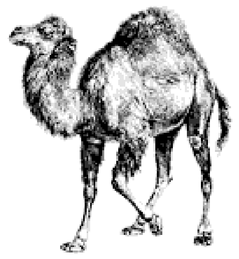
www.perl.org

There are many File IO related special variables in Perl

For example:

`$/`	Input file line separator (delimiter)
`$\`	Output file line separator (delimiter)
`$.`	Line number
`$|`	No buffer flush (if set to true, flush right away)
`$!`	Error information

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Common Scenarios involving file/data processing

1. Conversion between file formats.

   For example, convert Excel spreadsheet to plain text (.csv or .tab) for downstream processing (pre-processing).

2. Data filtering/cleaning/verification

   For example, clean and preview/summary the input data (pre-processing/in-processing)

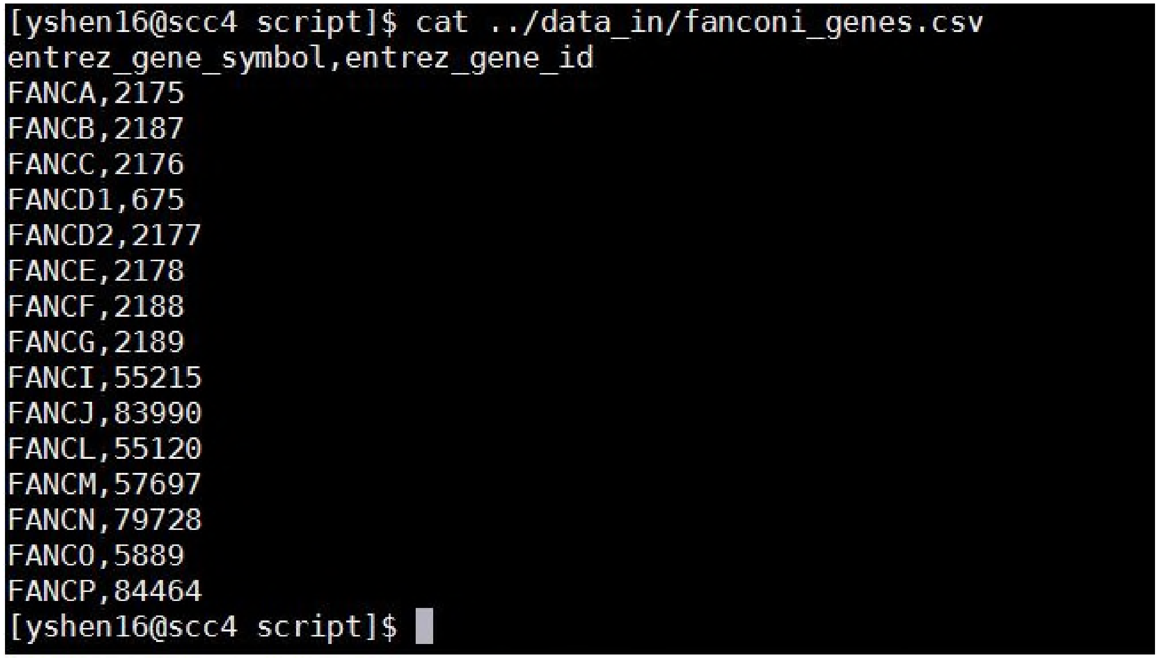
3. Apply business logic to the clean/filtered input data file.

4. Output is not restricted to files. Could be a table in database, or memory block to feed the downstream in an integrated pipeline setting (but beyond this tutorial).

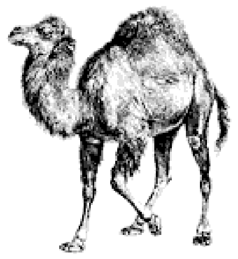
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Input file – fanconi_genes.csv



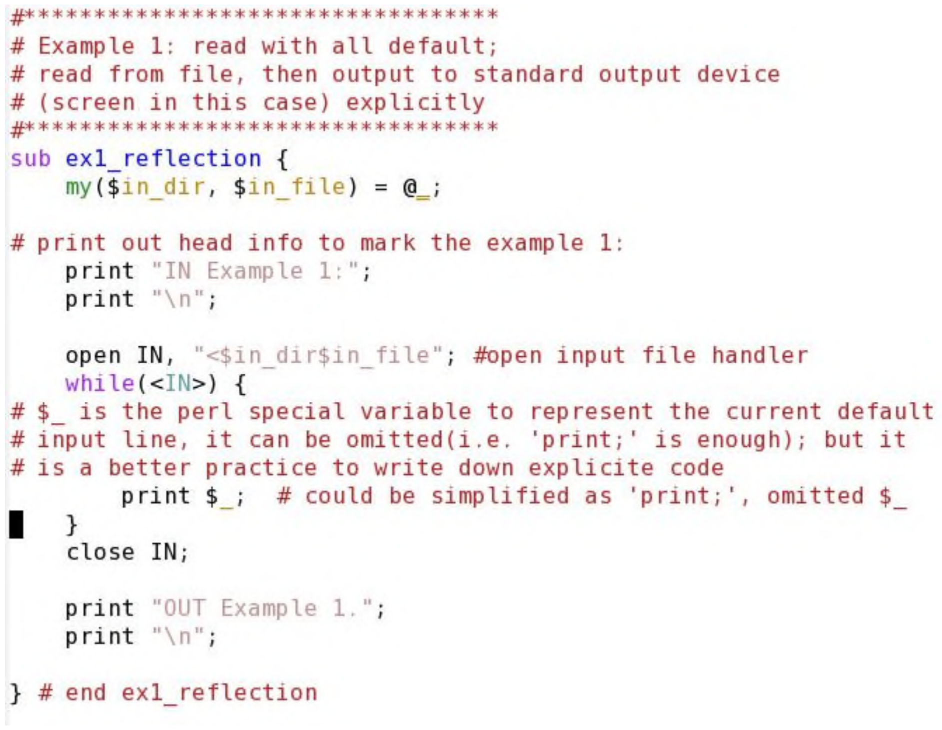
```
[yshen16@scc4 script]$ cat ../data_in/fanconi_genes.csv
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
[yshen16@scc4 script]$ 
```



www.perl.org

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO - Example 1 code (use default)

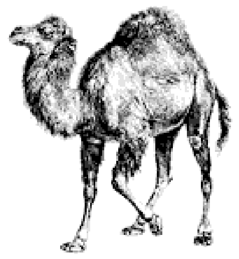


```perl
#**********************************
# Example 1: read with all default;
# read from file, then output to standard output device
# (screen in this case) explicitly
#**********************************
sub ex1_reflection {
    my($in_dir, $in_file) = @_;

# print out head info to mark the example 1:
    print "IN Example 1:";
    print "\n";

    open IN, "<$in_dir$in_file"; #open input file handler
    while(<IN>) {
# $_ is the perl special variable to represent the current default
# input line, it can be omitted(i.e. 'print;' is enough); but it
# is a better practice to write down explicite code
        print $_;  # could be simplified as 'print;', omitted $_
    }
    close IN;

    print "OUT Example 1.";
    print "\n";

} # end ex1_reflection
```

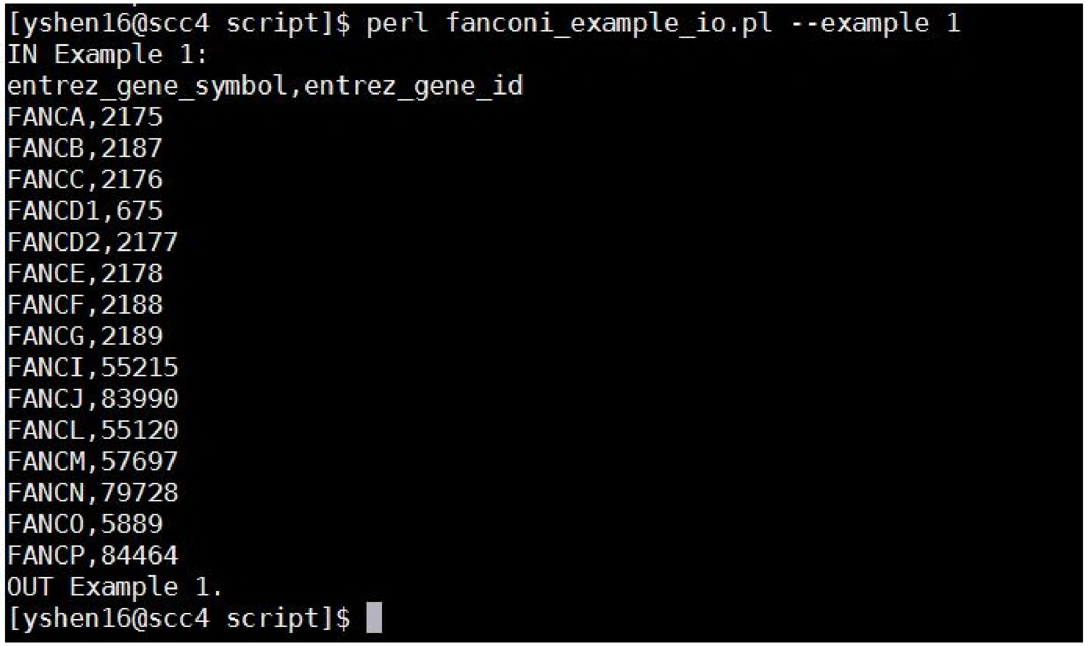
www.perl.org

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 1 output



www.perl.org

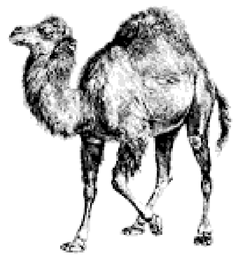


```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 1
IN Example 1:
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
OUT Example 1.
[yshen16@scc4 script]$ 
```

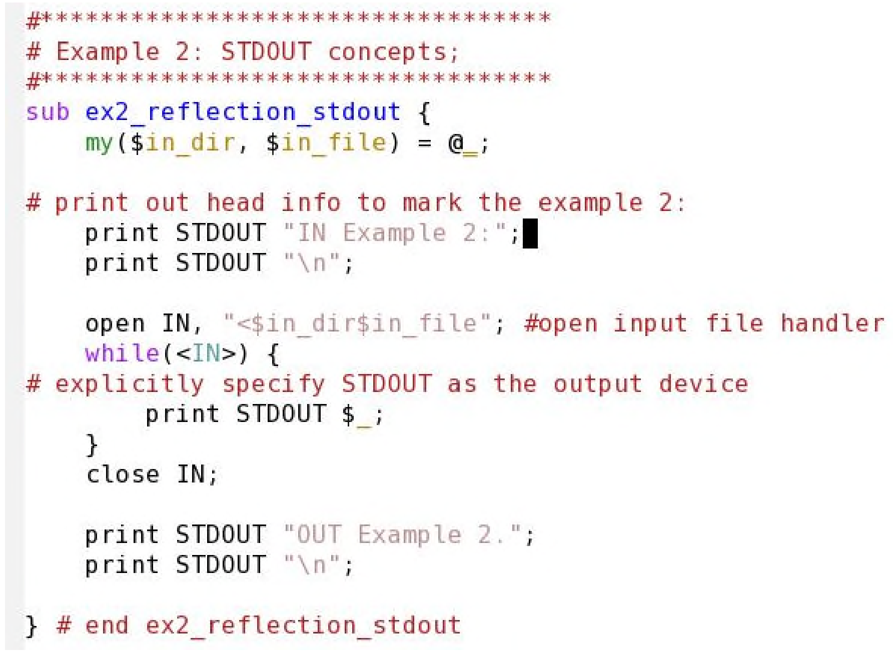
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 2 code (explicit)



www.perl.org



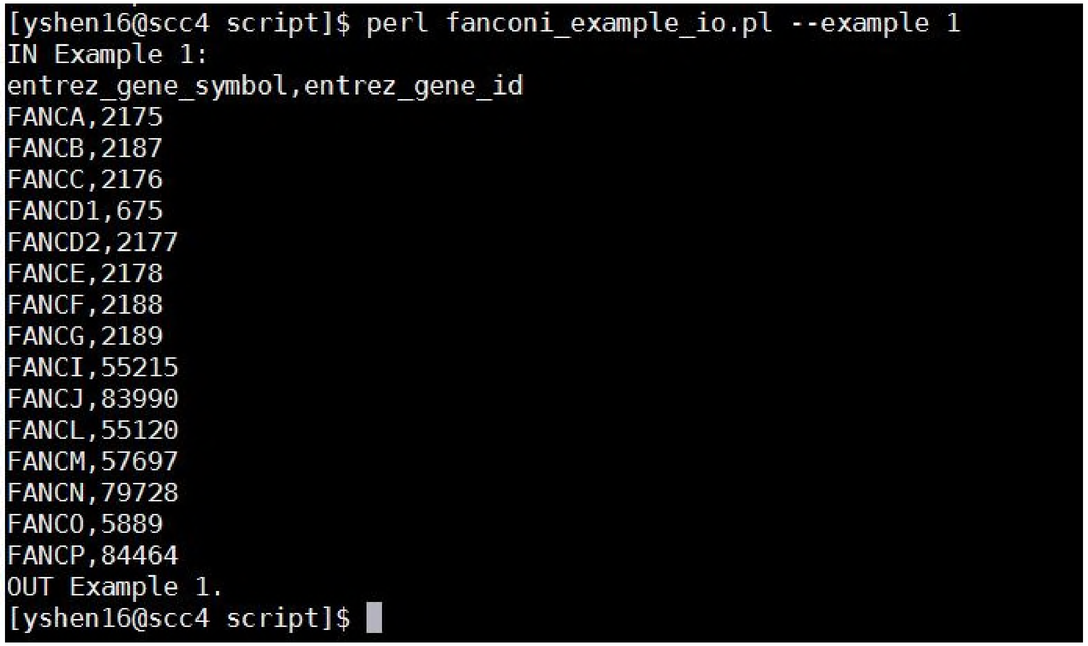
```
#***********************************
# Example 2: STDOUT concepts;
#***********************************
sub ex2_reflection_stdout {
    my($in_dir, $in_file) = @_;

# print out head info to mark the example 2:
    print STDOUT "IN Example 2:";
    print STDOUT "\n";

    open IN, "<$in_dir$in_file"; #open input file handler
    while(<IN>) {
# explicitly specify STDOUT as the output device
        print STDOUT $_;
    }
    close IN;

    print STDOUT "OUT Example 2.";
    print STDOUT "\n";

} # end ex2_reflection_stdout
```

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 2 output

www.perl.org



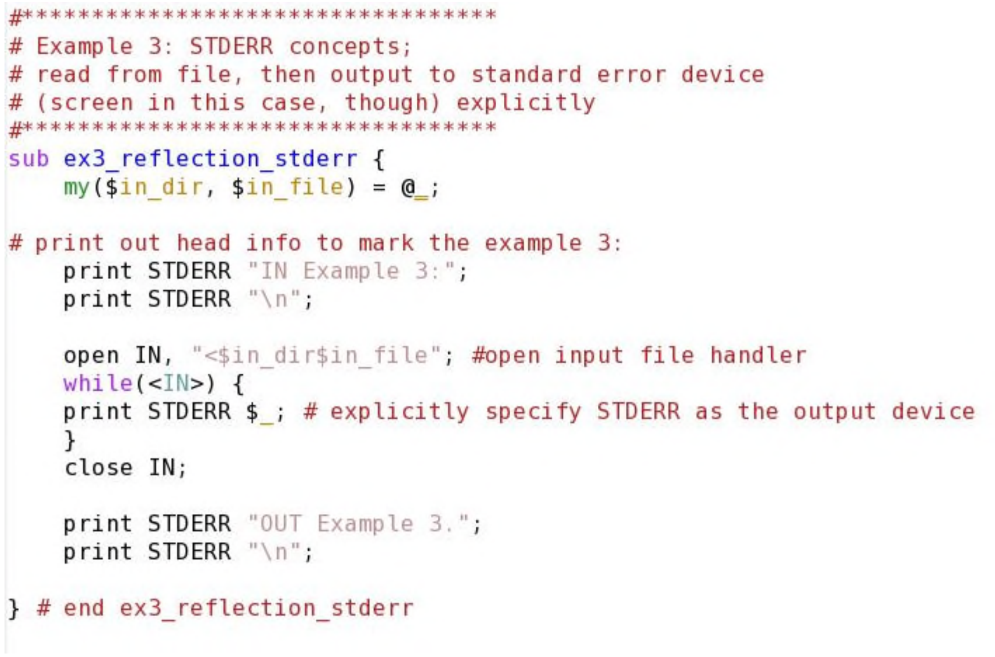
```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 1
IN Example 1:
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
OUT Example 1.
[yshen16@scc4 script]$
```

Observation: compare Example 1 and Example 2, the output are the same.

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 3 code (STDERR)

www.perl.org

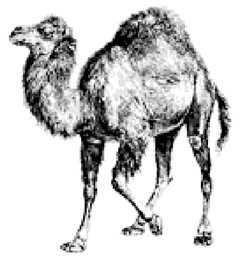


```
#*********************************
# Example 3: STDERR concepts;
# read from file, then output to standard error device
# (screen in this case, though) explicitly
#*********************************
sub ex3_reflection_stderr {
    my($in_dir, $in_file) = @_;

# print out head info to mark the example 3:
    print STDERR "IN Example 3:";
    print STDERR "\n";

    open IN, "<$in_dir$in_file"; #open input file handler
    while(<IN>) {
    print STDERR $_; # explicitly specify STDERR as the output device
    }
    close IN;

    print STDERR "OUT Example 3.";
    print STDERR "\n";

} # end ex3_reflection_stderr
```

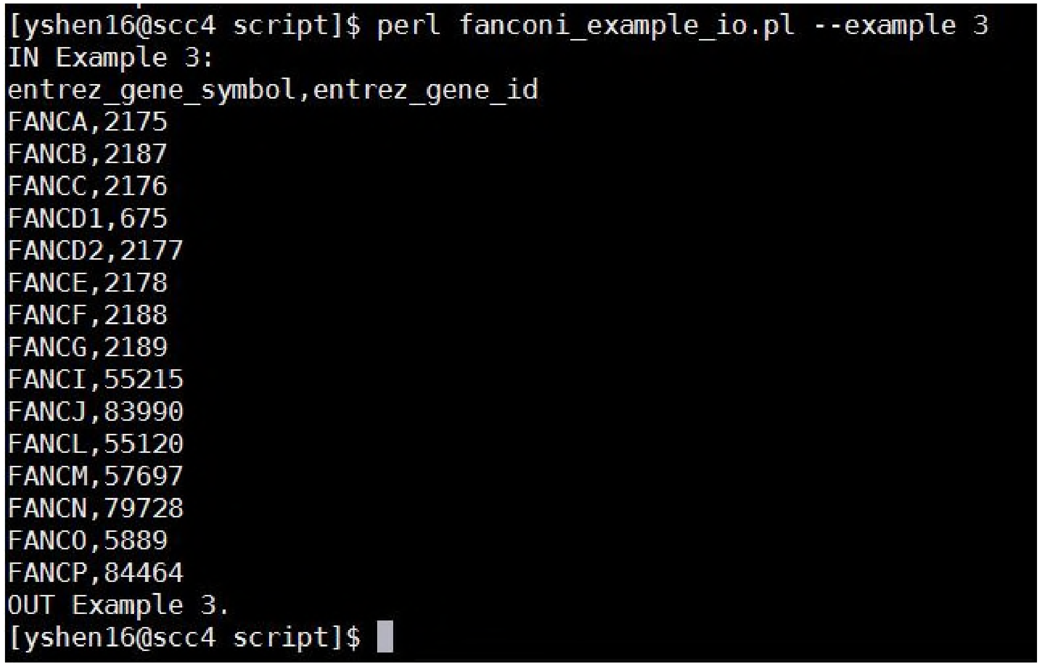
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 3 output



www.perl.org



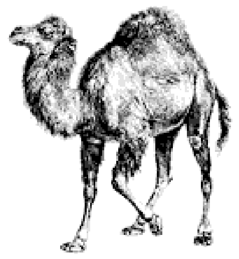
```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 3
IN Example 3:
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
OUT Example 3.
[yshen16@scc4 script]$
```

Observation: Example 3 seems to output same result as Example 1 and Example 2.

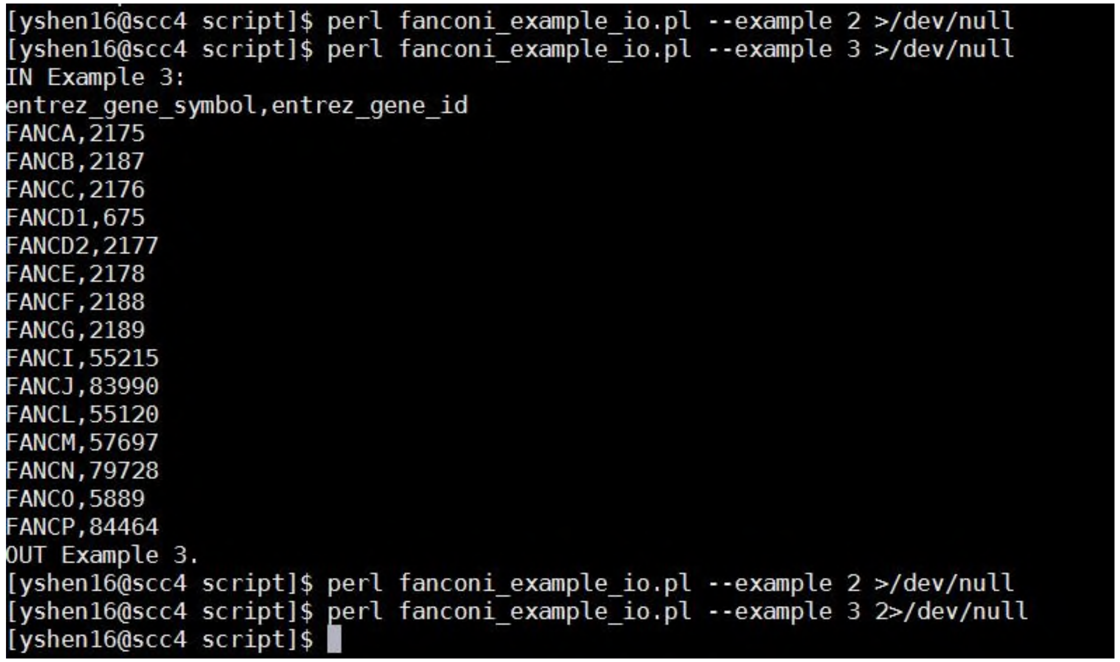
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 2 vs Example 3 output



www.perl.org



```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 2 >/dev/null
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 3 >/dev/null
IN Example 3:
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
OUT Example 3.
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 2 >/dev/null
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 3 2>/dev/null
[yshen16@scc4 script]$ 
```

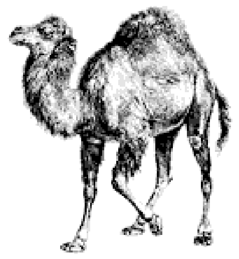
Observation:

Example 3 actually is not as same as example 1 and ;

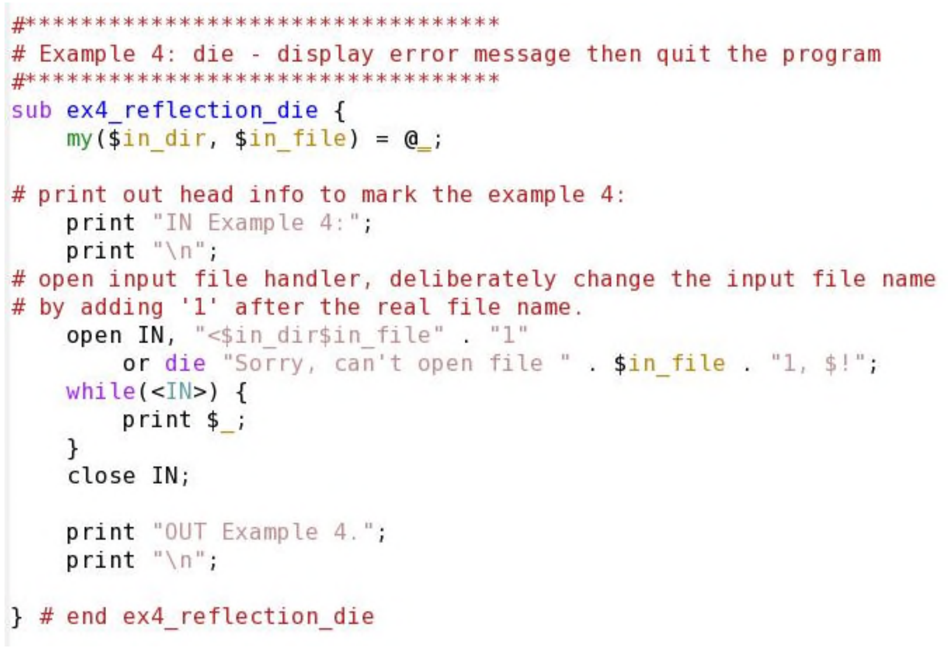
Example 1 and 2 -> STDOUT
Example 3 -> STDERR

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 4 code (die)



www.perl.org

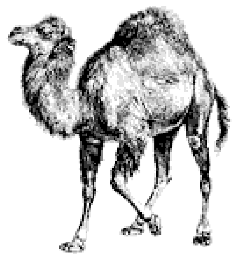


```
#************************************
# Example 4: die - display error message then quit the program
#************************************
sub ex4_reflection_die {
    my($in_dir, $in_file) = @_;

# print out head info to mark the example 4:
    print "IN Example 4:";
    print "\n";
# open input file handler, deliberately change the input file name
# by adding '1' after the real file name.
    open IN, "<$in_dir$in_file" . "1"
        or die "Sorry, can't open file " . $in_file . "1, $!";
    while(<IN>) {
        print $_;
    }
    close IN;

    print "OUT Example 4.";
    print "\n";

} # end ex4_reflection_die
```

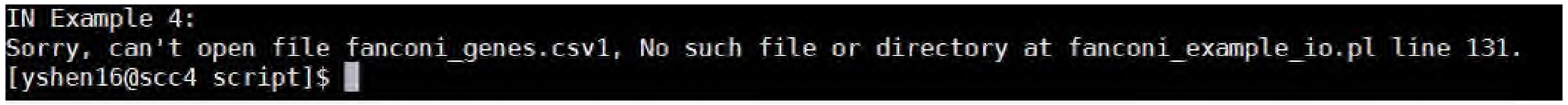
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 4 output



www.perl.org

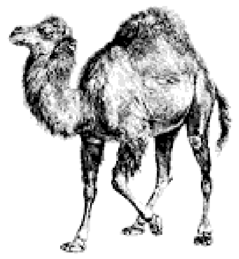


```
IN Example 4:
Sorry, can't open file fanconi_genes.csv1, No such file or directory at fanconi_example_io.pl line 131.
[yshen16@scc4 script]$ █
```

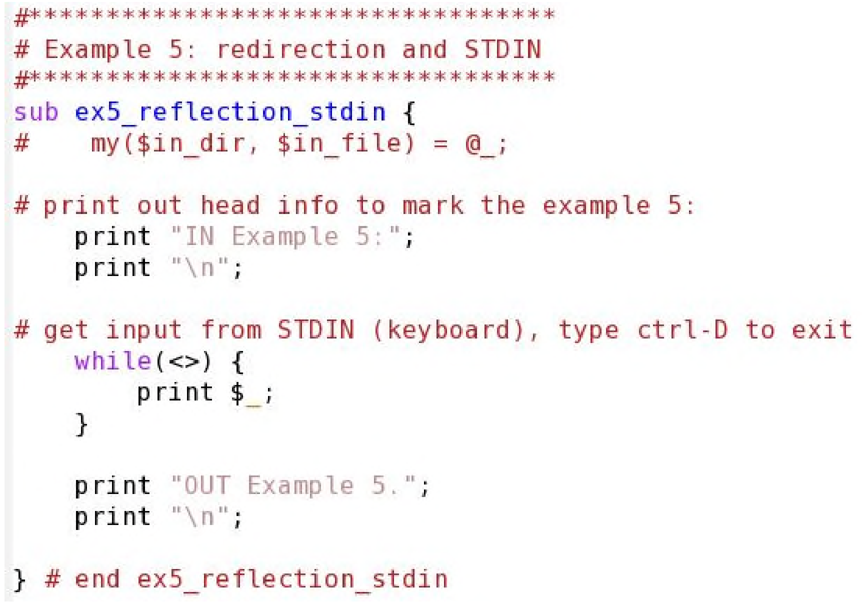
Observation 1: using ‘die’ is a good practice when open file;
Observation 2: using ‘$!’ special variable shows the system error message;

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 5 code (STDIN)



www.perl.org



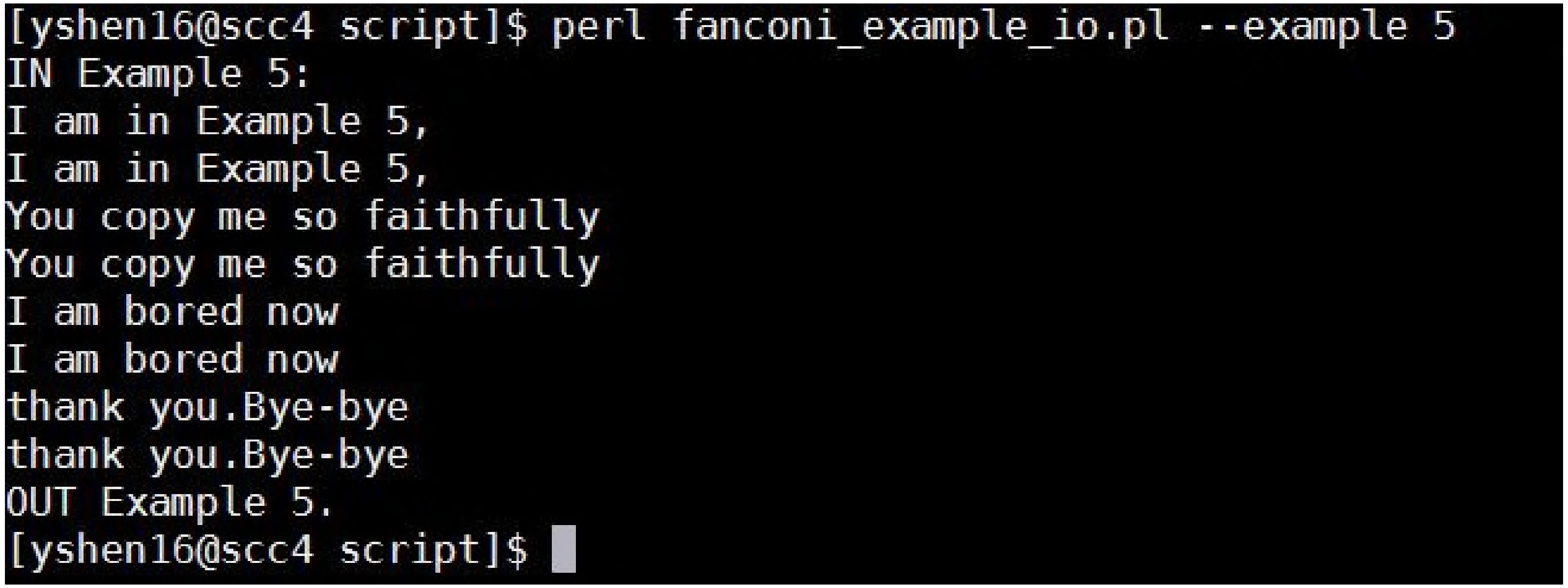
```
#*********************************
# Example 5: redirection and STDIN
#*********************************
sub ex5_reflection_stdin {
#    my($in_dir, $in_file) = @_;

# print out head info to mark the example 5:
    print "IN Example 5:";
    print "\n";

# get input from STDIN (keyboard), type ctrl-D to exit
    while(<>) {
        print $_;
    }

    print "OUT Example 5.";
    print "\n";

} # end ex5_reflection_stdin
```

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 5 output

www.perl.org

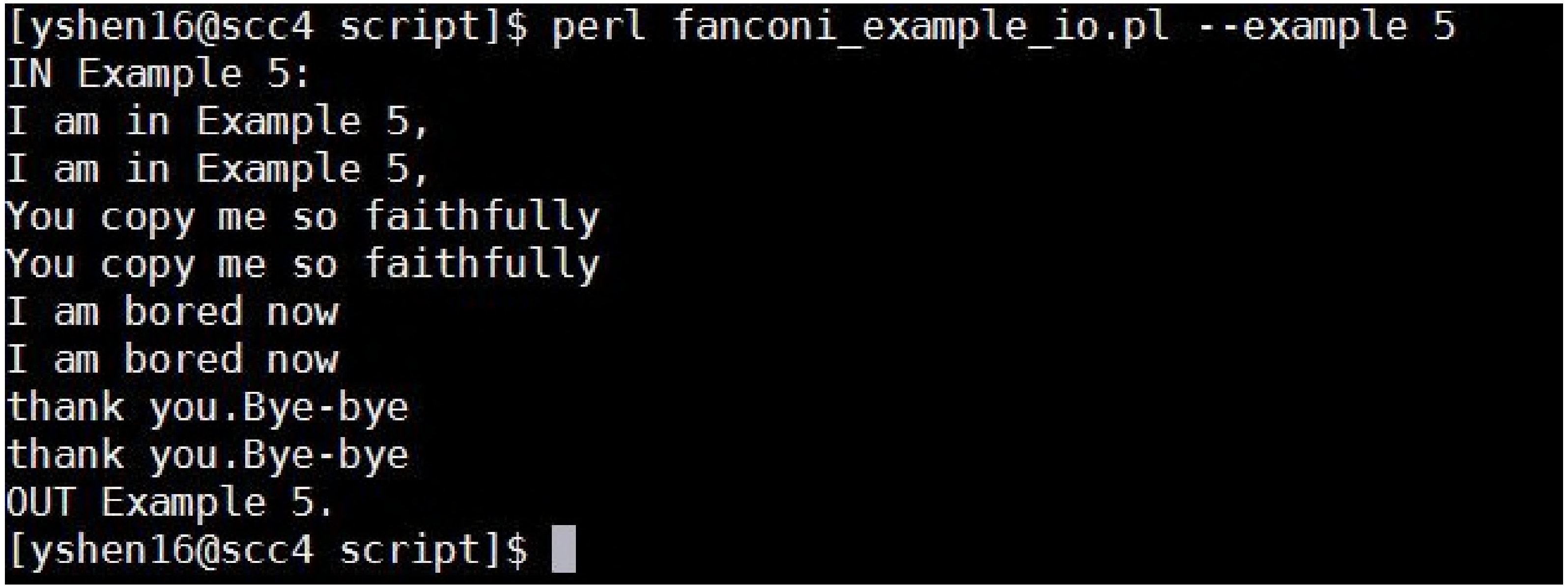


```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 5
IN Example 5:
I am in Example 5,
I am in Example 5,
You copy me so faithfully
You copy me so faithfully
I am bored now
I am bored now
thank you.Bye-bye
thank you.Bye-bye
OUT Example 5.
[yshen16@scc4 script]$
```

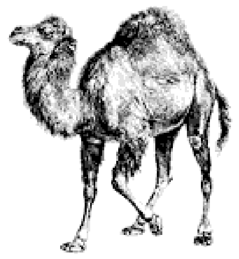
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 5 output



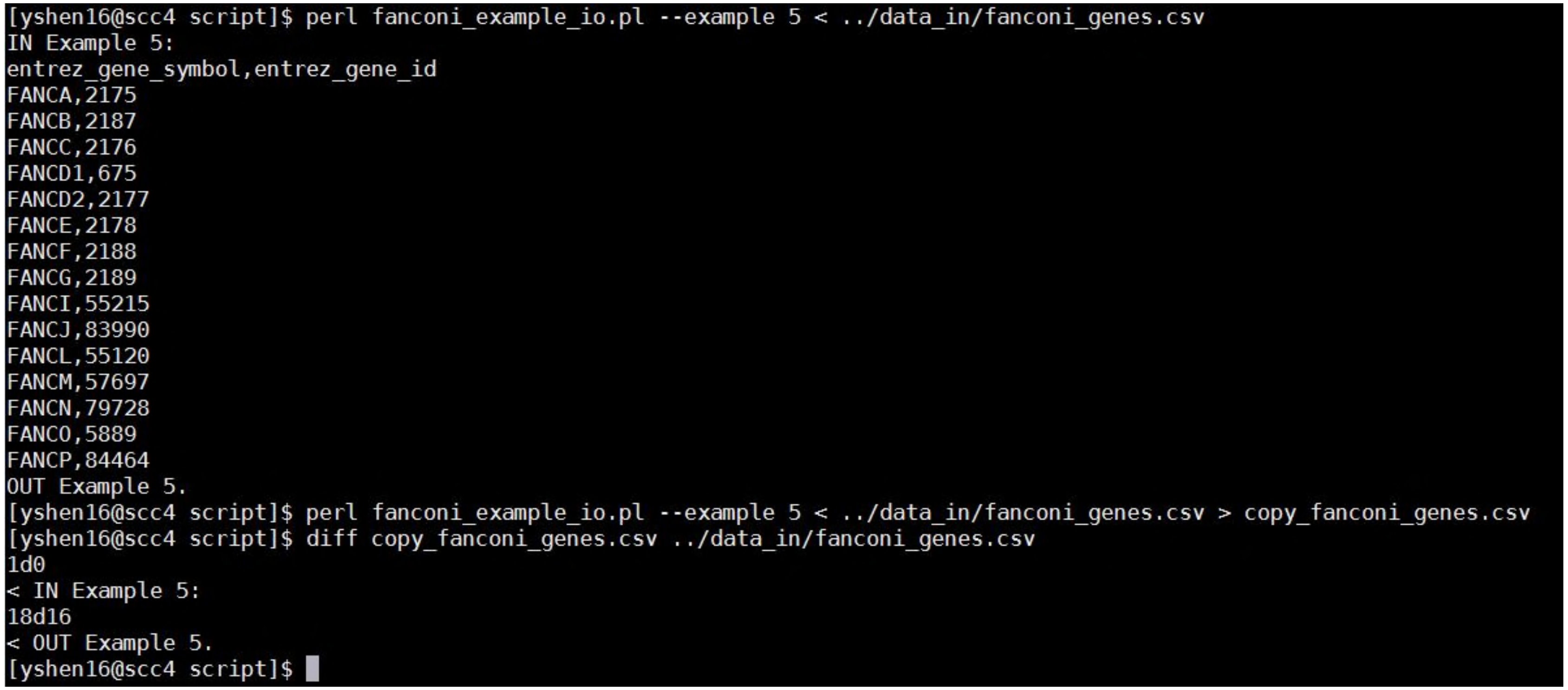
```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 5
IN Example 5:
I am in Example 5,
I am in Example 5,
You copy me so faithfully
You copy me so faithfully
I am bored now
I am bored now
thank you.Bye-bye
thank you.Bye-bye
OUT Example 5.
[yshen16@scc4 script]$ 
```



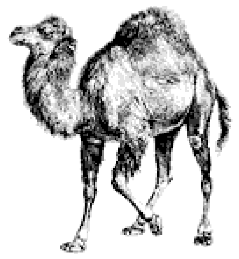
www.perl.org

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 5 redirect



```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 5 < ../data_in/fanconi_genes.csv
IN Example 5:
entrez_gene_symbol,entrez_gene_id
FANCA,2175
FANCB,2187
FANCC,2176
FANCD1,675
FANCD2,2177
FANCE,2178
FANCF,2188
FANCG,2189
FANCI,55215
FANCJ,83990
FANCL,55120
FANCM,57697
FANCN,79728
FANCO,5889
FANCP,84464
OUT Example 5.
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 5 < ../data_in/fanconi_genes.csv > copy_fanconi_genes.csv
[yshen16@scc4 script]$ diff copy_fanconi_genes.csv ../data_in/fanconi_genes.csv
1d0
< IN Example 5:
18d16
< OUT Example 5.
[yshen16@scc4 script]$
```



www.perl.org

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 5 redirect

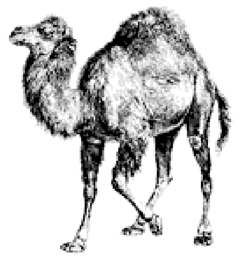
www.perl.org

What extra we learn from Example 5:

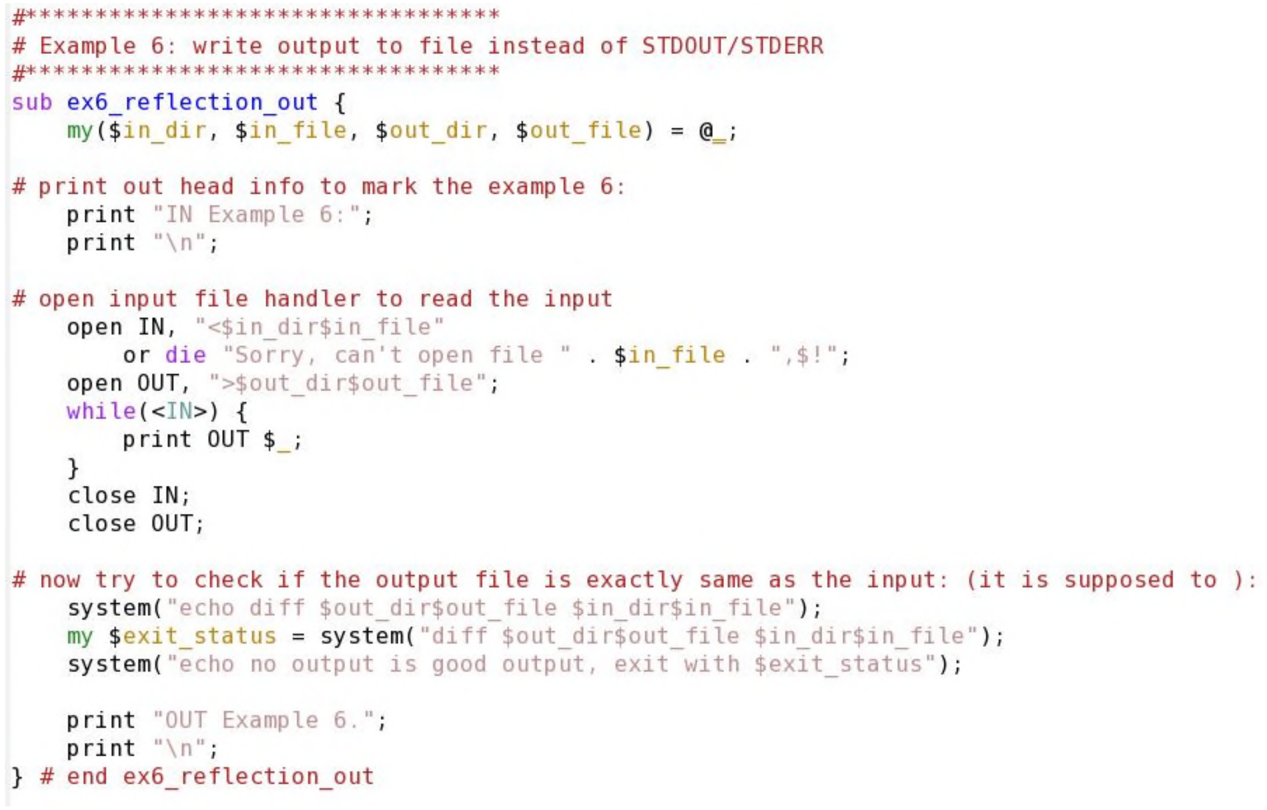
Redirection is a very powerful mechanism in Linux;
It can make code a lot more flexible;
It is built upon the concept of Linux/Unix fundamentals –
everything is a file

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 6 code (output to file)



www.perl.org

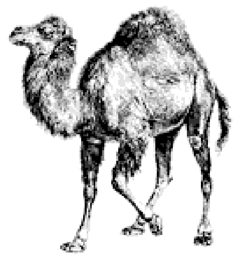


```perl
#***********************************
# Example 6: write output to file instead of STDOUT/STDERR
#***********************************
sub ex6_reflection_out {
    my($in_dir, $in_file, $out_dir, $out_file) = @_;

# print out head info to mark the example 6:
    print "IN Example 6:";
    print "\n";

# open input file handler to read the input
    open IN, "<$in_dir$in_file"
        or die "Sorry, can't open file " . $in_file . ",$!";
    open OUT, ">$out_dir$out_file";
    while(<IN>) {
        print OUT $_;
    }
    close IN;
    close OUT;

# now try to check if the output file is exactly same as the input: (it is supposed to ):
    system("echo diff $out_dir$out_file $in_dir$in_file");
    my $exit_status = system("diff $out_dir$out_file $in_dir$in_file");
    system("echo no output is good output, exit with $exit_status");

    print "OUT Example 6.";
    print "\n";
} # end ex6_reflection_out
```

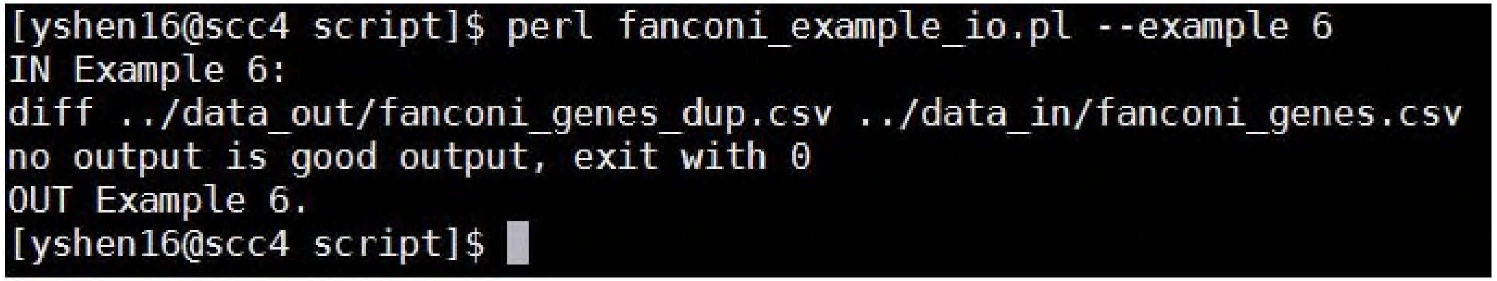
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 6 output



www.perl.org

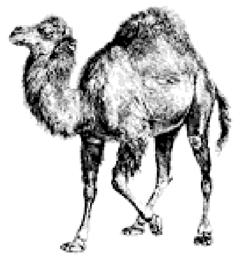


```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 6
IN Example 6:
diff ../data_out/fanconi_genes_dup.csv ../data_in/fanconi_genes.csv
no output is good output, exit with 0
OUT Example 6.
[yshen16@scc4 script]$ 
```

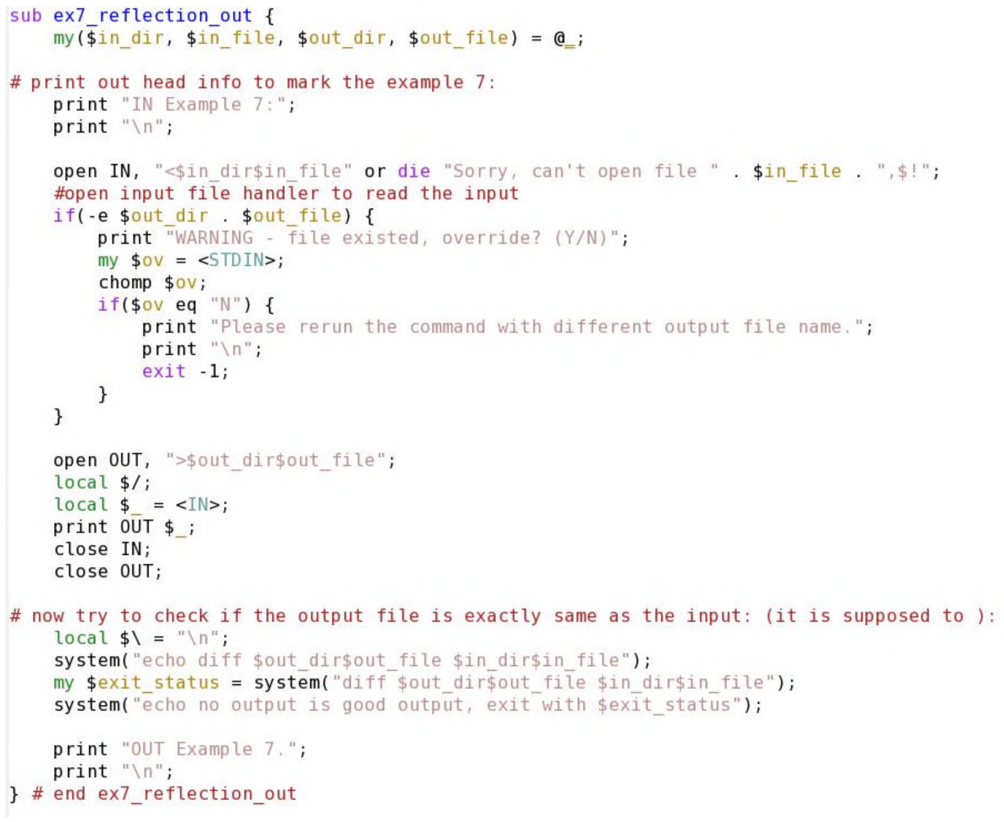
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

File IO – Example 7 code (special variables)



www.perl.org



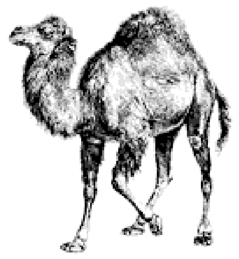
```
sub ex7_reflection_out {
    my($in_dir, $in_file, $out_dir, $out_file) = @_;

# print out head info to mark the example 7:
    print "IN Example 7:";
    print "\n";

    open IN, "<$in_dir$in_file" or die "Sorry, can't open file " . $in_file . ",$!";
    #open input file handler to read the input
    if(-e $out_dir . $out_file) {
        print "WARNING - file existed, override? (Y/N)";
        my $ov = <STDIN>;
        chomp $ov;
        if($ov eq "N") {
            print "Please rerun the command with different output file name.";
            print "\n";
            exit -1;
        }
    }

    open OUT, ">$out_dir$out_file";
    local $/;
    local $_ = <IN>;
    print OUT $_;
    close IN;
    close OUT;

# now try to check if the output file is exactly same as the input: (it is supposed to ):
    local $\ = "\n";
    system("echo diff $out_dir$out_file $in_dir$in_file");
    my $exit_status = system("diff $out_dir$out_file $in_dir$in_file");
    system("echo no output is good output, exit with $exit_status");

    print "OUT Example 7.";
    print "\n";
} # end ex7_reflection_out
```

BOSTON UNIVERSITY

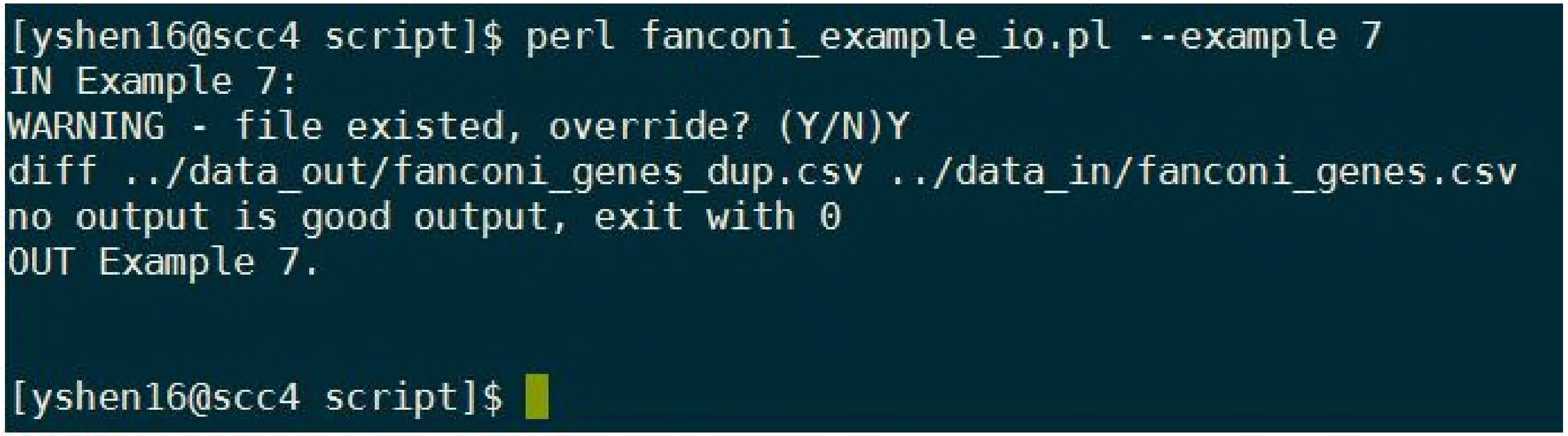
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

File IO – Example 7 output



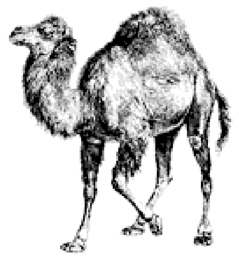
www.perl.org



```
[yshen16@scc4 script]$ perl fanconi_example_io.pl --example 7
IN Example 7:
WARNING - file existed, override? (Y/N)Y
diff ../data_out/fanconi_genes_dup.csv ../data_in/fanconi_genes.csv
no output is good output, exit with 0
OUT Example 7.


[yshen16@scc4 script]$ 
```

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Learn Through Example - RegEx

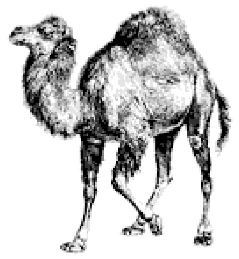
fanconi_example_str_process.pl

BOSTON UNIVERSITY

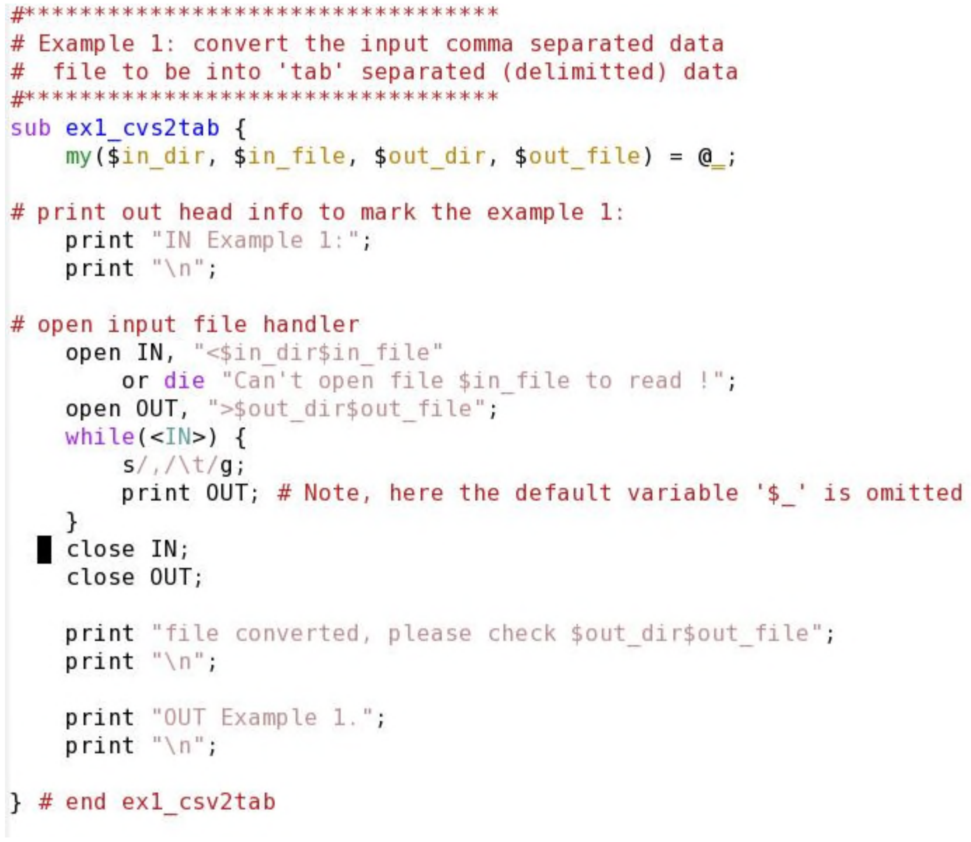
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

RegEx – String Process Example 1 code



www.perl.org

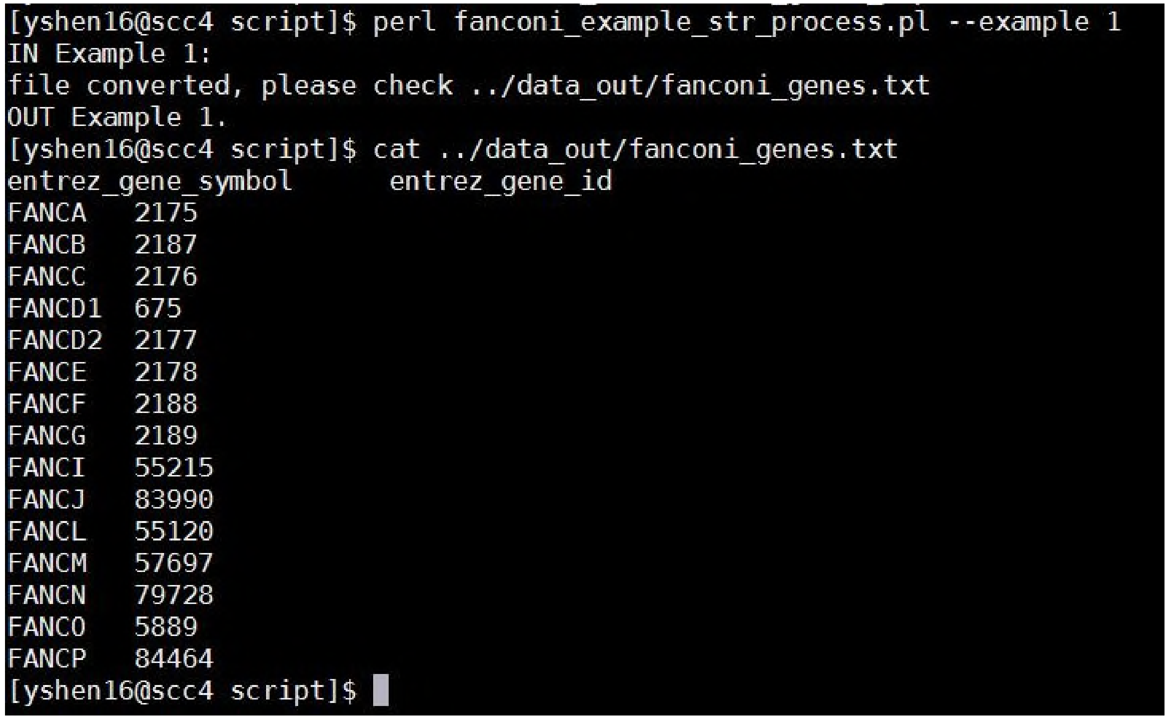


```
#***********************************
# Example 1: convert the input comma separated data
#  file to be into 'tab' separated (delimitted) data
#***********************************
sub ex1_cvs2tab {
    my($in_dir, $in_file, $out_dir, $out_file) = @_;

# print out head info to mark the example 1:
    print "IN Example 1:";
    print "\n";

# open input file handler
    open IN, "<$in_dir$in_file"
        or die "Can't open file $in_file to read !";
    open OUT, ">$out_dir$out_file";
    while(<IN>) {
        s/,/\t/g;
        print OUT; # Note, here the default variable '$_' is omitted
    }
    close IN;
    close OUT;

    print "file converted, please check $out_dir$out_file";
    print "\n";

    print "OUT Example 1.";
    print "\n";

} # end ex1_csv2tab
```

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

RegEx – String Process Example 1 output

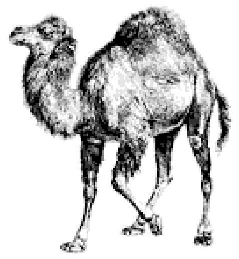


```
[yshen16@scc4 script]$ perl fanconi_example_str_process.pl --example 1
IN Example 1:
file converted, please check ../data_out/fanconi_genes.txt
OUT Example 1.
[yshen16@scc4 script]$ cat ../data_out/fanconi_genes.txt
entrez_gene_symbol      entrez_gene_id
FANCA   2175
FANCB   2187
FANCC   2176
FANCD1  675
FANCD2  2177
FANCE   2178
FANCF   2188
FANCG   2189
FANCI   55215
FANCJ   83990
FANCL   55120
FANCM   57697
FANCN   79728
FANCO   5889
FANCP   84464
[yshen16@scc4 script]$
```

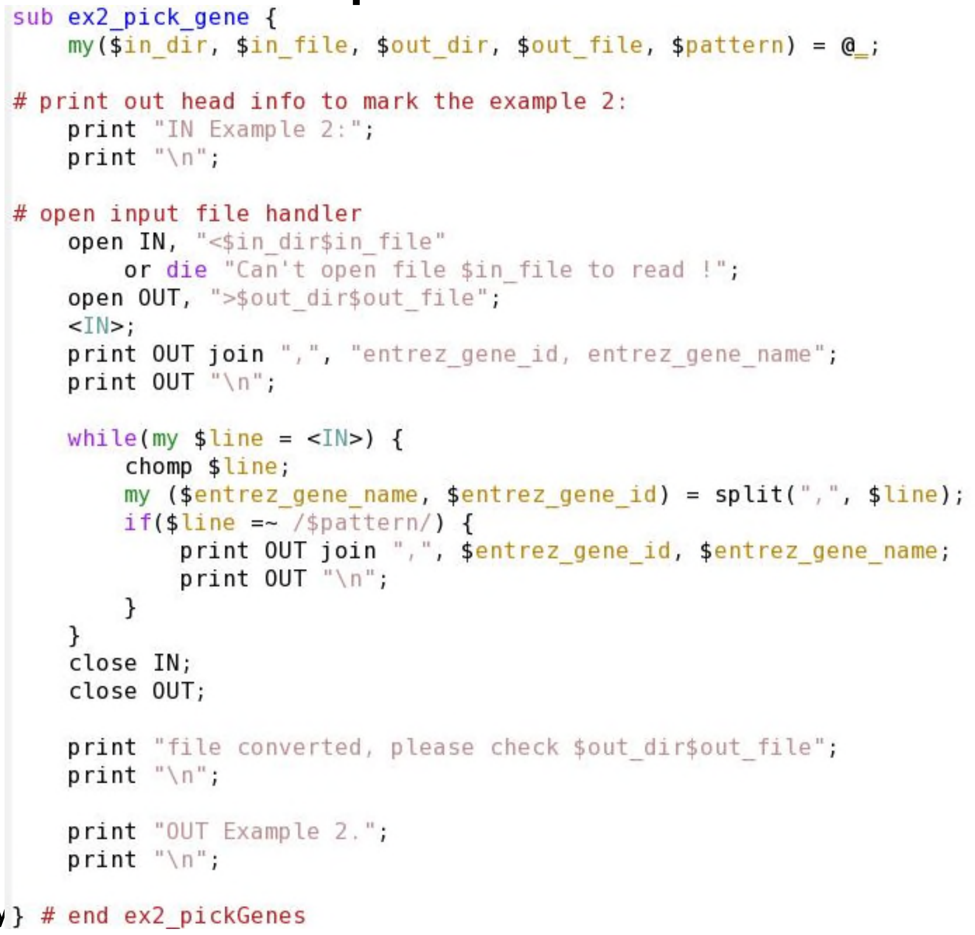
www.perl.org

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

RegEx – String Process Example 2 code



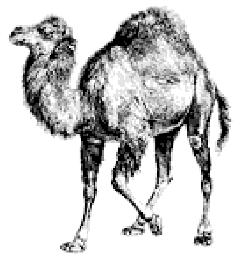
www.perl.org



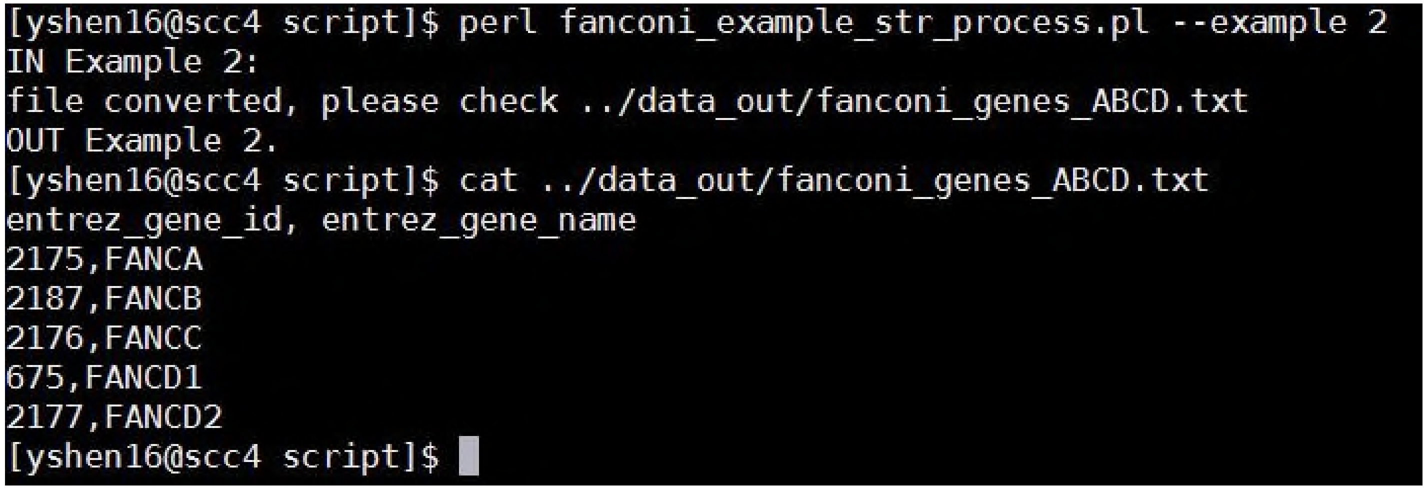
```
sub ex2_pick_gene {
    my($in_dir, $in_file, $out_dir, $out_file, $pattern) = @_;

# print out head info to mark the example 2:
    print "IN Example 2:";
    print "\n";

# open input file handler
    open IN, "<$in_dir$in_file"
        or die "Can't open file $in_file to read !";
    open OUT, ">$out_dir$out_file";
    <IN>;
    print OUT join ",", "entrez_gene_id, entrez_gene_name";
    print OUT "\n";

    while(my $line = <IN>) {
        chomp $line;
        my ($entrez_gene_name, $entrez_gene_id) = split(",", $line);
        if($line =~ /$pattern/) {
            print OUT join ",", $entrez_gene_id, $entrez_gene_name;
            print OUT "\n";
        }
    }
    close IN;
    close OUT;

    print "file converted, please check $out_dir$out_file";
    print "\n";

    print "OUT Example 2.";
    print "\n";

} # end ex2_pickGenes
```

Yun Shen, Programmer Analy
yshen16@bu.edu
IS&T Research Computing Services

RegEx – String Process Example 2 output

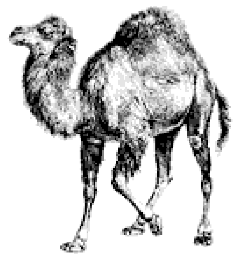


www.perl.org



```
[yshen16@scc4 script]$ perl fanconi_example_str_process.pl --example 2
IN Example 2:
file converted, please check ../data_out/fanconi_genes_ABCD.txt
OUT Example 2.
[yshen16@scc4 script]$ cat ../data_out/fanconi_genes_ABCD.txt
entrez_gene_id, entrez_gene_name
2175,FANCA
2187,FANCB
2176,FANCC
675,FANCD1
2177,FANCD2
[yshen16@scc4 script]$
```

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services



www.perl.org

Learn Through Example - BioPerl

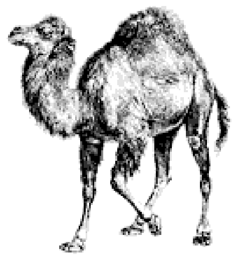
fanconi_example_gene_anno.pl

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017

Fanconi_example_gene_anno.pl structure



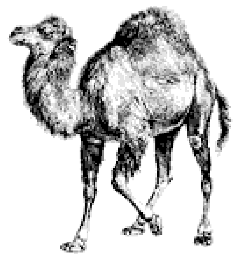
www.perl.org

1. Set up environment;
2. Open input file and get the gene list of interest
3. Initalize EntrezGene factory object
4. Call get_Stream_by_id() to fetch gene annotation info through gene id info
5. Go through the iteration of each gene
6. In each iteration, parse all the annotation attributes returned and print out result
7. Close file handle and exit.

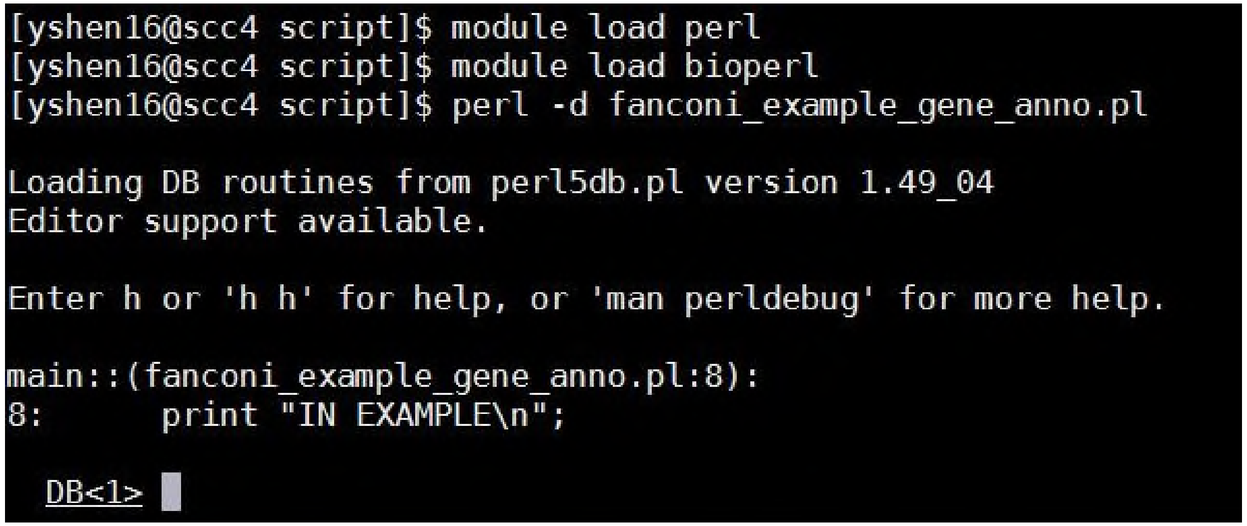
Next, we will go through it step by step…

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Load Perl on SCC



www.perl.org

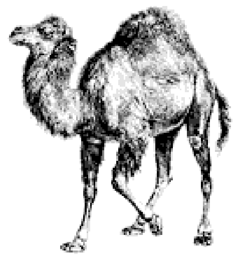


```
[yshen16@scc4 script]$ module load perl
[yshen16@scc4 script]$ module load bioperl
[yshen16@scc4 script]$ perl -d fanconi_example_gene_anno.pl

Loading DB routines from perl5db.pl version 1.49_04
Editor support available.

Enter h or 'h h' for help, or 'man perldebug' for more help.

main::(fanconi_example_gene_anno.pl:8):
8:      print "IN EXAMPLE\n";

  DB<1> 
```

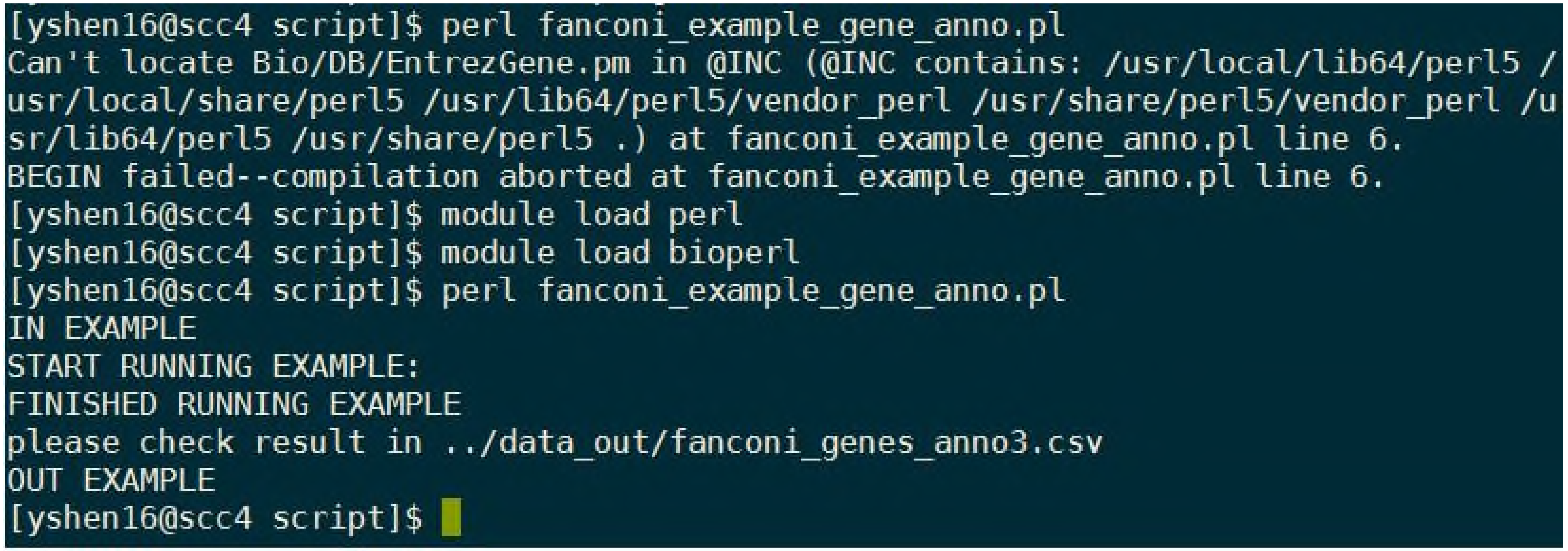
Notes, we will use Bio::DB:EntrezGene module in BioPerl suites, which is only available on SCC. So we need the two ‘module load’ commands;
Start Perl debugger by add ‘-d’ command option;

BOSTON UNIVERSITY
Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Preview the Result – screen output



www.perl.org

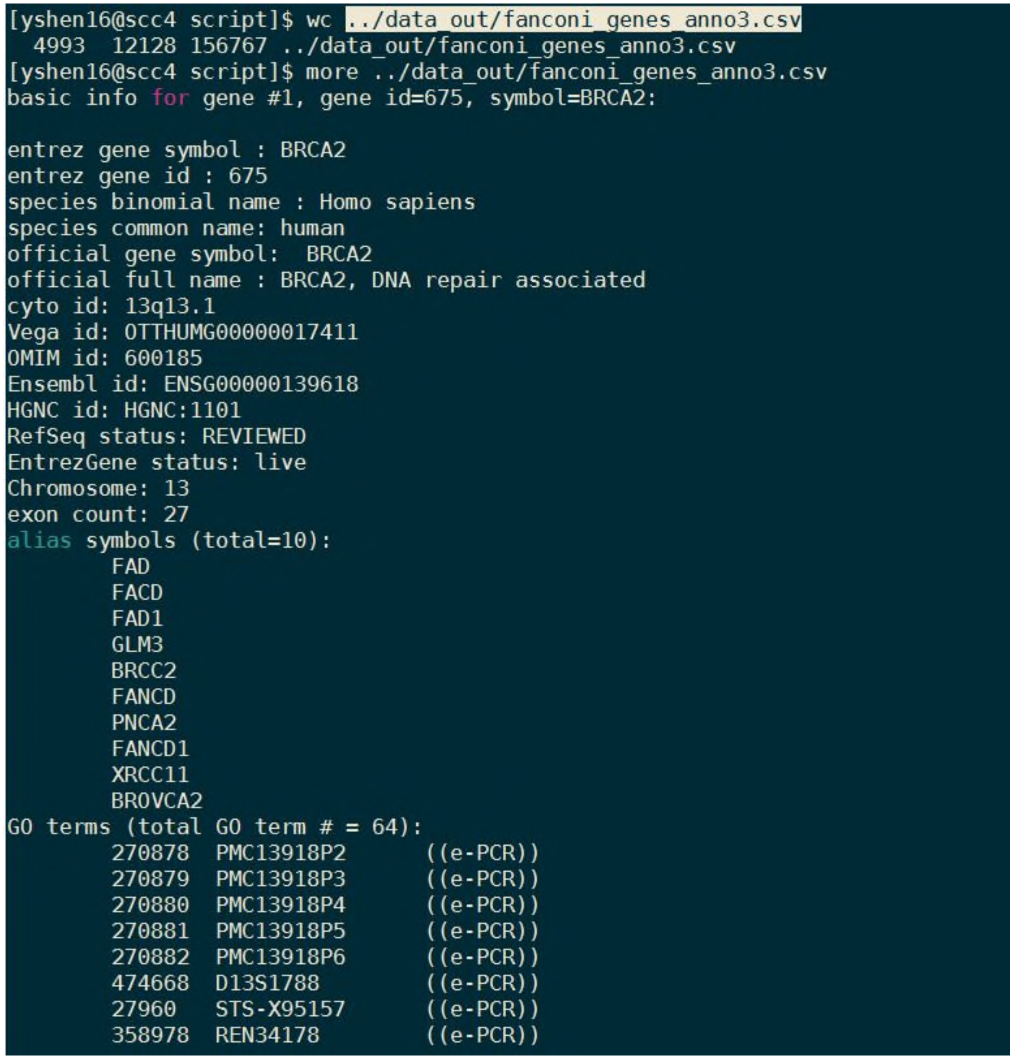


```
[yshen16@scc4 script]$ perl fanconi_example_gene_anno.pl
Can't locate Bio/DB/EntrezGene.pm in @INC (@INC contains: /usr/local/lib64/perl5 /
usr/local/share/perl5 /usr/lib64/perl5/vendor_perl /usr/share/perl5/vendor_perl /u
sr/lib64/perl5 /usr/share/perl5 .) at fanconi_example_gene_anno.pl line 6.
BEGIN failed--compilation aborted at fanconi_example_gene_anno.pl line 6.
[yshen16@scc4 script]$ module load perl
[yshen16@scc4 script]$ module load bioperl
[yshen16@scc4 script]$ perl fanconi_example_gene_anno.pl
IN EXAMPLE
START RUNNING EXAMPLE:
FINISHED RUNNING EXAMPLE
please check result in ../data_out/fanconi_genes_anno3.csv
OUT EXAMPLE
[yshen16@scc4 script]$ 
```

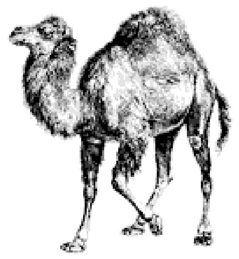
BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Preview the Result – how output file looks like

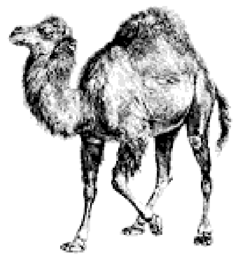


```
[yshen16@scc4 script]$ wc ../data_out/fanconi_genes_anno3.csv
  4993  12128 156767 ../data_out/fanconi_genes_anno3.csv
[yshen16@scc4 script]$ more ../data_out/fanconi_genes_anno3.csv
basic info for gene #1, gene id=675, symbol=BRCA2:

entrez gene symbol : BRCA2
entrez gene id : 675
species binomial name : Homo sapiens
species common name: human
official gene symbol:  BRCA2
official full name : BRCA2, DNA repair associated
cyto id: 13q13.1
Vega id: OTTHUMG00000017411
OMIM id: 600185
Ensembl id: ENSG00000139618
HGNC id: HGNC:1101
RefSeq status: REVIEWED
EntrezGene status: live
Chromosome: 13
exon count: 27
alias symbols (total=10):
        FAD
        FACD
        FAD1
        GLM3
        BRCC2
        FANCD
        PNCA2
        FANCD1
        XRCC11
        BROVCA2
GO terms (total GO term # = 64):
        270878  PMC13918P2      ((e-PCR))
        270879  PMC13918P3      ((e-PCR))
        270880  PMC13918P4      ((e-PCR))
        270881  PMC13918P5      ((e-PCR))
        270882  PMC13918P6      ((e-PCR))
        474668  D13S1788        ((e-PCR))
        27960   STS-X95157      ((e-PCR))
        358978  REN34178        ((e-PCR))
```



www.perl.org

Fall 2017



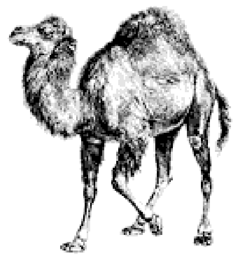
www.perl.org

Q & A

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017



www.perl.org

Evaluation Please @

http://scv.bu.edu/survey/tutorial_evaluation.html

Thank You !!

BOSTON UNIVERSITY

Yun Shen, Programmer Analyst
yshen16@bu.edu
IS&T Research Computing Services

Fall 2017